Universidad de las Ciencias Informáticas

Facultad 3

Centro CEIGE

Título: Sistema para la Gestión de la Información de la UJC en la Facultad 3.

Trabajo final presentado en opción al título de Ingeniero en Ciencias Informáticas

Autor (es): Thais Sánchez Martínez

Rolando Quintana Safont

Tutor (es): Ing. Leodanny W. Polanco Garay

Ing. Josué Rivera Riquenes

Ciudad de La Habana, junio de 2015.

"Año 57 de la Revolución"

"… hay que tener temple para ser un Joven Comunista, hay que tener carácter para ser un Joven Comunista, hay que tener abnegación para ser un Joven Comunista, hay que tener vocación para ser un Joven Comunista, hay que saber cumplir. Si se es estudiante, hay que ser inexorablemente buen estudiante; si se es trabajador de una fábrica, hay que ser obrero modelo en esa fábrica; hay que ser ejemplo de buen compañero, hay que ser ejemplo de sacrificio, hay que ser ejemplo de voluntad; han de ser los primeros en todo, en el trabajo, en el estudio, en los deportes, en la vida de relación con los demás compañeros. "

Fidel Castro

DEDICATORIA

A mi mamá, mi papá y mi abuela por ser mi ejemplo y guía.

A mis hermanitas por ser las personas que más quiero en la vida.

A mi familia por todo su apoyo y cariño.

A mi novio por todo lo que hemos vivido.

A bichitín que está por nacer.

A los que alguna vez dudaron de que si podía.

Thais

Thais

A mis padres por quererme tanto, brindarme su apoyo en todo momento y hacer realidad mis sueños como profesional. Los quiero mucho, gracias por guiarme siempre por un buen camino.

A mi novio Yunior por confiar siempre en mí, brindarme su apoyo incondicional y estar a mi lado en todo momento. Te quiero bichito.

A mi familia, especialmente a mi abuela, que de una u otra forma han influido en mi educación y me han apoyado en mis decisiones.

A mis compañeros del 3102: Juan Ignacio, Carlos Miguel, Elizabeth y Rachel, por su apoyo en los momentos difíciles. Y a los del 3503: Felix, Jorge Ariel, Jose Ángel y Aniel, por aguantarme.

A Yasmary por ser una buena amiga y hacer que estos años sean inolvidables.

A mi compañero de tesis Rolando por su esfuerzo y dedicación.

A los profes que me ayudaron a llegar hasta aquí: Alejandro, Esther Cristina, Haydee, Hugo, Daimara, Lizandra, Maigret, Yoel y Mavis.

A mi tutor Josué por ser uno más del equipo. Gracias por las noches en vela.

A mis otros tutores, Maylevis y Adrián por las horas extras que me dedicaron.

Y a Polanco por su apoyo y consejos durante estos 5 años.

Rolando

El presente trabajo es dedicado en especial a mi madre Mercy B. Safont Furet, por su amor, su dedicación, por ser luz cuando ya el camino se tornaba oscuro. A ti mamita linda, he aquí tu sueño convertido en realidad, he aquí el resultado de todos los esfuerzos que has hecho, por no dormir cuando yo no lo hacía, por siempre guiarme cuando más perdido estaba, gracias mamá.

AGRADECIMIENTOS

Dedicado también a mi hermano Ernesto Quintana Safont , por ser un amigo especial, por contar siempre conmigo y por siempre darme confianza y ser mi guía a seguir, por cuidar a mami mientras no estuve y por quererme como sabes que te quiero a ti.

A toda mi familia que siempre me ha apoyado y preocupado por mí, a mis vecinos, mis amigos, amigas, que a todos los quiero.

En fin dedicado a todos aquellos que siempre han estado presentes en mi vida.

DECLARACIÓN JURADA 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 firmamos la presente a los ____ días del mes de junio del año 2015.

Thais Sánchez Martínez	Rolando Quintana Safont
Ing. Leodanny W. Polanco Garay	Ing. Josué Rivera Riquenes

RESUMEN:

En la dirección de las organizaciones de masas, sociales o políticas en el mundo, se realiza la gestión de la información entre otros procesos. La calidad de esta es un reto imponente en el quehacer diario de cada una de ellas. En la actualidad existen diferentes organizaciones de gran importancia que requieren de un estricto manejo de su información por ser la misma de relevancia para el país. Los procesos de gestión dentro de estas son realizados de forma tradicional y no con la calidad requerida. Por esta razón surge la necesidad de esta investigación, para proveer a las organizaciones como la Unión de Jóvenes Comunistas, de un sistema que gestione su información. Por tal motivo, en esta investigación se estudian herramientas utilizadas para la gestión de la información en las organizaciones y se realiza un análisis de los principales conceptos asociados, estudio que permitió llevar a cabo el modelado del negocio así como definir los requisitos funcionales y no funcionales necesarios para la gestión de los procesos de la UJC.

Palabras claves:

Gestionar, organizaciones

Índice

Índice

Introducción

La gestión de la información es el proceso mediatizado por un conjunto de actividades que permiten la obtención de información, lo más pertinente, relevante y económicamente posible, para ser usada en el desarrollo y el éxito de una organización y que genera nuevos conocimientos (Ponjuán, 1998). El proceso de gestión de la información en las organizaciones políticas del país es amplio y realizado por personas que se encargan de recopilar la información procedente de sus homólogos en la base. Esta información debe ser revisada, auditada, analizada y comparada con los datos archivados para por último, almacenarla para próximas consultas que se quieran realizar.

En la Facultad 3 de la Universidad de las Ciencias Informáticas, la información se entrega digitalmente, pero los métodos para el procesamiento y gestión de la misma son los tradicionales y no los más correctos. Cada documento se recibe por correo electrónico o desde la red. Luego se procede a auditar cada documento.

El documento recibido depende del tipo de reunión realizada e información dada. Siendo así que se reciben actas ordinarias y extraordinarias. Pueden recibirse además otros documentos como informes (de estudio sociopolítico, de asambleas, etc.), actas de procesos de crecimientos, de asambleas de ejemplares, etc.

Los documentos de actas ordinarias son las que proceden de las reuniones mensuales de cada comité de base (CB). En la facultad se cuenta con un total de 34 organizaciones de base (OB) por lo que como promedio se deben procesar 34 actas mensualmente. A partir de estos documentos se generan los cómputos del Comité Primario (CP) y se actualizan todas las plantillas del control de la militancia que hayan notificado cambios, en conjunto con el ID2 (Plantilla de control de toda la militancia). Además se pueden recibir documentos extraordinarios procedentes de las reuniones extraordinarias realizadas en los CB. No necesariamente se reciben documentos de todas las OB, sin embargo en un caso crítico esto puede ocurrir.

Independientemente de la recepción de las actas, la calidad de las mismas se ve afectada debido a que los secretarios de actas en muchas ocasiones no cuentan con los modelos adecuados para el tipo de reunión que están realizando y por la variación reiterada de los mismos. Estos documentos deben soportar toda la información que se debata en los encuentros de los CB, por lo que existen diversos modelos en dependencia de los puntos que se desarrollen en cada reunión y del tipo de reunión que se efectúe. Todas las actas y modelos deben perdurar en el tiempo debido a las

consultas que se puedan realizar sobre ellas. De las actas además se generan diversos reportes que tributan al correcto funcionamiento de la organización.

El CP genera de forma manual estos reportes que constituyen el listado de acuerdos tomados en cada una de las reuniones realizadas en los CB, así como la reseña de acta encargada de guardar los principales planteamientos de las mismas, reportes sobre sanciones, rendiciones de cuentas, entre otros. Sobre las reuniones ordinarias se genera el listado de ausentes, de presentes, por cientos de asistencia, causas de los ausentes justificados, la participación de estructuras superiores, etc. De las reuniones extraordinarias además de los mismos elementos de las reuniones ordinarias se debe de recoger las causas de las reuniones extraordinarias.

El equipo de funcionamiento del CP realiza además la planificación de las reuniones y notifica a cada secretario de CB el lugar, local y hora de su reunión, aspectos todos que en ocasiones se ven afectados por el olvido de la notificación a un secretario o la duplicidad en los horarios o locales al momento de realizar sus reuniones los CB.

Dentro de los documentos que se gestionan en la organización se encuentra la Plantilla de control de la militancia (ID2), que es el documento rector que controla los militantes que integran un CB. En ocasiones se producen traslados, altas o bajas en un CB que no son notificadas al organismo superior, lo cual trae consigo errores en la asistencia a la reunión provocando ausentes injustificados que no debieran existir. El ID2 es el documento donde se asientan las altas y bajas de los militantes y de los CB, así como de los movimientos que se realicen con estos.

Por la gestión de toda la documentación que se realiza en el CP se evidencia que en un caso crítico es necesario revisar más de 600 archivos mensualmente. En esta recopilación de datos existe gran dificultad para el trabajo y manipulación de la información, obteniéndose al final resultados de baja calidad. Esta descentralización de la información también se ve evidenciada en la evaluación de tareas asignadas a diferentes niveles, provocando que exista más de un responsable de la información y que se archive en diferentes lugares, impidiendo el acceso a la misma de ser necesario y retrasando así los análisis estadísticos, que de manera muy periódica se ven afectados debido a la gran cantidad de documentos que son necesarios gestionar por períodos ya que la información es difícil de procesar al encontrarse descentralizada, una parte en formato duro y otra en formato digital. Por otra parte la información archivada en formato duro, constantemente sufre deterioro por la periodicidad y cantidad de personas que trabajan con los documentos.

Por lo anteriormente expuesto, se plantea el siguiente problema a resolver: ¿Cómo gestionar la información asociada a los procesos de la UJC, de manera que aumente la calidad de la información y contribuya a la toma de decisiones en el Comité Primario de la Facultad 3?

Objeto de estudio: Proceso de gestión de información de la UJC.

Campo de acción: Informatización de la gestión de los procesos de la UJC en la Facultad 3.

Objetivo General: Desarrollar un sistema para la gestión de la información de la Unión de Jóvenes Comunistas en la Facultad 3 que favorezca el aumento de la calidad de la información y contribuya a la toma de decisiones en el Comité Primario.

Objetivos Específicos:

- Construir el marco teórico referencial de la investigación a partir del estudio de las particularidades intrínsecas en los sistemas de información que soportan funcionamiento de organizaciones pequeñas con dirección vertical.
- Diseñar una propuesta de solución que responda a las necesidades de gestión de información de la UJC en la Facultad 3.
- Implementar un sistema informático que responda a las características del diseño planteado.
- Validar la investigación a partir de la verificación y validación de la solución propuesta mediante pruebas de caja blanca y caja negra.

Se esperan como **posibles resultados**: Obtener un sistema para la gestión de la información de la Unión de Jóvenes Comunistas en la Facultad 3 que favorezca el aumento de la calidad de la información y contribuya a la toma de decisiones en el Comité Primario.

Marco conceptual:

- Militante: miembro activo de una organización.
- UJC: Unión de Jóvenes Comunistas. Organización que agrupa a jóvenes entre 14 y 32 años de edad que poseen cualidades especiales.
- PCC: Partido Comunista de Cuba.
- Universo Juvenil: todo joven en edad para ser militante activo que no forma parte de la UJC.
- Estructuras de la UJC: partes en las que se organizan los miembros de la UJC en diferentes niveles para una mayor organización.

- ID2: Plantilla de Control de la Militancia
- Cómputo: documento que recoge los datos de todas las reuniones realizadas en un Comité Primario.

Métodos de investigación

1. Métodos teóricos

   - Método histórico-lógico: Analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes periodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales. Este método se empleó para la fundamentación y sistematización de los aspectos teóricos contemplados en el desarrollo de la investigación acerca de las organizaciones políticas, la gestión de la información, las metodologías a utilizar, entre otros temas.

2. Métodos empíricos

   - La entrevista: La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos o sobre características personales del entrevistado y puede influir en determinados aspectos de la conducta humana por lo que es importante una buena comunicación. Se utilizó la entrevista individual de contenido investigativo.
   - Método de la modelación: La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. La necesidad práctica para la cual se ejecuta la modelación y la posible solución del problema de investigación da la medida en que se logra dicha relación, la que es determinada por el sujeto escogiendo una alternativa de acuerdo con su criterio. Este método se empleó para representar mediante diagramas algunos aspectos necesarios para una mejor compresión del negocio como el modelo conceptual, las relaciones entre clases, etc.

A continuación se presenta la estructura del trabajo, el cual consta de tres capítulos y varios anexos que representan los diagramas que fueron generados en cada fase.

- Capítulo 1: En este capítulo se exponen los principales conceptos asociados al problema de investigación, se presentan los lenguajes de modelado y programación, así como las tecnologías y metodologías que se ajustan al desarrollo del trabajo, fundamentando su

selección sobre la base del estudio realizado. Además, se describen las soluciones similares existentes.

- Capítulo 2: En este capítulo se describe la propuesta del sistema, especificando el alcance del proyecto; se da a conocer los roles relacionados con el sistema informático, se realiza la especificación de los requerimientos funcionales a través de las historias de usuario y los no funcionales. Se realiza además la descripción del sistema propuesto.
- Capítulo 3: En este capítulo se describen las tres iteraciones llevadas a cabo durante la etapa de implementación del sistema, así como las pruebas efectuadas sobre el mismo y el manejo de la seguridad.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se realiza un estudio de forma valorativa y crítica acerca de los sistemas de gestión de información en cuanto a su definición y características, centrándose en el estudio de los sistemas de gestión de las organizaciones políticas y de masas a través del estudio del estado del arte, para analizar sus servicios y funcionalidades. Se efectuará un estudio de los sistemas similares, lenguajes de programación, herramientas, metodologías, sistemas gestores de bases de datos, entornos de desarrollo y sistemas de gestión de contenido posibles a utilizar para el desarrollo de la solución propuesta.

1.1- Sistemas de gestión de la información.

Un sistema de información se puede definir técnicamente como un conjunto de componentes relacionados que recolectan o recuperan, procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización. Hay tres actividades en un sistema de información que producen la información que esas organizaciones necesitan para tomar decisiones, controlar operaciones, analizar problemas y crear nuevos productos o servicios. Estas actividades son:

- Entrada: captura o recolecta datos en bruto tanto del interior de la organización como de su entorno.
- Procesamiento: convierte esa entrada de datos en una forma más significativa.
- Salida: transfiere la información procesada a la persona que la usará o a las actividades para las que se utilizará.

Los sistemas de información también requieren retroalimentación, que es la salida que se devuelve al personal adecuado de la organización para ayudarle a evaluar o corregir la etapa de entrada (Sonora, s.f.).

1.2- Análisis de sistemas informáticos vinculados al campo de acción

No existen antecedentes de sistemas a nivel internacional relacionados con la UJC; esta es una organización que solo radica en Cuba y en el ámbito nacional los sistemas que existen que pudieran constituir antecedentes de esta investigación, son sitios web en varios lugares, principalmente en las

universidades y joven club que se dedicaban solamente a informar y tener noticias actualizadas del funcionamiento de la organización, sin embargo se estudiaron algunos sistemas con propósitos similares de los cuales se realiza un análisis a continuación:

Internacional

- Sistema de Gestión Escolar Módulo Registro Académico: Fue creado con la finalidad de permitir el registro, procesamiento y recuperación de datos de los estudiantes que se manejan en cada centro Educativo de El Salvador, con el propósito de mecanizar el procesamiento de la matrícula, notas y misión de reportes de una manera efectiva y eficiente. Este módulo a pesar de llevar un registro histórico por alumnos, no se ajusta a los requerimientos de la UJC en la UCI y al ser una aplicación desktop (Escritorio) no cumple con los objetivos trazados en la organización, no obstante, se usará como guía para el control de los datos de los militantes.
- Federación de Estudiantes Universidad Técnica Federico Santa María Campus Santiago (FEUSAM): Este sitio tiene la obligación de dar solución a las problemáticas que tiene esta organización ya sean, las necesidades e intereses de los estudiantes, creando un espacio donde se generen debates logrando así una comunicación directa, a través de instancias democráticas, consultivas y representativas; permitiendo que se creen lazos con la dirección y los departamentos que conforman la universidad, para un desarrollo académico, cultural y político que enriquezca a la comunidad de espacios donde puedan desarrollar todo el potencial humano, participando activamente en la política nacional junto con las distintas federaciones universitarias del país.

En la Universidad de Ciencias Informáticas

- Análisis y diseño de una aplicación informática para la gestión de información de la UJC en la UCI: Es un trabajo de diploma que recoge el análisis y diseño de una aplicación informática para la gestión de información de la UJC en la UCI, con el propósito de que posteriormente sirviera como base para la implementación del mismo. Este trabajo se adopta por parte del proyecto y del cliente con la idea de realizar un nuevo diseño e incorporar algunas ideas y procesos fundamentales que se debían poner en práctica.
- Sistema Integral de Gestión de la UJC Nacional: Módulo Militante: Es un trabajo de diploma que presenta similitudes con el presente trabajo, sin embargo presenta características diferentes en cuanto a la gestión de las estructuras debido a la reciente actualización del reglamento y los estatutos de la organización. Además la aplicación que se propone como resultado será una aplicación Desktop, la cual no cumple con las expectativas del cliente.

- Sistema Informático para la Gestión de la Información de la UJC de la Facultad 2: Es un trabajo de diploma que presenta similitudes con el presente trabajo, sin embargo presenta diferencias en cuanto a las funcionalidades que se quieren gestionar y aunque fue presentado como proyecto de tesis en el 2009, hasta nuestros días no se tiene conocimiento sobre su implantación o puesta en práctica en la UCI, ni que se haya continuado su mantenimiento o soporte.
- Sistema de Control de Militantes para la Unión de Jóvenes Comunistas (SCM-UJC): Este sistema fue creado en el proyecto de la universidad nombrado “Kainos”. El Sistema de Control de Militantes para la Unión de Jóvenes Comunistas, fue probado en el municipio de La Lisa en La Habana y no fue de muy buena aceptación pues no abarcaba todas las necesidades para las cuales fue principalmente ideado. El SCM-UJC, es una aplicación de escritorio con base de datos en Access, que requiere para su instalación de la máquina virtual de Java, lo que hace lento este proceso. El sistema posee autenticación y realiza acciones según sean definidos los roles en la aplicación, permite la creación de estructuras de la UJC en cualquiera de los niveles. Presenta errores en la gestión de militantes, gestión de traslados y búsquedas de miembros en los CB, por lo que no es totalmente funcional, además de procesar lentamente las peticiones realizadas por el usuario.

Estos sistemas no cumplen con los requerimientos necesarios para la informatización de los procesos de la UJC ya que entre sus principales limitantes están que la mayoría son propietarios, lo cual va en contra del paradigma de independencia tecnológica por el cual apuesta el país y son muy costosos; se gestionan de manera diferente procesos fundamentales de la organización ya que luego del cambio del reglamento y de los estatutos, estos cambiaron también, al igual que algunos modelos de documentos que variaron su estructura y/o contenido y por último, algunos de estos sistemas no fueron funcionales y no cumplieron los objetivos para los que fueron previstos teniendo en cuenta que no se conoce prácticamente de su existencia. Tales motivos avalan la necesidad de crear un sistema que cumpla con los requerimientos necesarios para una buena gestión de los efectivos de la organización y el manejo de los datos referentes de cada uno de ellos y que se ajuste a las necesidades del cliente.

1.3- Aplicaciones web

Una aplicación web es un sistema informático enmarcado dentro de la arquitectura cliente-servidor. Su funcionamiento se basa en que un ordenador (el cliente) solicita servicios, los cuales son respondidos por otro ordenador (el servidor), el cual se encuentra en todo momento esperando

solicitudes provenientes de la web. En otras palabras, es una aplicación que facilita la interacción del cliente con un servidor web haciendo uso de un navegador y a la cual puede acceder mediante internet o intranet (Luján-Mora, 2002).

El objetivo de desarrollar el sistema como una aplicación web es facilitar su administración debido a que es un sistema de grandes dimensiones, además de que no existiría la necesidad de instalar el software a todos los usuarios, evitando ocupar espacio en discos duros y evitando también problemas de compatibilidad ya que estos solo necesitarían un navegador para acceder a la aplicación donde se van actualizando los cambios realizados, posibilitando un alto grado de comunicación.

1.4- Metodologías de desarrollo de software

“Se define metodología de desarrollo de software como un conjunto de procedimientos, técnicas, herramientas y un soporte documental para el desarrollo de productos de software” (Pressman, 2005). Una metodología puede seguir uno o varios modelos de ciclo de vida. El ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales.

Es importante que el proyecto llegue a término en el tiempo acordado con los clientes y con la calidad requerida. Por este motivo el éxito del producto final depende en gran medida de la metodología que se escoja por el equipo de desarrollo, teniendo en cuenta que unas se adaptan mejor que otras al contexto del proyecto brindando mejores ventajas (G. Figueroa , Solís, & Cabrera).

Actualmente las metodologías más utilizadas se dividen en Metodologías Ágiles y Metodologías Tradicionales.

Metodologías ágiles

Promueven iteraciones durante todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en poco tiempo, debido a que en la actualidad los entornos de los sistemas son muy cambiantes y se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Están especialmente orientadas para proyectos pequeños (Tejeda).

Entre las principales metodologías ágiles tenemos XP (eXtreme Programming), Scrum, Iconix, Crystal Clear, AUP (Agile Unified Process), entre otras. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan (G. Figueroa, Solís, & Cabrera).

Metodologías tradicionales

Son metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado. También reciben el nombre de Metodologías Pesadas, o Peso Pesado (Abrahamsson, 2002). Estas metodologías se centran en la definición detallada de los procesos mediante una estricta definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada del producto.

Entre las metodologías más conocidas y utilizadas que pertenecen a este grupo se encuentran:

- MSF (Microsoft Solution Framework)
- RUP (Rational Unified Process)

Metodologías Agiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Mas artefactos
Pocos roles	Mas roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Imagen 1.1- Comparación entre las metodologías ágiles y tradicionales

Después de haber estudiado las anteriores metodologías, el equipo de desarrollo decidió utilizar la metodología XP debido a que se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico. Además, ofrece como principales ventajas una programación organizada, menor taza de errores y mayor satisfacción del programador (Software, 2015).

1.5. Pruebas a realizar al sistema

Las pruebas son un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores, promueve la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Las pruebas permiten validar y verificar el software, entendiendo como validación del software, el proceso, externo al equipo de desarrollo, que determina si el software satisface los requisitos y verificación como el proceso interno que determina si los productos de una fase satisfacen las condiciones de dicha fase (Pressman, 2002).

El sistema, puede probarse de dos formas:

- conociendo la función específica para la que fue diseñado (pruebas de caja negra)
- conociendo el funcionamiento del producto (pruebas de caja blanca)

Según Pressman, las pruebas de caja negra se llevan a cabo sobre la interfaz del software. Se trata de demostrar que las funciones del software son operativas, que las entradas se manejan de forma adecuada y que se produce el resultado esperado. Las pruebas de caja negra buscan encontrar errores en cinco categorías (Pressman, 2002):

- Funciones incorrectas o ausentes;
- Errores de interfaz;
- Errores en estructuras de datos o en accesos a bases de datos externas;
- Errores de rendimiento;
- Errores de inicialización y terminación.

Las pruebas de caja blanca se centran en la estructura lógica interna del software. Se basan en un examen detallado de los procedimientos y caminos lógicos del sistema. Las principales técnicas de diseño de pruebas de caja blanca son:

- Pruebas de flujo de control
- Pruebas de flujo de datos
- Pruebas de bifurcación (branch testing)
- Pruebas de caminos básicos

Para la validación del sistema el equipo de desarrollo hará uso de las pruebas de caja negra aplicando la técnica de la partición de equivalencia y de caja blanca aplicando la técnica de caminos básicos.

A continuación se definirán los elementos principales de la arquitectura de la propuesta de solución especificando los estilos arquitectónicos a utilizar, los frameworks de desarrollo y patrones de diseño.

1.6- Arquitectura de Software

La Arquitectura de Software es la forma en la que se organizan y estructuran los componentes de un sistema, interactuando y relacionándose entre sí y con el contexto, aplicando normas y principios de diseño y calidad, que fortalezcan y fomenten la usabilidad a la vez que dejan preparado el sistema, para su propia evolución. De este modo, los ingenieros, analistas, diseñadores y programadores, pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo propuesto.

1.6.1- Estilos arquitectónicos

Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, que realiza una función requerida por el sistema, un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se pueden integrar los componentes que forman el sistema y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes (Reynoso & Kicillof, Marzo de 2004). Los estilos se clasifican en:

- Estilo de Flujo de Datos
    1. Tuberías y Filtros
- Estilos Centrados en Datos
    1. Arquitecturas de Pizarra o Repositorio
- Estilos de Llamada y Retorno
    1. Modelo-Vista-Controlador (MVC)
    2. Arquitecturas en Capas
    3. Arquitecturas Orientadas a Objetos

4. Arquitecturas Basadas en Componentes

- Estilos de Código Móvil

1. Arquitectura de Máquinas Virtuales

- Estilos heterogéneos

1. Sistemas de control de procesos
2. Arquitecturas Basadas en Atributos

- Estilos Peer-to-Peer

1. Arquitecturas Basadas en Eventos
2. Arquitecturas Orientadas a Servicios (SOA)
3. Arquitecturas Basadas en Recursos

Por parte del equipo de arquitectura se tuvo en cuenta el primer patrón del estilo de Llamada y Retorno: Modelo-Vista-Controlador. Es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario, tratando al modelo, las vistas y los controladores como entidades separadas; lo que hace posible que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas. Entre sus ventajas se encuentran:

- Soporte de vistas múltiples: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

- Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Entre las desventajas, se han señalado:

- Complejidad: El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del

código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por separado.

- Costo de actualizaciones frecuentes: Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas.

Se seleccionó la arquitectura MVC teniendo en cuenta las facilidades de adaptación al cambio que ofrece, además de estar diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. También al separar la vista de la lógica de negocio y el controlador, permite la reutilización de los mismos y si en algún momento uno de sus componentes funciona mal, puede reemplazarse sin que los demás se vean afectados.

1.6.2- Framework de desarrollo

Según el diccionario de Oxford, la palabra framework significa: Una estructura básica que subyace a un sistema, concepto o texto. Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Un framework facilita el desarrollo de software y evita los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software (Santa Fe, 1998 - 2015).

1.6.2.1- Symfony2

Este framework separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Es la versión más reciente de Symfony y ha sido ideado para explotar todas las nuevas características de PHP 5.3 y por eso es uno de los framework PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un proyecto. Symfony2 también es el framework que más ideas incorpora del resto de los framework, incluso de aquellos que no están programados con PHP (Eguiluz, 2011)

Entre sus características más importantes se tienen:

- Más rápido y menos codicioso (Alto rendimiento):Symfony2fueconcebidodesdeelprincipio para ser rápido y para favorecer el rendimiento como mayor prioridad, por lo que es uno de los framework más rápidos.
- Flexible: Symfony2 cuenta con un micro-kernel basado en un contenedor de inyección de dependencia y un manejador de eventos por lo que es muy fácil configurarlo a voluntad.
- Construido para desarrolladores: Symfony2 proporciona las herramientas que en gran medida mejoran la productividad de los desarrolladores, como la famosa barra de depuración web, soporte nativo de entornos, páginas detalladas de errores y mucho más.
- Construido en base a otros grandes framework: Symfony2 tomó lo mejor de los conceptos de otros framework de desarrollo como Django, Spring y Ruby on Rails. También aprovecha componentes de Zend Framework y de Doctrine.
- Open Source (Código abierto).
- Usabilidad avanzada: Es un framework fácil de utilizar gracias a que cuenta con una API (del inglés: Application Programming Interface) de desarrollo muy intuitiva.
- Extensible: Symfony2 se construye a base de bundles (plugins en Symfony 1).

Teniendo en cuenta lo anterior, se definió el uso de Symfony2 en la versión 2.5 para el desarrollo del sistema.

1.6.2.3- Twig

Twig es un motor y lenguaje de plantillas para PHP predeterminado de Symfony. Es muy rápido ya que compila las plantillas hasta código PHP regular optimizado y el costo general en comparación con código PHP regular se ha reducido al mínimo y es seguro pues tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable, lo que permite utilizarlo como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla. Además es flexible debido a que es alimentado por flexibles analizadores léxico y sintáctico, que permiten al desarrollador definir sus propias etiquetas y filtros personalizados (Pacheco, 2013)

Twig proporciona una serie de beneficios entre los que se encuentran:

- Plantillas fáciles de hacer y resultan muy intuitivas en el caso de contar con maquetadores o diseñadores de interfaces.

- Su sintaxis corta y concisa es muy similar a la de otros famosos framework como Django, Jinja, Ruby on Rails y Smarty.
- Implementa un novedoso mecanismo de herencia múltiple de plantillas.
- Gran rapidez: las plantillas son transformadas en PHP.
- El código se escribe en bloques, facilitando la herencia.
- Es extensible ya que permite la creación de extensiones propias. (Sensio Labs, 2010-2012)

Dentro del patrón de arquitectura MVC, la vista es la interfaz final del usuario encargada de representarlos datos del modelo gestionados por el controlador y su misión es la representación. Además define formatos de visualización agregando algunos assets[1] necesarios (Emagister Engineering, 2012).

Para la generación de las vistas del sistema se decidió usar Twig ya que cuando se utiliza el frameworkSymfony2 se recomienda su uso para crear todas las plantillas de la aplicación y también debido a sus grandes potencialidades.

1.6.2.4- Bootstrap

Bootstrap es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS (Cascading Style Sheets, u Hojas de Estilo en Cascada), así como, extensiones de JavaScript opcionales adicionales.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS 3, pero es compatible con la mayoría de los navegadores web. Desde laversión2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles).Bootstrap es de código abierto y está disponible en GitHub.

Ventajas

- Utiliza componentes y servicios creados por la comunidad web, tales como: HTML5 shim, Normalize.css, OOCSS, jQuery UI, LESS (lenguaje de hojas de estilo) y GitHub.
- Es un conjunto de buenas prácticas que perduran en el tiempo.
- Dentro de sus características más destacadas, podríamos mencionar:

1. La implementación de HTML5 + CSS3
2. Son archivos css, js, imágenes, textos y otros, necesarios para que la interfaz del usuario funcione correctamente.
3. El famoso Grid system, que por defecto incluye 12 columnas fijas o fluidas, dependiendo de si el diseño será responsivo o no.
4. El uso de LESS, que es una ampliación a las famosas hojas de estilo CSS, pero a diferencia de estas, funciona como un lenguaje de programación, permitiendo el uso de variables, funciones, operaciones aritméticas, entre otras, para acelerar y enriquecer los estilos en un sitio web.
5. Herramienta sencilla y ágil para construir sitios web e interfaces que ahorra realmente mucho tiempo de trabajo.
6. Además tiene un theme (tema) por defecto bastante optimizado y que se puede modificar o crear uno propio.

Desventajas

- Es necesario adaptarse a su forma de trabajo, si bien su curva de aprendizaje es liviana, se debe comprender y familiarizar con su estructura y nomenclatura.
- Se debe adaptar el diseño a un grid de 12 columnas, que se modifican según el dispositivo.
- Bootstrap por defecto trae anchos márgenes y altos de línea y realizar cambios específicos es un poco tedioso.
- Es complicado cambiar de versión si se han realizado modificaciones profundas sobre el core (microarquitectura de Intel en la que se basan algunos de sus microprocesadores).

Por lo expuesto anteriormente, se seleccionó este framework para el trabajo en la vista.

1.6.3- Mapeador Objeto-Relacional (ORM^2) - Doctrine

El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. Las bases de datos relacionales solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto, etc.) por lo que no se pueden guardar de forma directa los objetos de la aplicación en las tablas, sino que estos se deben de convertir antes en registros, que por lo general afectan a

varias tablas. En el momento de volver a recuperar los datos, hay que hacer el proceso contrario, se deben convertir los registros en objetos. Es entonces cuando el ORM cobra importancia ya que se encarga deforma automática de convertir los objetos en registros y viceversa, simulando así tener una base de datos orientada a objetos (Guardado, 2010).

Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas. No obstante los ORM traen consigo algunas desventajas como el tiempo invertido en el aprendizaje (Enríquez & Gracia del Busto, 2011).

Doctrine es una librería para PHP5.3.2 que permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos y no de tablas y registros. Doctrine está inspirado en Hibernate, que es uno de los ORM más populares y grandes que existen y brinda una capa de abstracción de la base de datos muy completa. La característica más importante es que ofrece la posibilidad de escribir consultas de base de datos en un lenguaje propio llamado Doctrine Query Language (DQL).Entre sus características más significativas están la posibilidad de trabajar con YAML, permite la generación automática del modelo, la facilidad de búsqueda y que posee un lenguaje propio entre otras (Doctrine| Marco de Desarrollo de la Junta de Andalucía).

Symfony en sus primeras versiones incluía dos ORM de forma oficial: Propel y Doctrine, pero Symfony2 solo incluye este último. Después de un análisis, el equipo de desarrollo decidió utilizar Doctrine en su versión 2.1 debido al conjunto de ventajas que este ofrece y por ser el ORM oficial de Symfony2.

1.6.4- Patrones de diseño orientados a objetos

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe la esencia de la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces (Alexander, 1997).Los patrones facilitan la reutilización de diseños y arquitecturas software que han tenido éxito (Gamma E., Software, & Addison-Wesley, 1995).

Los patrones de diseño se clasifican en dos grupos fundamentales: los patrones GRASP[3] (patrones generales de software para asignación de responsabilidades) y los patrones GoF[4]que es el nombre con el que se conoce por lo general a los autores del libro Design Patterns.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo contiene patrones como:

- Fabricación Pura
- Polimorfismo
- Bajo Acoplamiento
- Alta Cohesión
- Experto
- Creador
- Controlador

Los patrones de diseño GoF son 23 y se clasifican según su propósito en Creacionales, Estructurales y de Comportamiento y según su ámbito en Objeto y de Clase. Se decidió utilizar el patrón GoF estructural Decorador, el cual añade funcionalidades a objetos dinámicamente permitiendo no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. También se decide utilizar el patrón Inyección de dependencias. Este consiste en pasar a cada componente todo lo que necesita a través de sus constructores, métodos o campos. De los patrones GRASP se decide utilizar el Controlador, Creador, Experto, Bajo Acoplamiento y Alta cohesión pues son patrones que están definidos en el framework de desarrollo.

En los siguientes epígrafes se describen los lenguajes de programación utilizados en el desarrollo de la propuesta de solución tanto en el lado del cliente como del servidor.

1.7. Lenguajes de programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras. 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; es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones y que permite la comunicación entre el usuario y la computadora (Lutz, 2009)

1.7.1- PHP

PHP es un popular lenguaje de programación de propósito general del lado del servidor que es especialmente adecuado para el desarrollo web por ser rápido, flexible y pragmático (Group, 2012).Originalmente fue diseñado para el desarrollo web de contenido dinámico. Está actualmente

entre los proyectos de código abierto más populares (gracias en parte a la similitud de su sintaxis con el lenguaje C). El código es interpretado por un servidor web con un módulo procesador de PHP que genera la página Web resultante. Su funcionamiento es muy básico, cuando el cliente realiza una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP, este procesa el script solicitado que generará el contenido de manera dinámica y el resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente (código HTML).También mediante extensiones es posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Ventajas de este lenguaje:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, se destaca su conectividad con MySQL y PostgreSQL.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Tiene manejo de excepciones (desde PHP5).

1.7.2- jQuery

jQuery es una biblioteca de JavaScript, que contiene procesos o rutinas ya listos para ser usados, lo que permite simplificar la forma de interactuar con los documentos HTML. Facilita la búsqueda de elementos en un documento y su posterior manipulación ya sea por adición de contenido, edición de atributos HTML y propiedades CCS, definición de controladores de eventos o desarrollo de animaciones. Es software libre y de código abierto debido a que se encuentra registrado bajo la licencia MIT y la licencia pública general de GNU v2, lo que permite su uso en proyectos libres y privativos.

Principales características:

- Selección de elementos DOM (Document Objet Model).
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de Xpath (XML Path Language).
- Eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- AJAX (acrónimo de Asynchronous JavaScript And XML).
- Soporta extensiones.
- Utilidades varias 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.

jQuery es un producto muy beneficioso que cuenta con un buena aceptación por parte de los programadores y un grado de penetración en el mercado muy amplio, convirtiéndolo en una de las mejores opciones. Por otra parte, el archivo que contiene la librería solo ocupa unos 56 KB, lo que es bastante razonable y no retrasará mucho la carga de la página web (Álvarez M. Á., 2009).

1.8- Herramientas de desarrollo

A continuación se exponen las principales características, ventajas y desventajas de las herramientas seleccionadas para el desarrollo de la propuesta de solución.

1.8.1- Herramienta CASE[5]- Visual Paradigm for UML (Lenguaje Unificado de Modelado)

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. También proveen beneficios tales como:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.

- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (International, 2013).

Se caracteriza por:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Licencia gratuita y comercial.
- Soporta aplicaciones Web.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Soporte de UML versión 2.1.
- Diagramas de flujo de datos.
- Soporte ORM - Generación de objetos Java desde bases de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación entablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (SGBD) existentes a diagramas de Entidad-Relación.
- Generador de informes.

- Importación y exportación de ficheros XMI.

Se decidió utilizar la herramienta Visual Paradigm for UML en su versión 8.0 debido a las características antes mencionadas, así como por las ventajas que brinda la herramienta y por dos razones fundamentales: la primera razón es que la aplicación a desarrollar será escrita en PHP y esta herramienta se integra fácilmente con este lenguaje; la segunda razón es que el equipo de desarrollo cuenta con una mayor experiencia en el empleo de esta herramienta respecto a las otras, posibilitando un considerable ahorro de tiempo en el proceso de desarrollo del software.

1.8.2- Entorno de desarrollo Integrado (IDE[6]) - Netbeans

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).LosIDEproveenunmarcodetrabajoamigableparalamayoríadeloslenguajesdeprogramación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (fergarciac, 2013)

Netbeans es un entorno de desarrollo gratuito y de código abierto para el desarrollo de aplicaciones en los sistemas operativos Windows, Mac, Linux y Solaris. El IDE simplifica el desarrollo web de escritorio y aplicaciones móviles que usan plataforma Java y HTML 5. También ofrece soporte para el desarrollo de aplicaciones en PHP y C/C++ (Oracle, s.f.)

Características principales:

- Suele dar soporte a casi todas las novedades en el lenguaje Java.
- Buen editor de código, multilenguaje, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y semánticas, plantillas de código, herramientas de refactorización y mucho más.
- Herramientas para depurado de errores: el debugger que incluye el IDE es bastante útil para encontrar dónde fallan las cosas.
- Optimización de código: por su parte el Profiler nos ayuda a optimizar nuestras aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria.

- Acceso a base de datos: desde el propio Netbeans podemos conectarnos a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySql y demás y ver las tablas, realizar consultas y modificaciones y todo ello integrado en el propio IDE.
- Es fácilmente extensible a través de plugins (Genbeta:dev, desarrollo y software., 2014)
- La curva de aprendizaje es muy baja.
- Es más ágil y óptimo para los usuarios que no son expertos en manejo de consola.
- Formateo de código.
- Funciones para renombrar variables, funciones.
- Warnings (alertas) y errores de sintaxis en pantalla de algo que nova a funcionar al interpretar o compilar.
- Poder crear proyectos para poder visualizar los archivos de manera gráfica.

1.8.3- Sistema Gestor de Bases de Datos (SGBD) - PostgreSQL

Un sistema gestor de bases de datos es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, garantizando además la seguridad de los mismos (BaseDatosOfimaticas, 2015)

Un sistema gestor de base de datos está compuesto de:

- El gestor de la base de datos: Se trata de un conjunto de programas no visibles al usuario final que se encargan de la privacidad, la integridad, la seguridad de los datos y la interacción con el sistema operativo.
- Diccionario de datos: Es una base de datos donde se guardan todas las propiedades de la base de datos, descripción de la estructura, relaciones entre los datos, etc.
- El administrador de la base de datos: Es una persona o grupo de personas responsables del control del sistema gestor de base de datos.
- Los lenguajes: Un sistema gestor de base de datos debe proporcionar una serie de lenguajes para la definición y manipulación de la base de datos (Álvarez, 2007)

PostgreSQL es el sistema de gestión de bases de datos objeto-relacional de código abierto más potente del mercado, distribuido bajo licencia BSD y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizarla estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Características:

- Es una base de datos 100% ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad)
- Integridad referencial
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL
- Completa documentación
- Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de Oracle), PL/Perl, PL/Python y PL/Tcl
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido, etc.)
- Llaves primarias (primary keys) y foráneas (foreign keys)
- Columnas auto-incrementales
- Índices compuestos, únicos, parciales y funcionales en cualquiera de los métodos de almacenamiento disponibles, B-tree, R-tree, hash ó GiST
- Consultas recursivas
- Joins (uniones)
- Vistas (views)
- Disparadores (triggers) comunes, por columna, condicionales.

- Reglas (Rules)
- Herencia de tablas (Inheritance) (PostgreSQL-es, 2010)

1.8.4- Subversion 1.6 (SVN)

Subversion es una herramienta de control de versiones de código abierto basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD. Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer porque la calidad del mismo vaya a verse afectada si se ha hecho un cambio incorrecto a los datos, simplemente se deshace ese cambio (Apache Subversion, s.f.)

Esta versión crea ramas y fusiones más sólidas mediante la introducción de conflictos de árboles y ofrece mejoras en otras características existentes: opciones de resolución de conflictos más interactiva; definiciones externas basados en archivo y apoyo operacional de registro para svnserver similar al que ofreció mod_dav_svn. Además, el cliente de línea de comandos introduce una nueva sintaxis de acceso directo para hacer referencia a URLs de repositorio de Subversion (Ben Collins-Sussman)

1.8.5- Servidor Web - Apache 2.2

Un servidor web es un programa que sirve datos en forma de Páginas Web, hipertextos o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos. La comunicación de estos datos entre cliente y servidor se hace por medio un protocolo, concretamente del protocolo HTTP.

Apache es uno de los servidores web más utilizados, posiblemente porque ofrece instalaciones sencillas para sitios pequeños y si se requiere es posible expandirlo hasta el nivel de los mejores

productos comerciales. Además el servidor HTTP y de código abierto para las plataformas Windows, Mac OS X y UNIX (GNU, BSD, etc.). En cuanto a las características que posee Apache y que la llevó al éxito en la inserción y utilización en ámbitos empresariales, tecnológicos y educativos se encuentran:

- Fundamentalmente corre sobre una multitud de plataformas y sistemas operativos.
- Ofrece tecnología libre y de código abierto.
- Es un servidor Web configurable y de diseño modular, capaz de extender su funcionalidad y la calidad de sus servicios.
- Trabaja en conjunto con gran cantidad de lenguajes de programación interpretados como PHP, Perl, Java, JSP (Java Server Pages) y otros lenguajes de script (archivo de órdenes), que son el complemento ideal para los sitios web dinámicos.
- Es posible configurar y personalizar cada uno de los mensajes de error que se pueden producir por la utilización del servidor.
- Contar con los archivos log (registro oficial de eventos), en donde registra gran cantidad de información global del sistema, errores producidos en un determinado tiempo, en la cual estos archivos son de gran importancia para los administradores de sistemas y pueden influenciar de alguna manera las políticas de seguridad debido a la gran cantidad de información que contiene.
- Otra particularidad propia de Apache, es que al ser tan popular y utilizado, es posible encontrar gran cantidad de documentos, ejemplos y ayuda en internet en todos los idiomas.

Tener un servidor bajo apache es una solución sencilla, eficaz y rápida para tener nuestros sitios web funcionando al 100% sobre todo sin pagar un solo centavo. Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor http más usado. Alcanzó su máxima cuota de mercado en 2005siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años (Kabir, 2003).

1.9. Conclusiones parciales.

En este capítulo, basándose en el problema del trabajo y teniendo en cuenta que en la UJC de la Facultad 3 existen los requerimientos de hardware necesarios para instalar los servidores de aplicación y de bases de datos, se decidió desarrollar una aplicación web que permita implantar mejoras en el desarrollo de los procesos de la organización. Esta propuesta de solución facilitaría el

trabajo de los miembros del CP, disminuiría el tiempo a la hora de realizar búsqueda de información y se tendría una mejor organización de toda la documentación existente. Una vez decidido desarrollar este tipo de aplicación, se decide desarrollarla utilizando como lenguaje de programación PHP, como metodología de desarrollo XP, servidor web Apache y gestor de bases de datos PostgreSQL.

Capítulo 2: Descripción de la Solución Propuesta

Introducción

El presente capítulo describe los principales procesos implicados, así como sus actores, el modelo de negocio y el sistema, plasmando los principales requerimientos funcionales así como los no funcionales y llegando a una propuesta de solución.

2.1- Descripción de la solución propuesta.

La presente investigación propone la creación de una aplicación Web que provee en esencia las funcionalidades necesarias para la gestión de la información de los procesos de la UJC, además del perfeccionamiento de algunos de ellos, incorporado nuevos requisitos identificados.

La aplicación web estará programada en PHP usando el framework Symfony2, conservando la lógica de negocios actual desplegada en la BD PostgreSQL en el lenguaje PgSQL. El sistema debe permitir la autenticación de usuarios, otorgándose los debidos permisos de acuerdo a los roles que estos jueguen en la BD. Existen 6 roles: documentador, administrador, organizador, secretario, activista de acta y militante. El documentador, hará uso de la aplicación para la actualización y mantenimiento de la plantilla de la militancia así como de sus respectivos expedientes. El organizador, será el encargado de la gestión y confección de los horarios, orden del día y reportes de funcionamiento, así como de toda la información estadística. El secretario tiene un nomenclador que lo identifica, pues existen los secretarios de CB, los que poseen permisos totales sólo para su CB y el secretario del CP que tendrían acceso a la información de todos los CB de la facultad. Por su parte, el activista de acta, solo podrá hacer uso de la aplicación para la generación del acta de su CB y el militante solo podrá visualizar los datos de su expediente de militante. El administrador se encargará de la gestión de roles y permisos del sistema, así como de la configuración del mismo.

Entre las opciones que brinda el sistema se encuentran:

- Adicionar (dar alta), modificar y eliminar (dar baja) un militante de un CB introduciendo los datos del mismo en el caso que lo necesite.
- Adicionar, modificar o eliminar un CB del CP.
- La creación del acta especificando la fecha de realización, lugar de reunión, cantidad de presentes en el momento, por ciento de asistencia, lista de militantes ausentes seguido de sus

causas de ausencia, miembro del organismo superior que visita la reunión y cargo que ocupa el mismo, las opiniones por puntos y los acuerdos tomados.

- Generar y actualizar la plantilla del control de la militancia, que archiva por cada CB el listado de los militantes que posee, así como datos fundamentales de la misma: zona, esfera y sector de la economía, también se archivan datos como la fecha de ingreso a las filas de cada uno de los militantes, la fecha de arribo a la plantilla y la causa y la fecha de baja así como su causa.
- Generar el cómputo que es un documento donde se archivan los datos fundamentales de las actas obtenidas de todos los CB. En el mismo quedan registrados los datos correspondientes a: nombre del CB, fecha de reunión, número de militantes, cantidad de presentes, cantidad de ausentes injustificados, cantidad de ausentes justificados así como la causa, si se realizó algún tipo de análisis en el CB y la sanción propuesta por cada uno, si realizan la discusión de la Actividad fundamental, así como el tema tratado y la discusión de la Vía de instrucción Política, cantidad de opiniones de la última, cantidad de acuerdos y cantidad de puntos del orden del día desarrollados.
- Generar y actualizar el ID2, donde se archivan los datos fundamentales de los Comités de Base existentes, la cantidad de miembros, datos referentes a las altas, traslados y bajas. Este se encuentra estrechamente relacionado con la Plantilla del control de la militancia y se utiliza como referencia a la hora de especificar en el acta la cantidad de militantes que posee un CB.
- Permite a los militantes ver su expediente de militante que contiene además de los datos personales, las sanciones que ha recibido, los méritos que le han otorgado, así como distinciones u otros reconocimientos durante toda su trayectoria en la organización.
- Aplicar una sanción a un militante especificando, la fecha de aplicada la sanción, la causa de la sanción y el tipo de sanción.
- Otorgar un reconocimiento a un militante especificando la fecha de otorgamiento, el motivo y el reconocimiento.
- Generar reportes.

2.1.1- Especificación de los requisitos del sistema

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema

que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente (Redmine, 2015). Se definieron un total de 51 requisitos funcionales que se enumeran a continuación.

2.1.1.1- Requisitos funcionales del sistema

RF1 Gestionar los usuarios

RF1.1 Crear un usuario

RF1.2 Modificar los datos de un usuario

RF1.3 Eliminar un usuario

RF1.4 Buscar un usuario

RF2 Autenticar usuario

RF3 Gestionar una estructura organizativa

RF3.1 Crear una estructura organizativa

RF3.2 Modificar una estructura organizativa

RF3.3 Eliminar una estructura organizativa

RF3.4 Buscar una estructura organizativa

RF4 Gestionar militantes

RF4.1 Dar alta a militante

RF4.2 Modificar militante

RF4.3 Dar baja a militante

RF4.4 Buscar militante

RF5 Realizar un traslado interno

RF6 Gestionar una sanción a un militante.

RF6.1 Crear sanción a un militante

RF6.2 Modificar sanciones a un militante

RF6.3 Listar las sanciones de un militante

RF7. Gestionar plantilla del control de la militancia

RF7.1 Crear plantilla del control de la militancia

RF7.2 Buscar plantilla del control de la militancia

RF8. Gestionar acta

RF8.1 Crear acta

RF8.2 Buscar acta

RF8.3 Generar listado de acuerdos.

RF9 Gestionar lista de militantes ausentes

RF9.1 Adicionar militantes ausentes

RF9.2 Eliminar militantes ausentes

RF9.3 Generar listado de militantes ausentes

RF10 Gestionar acuerdo

RF10.1 Adicionar acuerdo

RF10.2 Modificar acuerdo

RF10.3 Buscar acuerdo

RF11 Gestionar orden del día

RF11.1 Adicionar orden del día

RF11.2 Modificar orden del día

RF12 Gestionar roles de usuarios

RF12.1 Adicionar roles de usuarios

RF12.2 Modificar roles de usuarios

RF12.3 Eliminar roles de usuarios

RF12.4 Buscar roles de usuarios

RF13 Gestionar cronograma

RF13.1 Adicionar un cronograma

Reglas del negocio a considerar

- Los nombres de los Comités de Base de un Comité Primario no deben ser iguales.
- Para que un Comité de Base pueda ser activado debe tener 5 militantes o más.
- Los movimientos hacia otra provincia se realizarían hacia uno de los municipios de las distintas provincias del país. Se consideran provincias: El municipio especial Isla de la Juventud, el Comité Especial UJC y la UCI.
- Para poder modificar datos el militante ya debe existir.
- Los movimientos internos son entre los Comités de Base dentro del Comité Primario. En esta categoría entran los traslados entre municipios, que serían entre C/B de diferentes Comités Primarios del Comité UJC de la UCI ya que en la universidad cada comité primario se considera un municipio.
- Los movimientos de municipio en un comité UJC, son entre los Comités de Base de un Comité Primario a otro o entre C/B de diferentes Municipios de la misma provincia.
- Los movimientos hacia otra provincia se realizarían hacia uno de los municipios de las distintas provincias del país. Se consideran provincias: El municipio especial Isla de la Juventud, el Comité Especial UJC y la UCI.

- Los movimientos hacia el exterior se realizarían entre cualquier C/B de la Universidad y los diferentes países hacia donde se dirigen esos militantes.
- El militante debe ser eliminado al cabo de los 5 años siempre que su último movimiento sea una baja guardando su nombre, número de carné de identidad, último Comité de Base y día de la baja.
- Para realizarse un crecimiento el joven debe estar entre los 16 y 32 años y debe formar parte del Universo Juvenil.

2.1.1.2- Requisitos no funcionales del sistema

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (SensioLabs, 2012)

Diseño e implementación

- RNDI 1 Se utilizará PostgreSQL en su versión 9.1 o superior como Sistema Gestor de Bases de Datos.
- RNDI 2. Se utilizará Apache 2.2 o superior como servidor Web de aplicaciones.
- RNDI 3 Se utilizará PHP 5.3 o superior como Lenguaje de Programación y Symfony2 como Framework Integrador.
- RNDI 4 Se utilizará como arquitectura del sistema Modelo – Vista – Controlador.
- RNDI 5 Se utilizará NetBeans en su versión 8 o superior como IDE para el desarrollo del sistema.
- RNDI 6 Se utilizará Ubuntu 10.4 ó 12.4 como Sistema Operativo para el desarrollo del sistema, así como para documentar y modelar el sistema.
- RNDI 7 Se utilizará Visual Paradigm como herramienta de modelado.
- RNDI 8 Las comunicaciones entre cliente y servidor se harán a través de la interfaz HTTP y HTTPS, utilizando redes LAN o WAN.
- RNDI 9 Todos los componentes del sistema deberán ser implementados siguiendo el patrón de Bajo Acoplamiento y Alta Cohesión.

- RNDI 10 Para el diseño de las páginas se utilizará el motor de plantillas Twig en su versión 1.6, complementado con CSS, CSS3, HTML, XHTML, HTML 5 y jQuery 1.7 o superior.

Requisitos Legales, de Derecho de Autor y otros.

- RNL 1 Los requisitos legales del Sistema para la Gestión de los Procesos de la UJC en la Facultad 3 quedan reservados para su uso por la Universidad de las Ciencias Informáticas.

Seguridad

- RNS 1 El acceso al sistema así como la información se encontrarán protegidos contra accesos no autorizados utilizando mecanismos de autenticación propios del sistema.
- RNS 2 La autenticación del sistema será la primera acción del usuario, el cual deberá proporcionar un usuario único y una contraseña, los cuales serán de uso exclusivo del propio usuario.
- RNS 3 Las diferentes áreas del sistema se encontrarán protegidas contra acceso no autorizado utilizando roles y grupos de usuarios.
- RNS 4 El administrador del sistema como política de seguridad podrá restringir el acceso a las diferentes áreas del sistema a los usuarios o grupos de usuarios.
- RNS 5 El sistema deberá estar disponible las 24 horas del día durante los 365 días del año, salvo sea necesario su mantenimiento o actualización.

Interfaz de usuario

- RNIU 1 Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la interpretación correcta de la información.
- RNIU 2 La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error y mostrando como título el detalle del error.
- RNIU 3 Todos los textos y mensajes en pantalla serán mostrados según el idioma seleccionado para el sistema.
- RNIU 4 El diseño de la interfaz del sistema responderá a la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.

2.2- Desarrollo del Sistema de Gestión de los Procesos de la UJC en la Facultad 3.

La metodología programación extrema tiene 4 fases fundamentales: Planificación, Diseño, Desarrollo y Prueba. En este capítulo se hace referencia a las fases de Planificación y Diseño.

2.2.1- Fase de Planificación.

XP plantea la planificación como un diálogo entre el desarrollador y el cliente los cuales llegarán a un consenso con respecto al alcance del proyecto, la prioridad a la hora de implementar las funcionalidades requeridas, la composición de las versiones y la fecha de entrega de las mismas. La metodología plantea una serie de liberaciones parciales al terminar cada iteración para retroalimentar el proceso de codificación y probar si se están cumpliendo los requisitos del cliente. Los principales artefactos de esta fase son las historias de usuario, que sirven de guía a todo el proceso de desarrollo. Son utilizadas para especificar los requisitos del software y las escriben los propios clientes según su percepción de las necesidades del sistema. Las historias de usuario proporcionan los detalles sobre la estimación del riesgo y tiempo que llevará la implementación de determinadas funcionalidades. Son una vista a muy alto nivel del costo y esfuerzo del proyecto que se desea desarrollar.

2.2.1.1- Historias de Usuarios.

Para la implementación del sistema se realiza la redacción de las historias de usuarios según las necesidades del cliente. Las historias de usuarios a implementar son:

- Gestionar usuario
- Autenticar usuario
- Gestionar una estructura organizativa
- Gestionar militante
- Gestionar una sanción a un militante
- Realizar un traslado interno
- Gestionar plantilla del control de la militancia
- Gestionar acta
- Gestionar listado de militantes ausentes
- Gestionar acuerdo
- Gestionar orden del día
- Gestionar roles de usuarios
- Gestionar cronograma
- Gestionar mérito
- Generar reportes

Historia de Usuario	
Número: 1	**Nombre:** Gestionar usuario
Prioridad en Negocio(Alta/Media/Baja): Alta	**Dependencia: -**
Desarrollador Encargado: Rolando Quintana Safont	
Descripción: Debe permitir crear un usuario con su nombre, contraseña, si está activo o no y el rol que tendrá. Los roles serán documentador, administrador, organizador, secretario, activista de acta y militante. Permitirá además modificar los datos de un usuario y eliminar un usuario.	
Observaciones:	

El resto de las historias de usuario se podrá encontrar en los anexos.

2.2.1.2- Planificación de las Historias de Usuario.

Para planificar acertadamente el desarrollo de la aplicación, es necesario estimar de forma ideal el tiempo que les tomará a los programadores la codificación de cada una de las historias de usuario. Las condiciones ideales se dan cuando se escribe el código sin distracciones y con una dedicación de tiempo completo. La estimación debe dar un máximo de 2 semanas, si es necesario más tiempo debe considerarse una división de la historia de usuario en partes más pequeñas. A partir de la prioridad de las historias se decide cuáles de ellas se implementarán en las primeras iteraciones, las funcionalidades críticas del sistema deben ser codificadas en iteraciones tempranas del ciclo.

No.	Nombre de la HU	Prioridad	Esfuerzo (Días)	Iteración
1	Gestionar usuario	Alta	8	1
2	Autenticar usuario	Alta	4	1
3	Gestionar una estructura organizativa	Media	7	2
4	Gestionar militante	Alta	9	1
5	Realizar un traslado interno	Alta	2	1

6	Gestionar una sanción a un militante.	Baja	7	3
7	Gestionar plantilla del control de la militancia	Alta	7	1
8	Gestionar Acta	Media	14	2
9	Gestionar lista de militantes ausentes	Media	4	2
10	Gestionar acuerdo	Media	4	2
11	Gestionar orden del día	Media	4	2
12	Gestionar roles de usuarios	Alta	6	1
13	Gestionar cronograma	Media	5	2
14	Gestionar méritos	Baja	5	3
15	Generar reportes	Baja	8	3

Tabla 2.1: Estimación del esfuerzo y prioridad por HU

2.2.1.3- Plan de Entrega.

Una se vez identificadas las HU del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas, se procede a la planificación de la etapa de implementación del proyecto. En base a lo antes mencionado se decide realizar esta etapa en tres iteraciones, las cuales se detallan a continuación especificando algunas de las tareas de ingeniería a realizar por cada HU que se implemente. El resto de las mismas se pueden encontrar en los anexos.

Iteración 1

Esta iteración tiene como objetivo la implementación de las HU de mayor prioridad. Durante el transcurso de la misma se creará la base de la arquitectura del sistema con una funcionalidad mínima haciendo mayor énfasis en la gestión de roles y usuarios del sistema, así como en la gestión de militantes, de las plantillas y de la configuración del sistema. Al final se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Tarea de Ingeniería	
Número de la Tarea: 1	**Número de la HU:** 1
Nombre de la Tarea: Definición y desarrollo de la interfaz crear usuario.	
Tipo de la Tarea: Desarrollo	
Programador Responsable: Rolando Quintana Safont	
Descripción: Se realiza el diseño de la pantalla crear usuario. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Iteración 2

El objetivo de ésta es la implementación de las HU de prioridad media. Al finalizar se contará con una versión de prueba con las funcionalidades concernientes a la gestión de las OB, la gestión de las actas y el listado de ausentes, de acuerdos, el orden del día y el cronograma, además de las funcionalidades implementadas en la iteración anterior. Esta será mostrada al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Tarea de Ingeniería	
Número de la Tarea: 1	**Número de la HU: 3**
Nombre de la Tarea: Definición y desarrollo de la interfaz crear un CB, comités primarios y comités de base independiente.	
Tipo de la Tarea: Desarrollo	
Programador Responsable: Thais Sánchez Martínez	
Descripción: Se realiza el diseño de la pantalla crear un CB, comités primarios y comités de base independiente. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Iteración 3

Durante el transcurso de ésta, se implementaron las HU de baja prioridad. Al finalizar la misma se constará con la versión 1.0 del producto final, adicionando todo lo concerniente al manejo de estadísticas y visualización de los reportes a mostrar, la gestión de sanciones y de méritos, además de las funcionalidades anteriores.

Tarea de Ingeniería	
Número de la Tarea: 1	**Número de la HU: 4**
Nombre de la Tarea: Definición y desarrollo de la interfaz modificar los datos generales del expediente del militante.	
Tipo de la Tarea: Desarrollo	
Programador Responsable: Rolando Quintana Safont	
Descripción: Se realiza el diseño de la pantalla modificar los datos generales del expediente del militante. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Módulo	Fin 1ra Iteración 1ra semana de Abril	Fin 2da Iteración 1ra semana de Mayo	Fin 3ra Iteración 4ta semana de Mayo
Militante	x		
ID2	x		
Acta		x	
Sanción y Mérito			x
SIGPU 1.0			x

Tabla 2.2 - Plan de Entregas

2.2.2- Diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Por lo que se procura hacerlo todo lo menos complicado posible para conseguir un diseño fácil de implementar y entender que a la larga costará menos tiempo y esfuerzo al desarrollar.

2.2.2.1- Arquitectura del Sistema

Durante el ciclo de vida de desarrollo de un software utilizando la metodología XP la fase de diseño se realiza justo después que se termina la fase de planificación. Durante esta fase se comienza a definir el diseño arquitectónico del software partiendo de las historias de usuario de la primera iteración y de la metáfora, un elemento que según Kent Beck el creador de la metodología, guía al equipo de proyecto durante todo el desarrollo del programa y que además sustituye en gran parte de lo que sería la arquitectura del sistema en modelos más pesados o tradicionales.

La arquitectura de la herramienta o de cualquier proyecto guiado por XP debe ser sencilla y responder al principio de “cuál es la solución más simple capaz de funcionar”. Debe ser una estructura que organiza la lógica del sistema de manera que al realizar cambios en partes no provoca cambios en otros componentes, a menos que sea absolutamente necesario. Debe enfocarse en cómo resolver los problemas identificados en el momento en que se diseña, sin tener en cuenta posibles variaciones o requerimientos futuros, los cuales serán enfrentados y agregados al diseño en el momento en que sean detectados.

2.2.2.1.1- Metáfora del sistema propuesto

La metáfora es una historia simple de cómo funciona todo el sistema compartida por todo el equipo de desarrollo. Su misión es describir el software de manera que los desarrolladores y clientes que trabajan en conjunto puedan comunicarse fácilmente a la hora de referirse al sistema, o una parte de éste. Cuando el sistema es pequeño la metáfora es extremadamente sencilla (Beck, 1999). Teniendo en cuenta los parámetros anteriormente expuestos, la metáfora del sistema sería: Sistema para gestionar los procesos de la UJC en la Facultad 3.

2.2.2.1.2- Arquitectura

Para definir correctamente la arquitectura de un software es necesario que se identifiquen los elementos arquitectónicamente significativos (clases, módulos, objeto, interfaces) y las relaciones que existen entre ellos. Para garantizar que los cambios o nuevas funcionalidades añadidas puedan ser

asimilados por la arquitectura del sistema, esta estará formada por capas (presentación, lógica del negocio y acceso a datos) organizadas por responsabilidades.

Imagen 2.1 - Arquitectura Modelo - Vista – Controlador

2.2.2.2 - Patrones del diseño

Los patrones son soluciones a problemas existentes y que pueden ser aplicados en varios contextos. Los patrones en sí, permiten codificar conocimientos, estilos y principios existentes y han sido validados. Las características que los distinguen y que los hacen tan importantes, es que ayudan a un diseño común entre diferentes desarrolladores, favorecen a la creación de sistemas sin depender del lenguaje a utilizar y pueden llegar a acortar la fase de diseño en un proceso de desarrollo de software.

- El patrón Creador: Este patrón permite darle solución al problema de quién es el responsable de crear alguna nueva instancia de alguna clase. Un ejemplo que muestra la aplicación de este patrón es cuando en las clases actions se crean instancias de las clases modelo. El “action” tiene el Id de una persona y desea ver los datos de la misma, para esto necesita entonces, crear una instancia de esa persona.

```php
public function adicionarModificarMilitanteAction($idMilitante) {
    $request = $this->getRequest();

    if ($request->getMethod() == 'POST') {
        $municipioNacimiento = $request->get('municipioNacimiento');
        $nombrePadre = $request->get('nombrePadre');
        $nombreApellidos = $request->get('nombreApellido');
        $nombreMadre = $request->get('nombreMadre');
        $carneMilitante = $request->get('carneMilitante');
        $carneIdentidad = $request->get('carneIdentidad');
        $fechaIngresofilas = $request->get('fechaIngresoFilas');
        $fechaIngresofilas = FechaUtil::StringToDate($fechaIngresofilas);
        $centroCrecimiento = $request->get('centroCrecimiento');
        $laborRealiza = $request->get('laborRealiza');
        $ocupacion = $request->get('ocupacion');
        $sectorEconomia = $request->get('sectorEconomia');
        $viaIngreso = $request->get('viaIngreso');
        $comiteBase = $request->get('comiteBase');
        $causaAlta = $request->get('causaAlta');
        if ($idMilitante != -1) {
            $militante = $this->getGestor()->ObtenerMilitanteXId($idMilitante);
        } else {
            $militante = new TbMilitante();
            $militante->setActivo(TRUE);
            $militante->setCiudadania('Cubana');
            $comiteB = $this->getGestor()->ObtenerComiteBaseXId($comiteBase);
            $militantecomite = new TbMilitanteComiteBase();
            $militantecomite->setComiteBase($comiteB);
            $fecha = new \DateTime();
            $militantecomite->setFechaInicio($fecha);
```

Imagen 2.2- Código del método adicionarModificarMilitanteAction ()

- El patrón Controlador: El controlador se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. En Symfony todas las peticiones al sistema son manejadas por un controlador que se divide en varios componentes encargados de la seguridad, validaciones, configuración y enrutamiento, de igual modo se definen las acciones a ejecutar por el sistema en las clases actions.

```php
class MilitanteController extends BaseController {

    public function indexAction() {
        $modelo = $this->get('gestionarMilitantesTM');
        $ruta = new RutasGrid();
        $ruta->setRutaDatosAjax('listar_militantes');

        return $this->render('PruebaBundle:Militante:militante.html.twig', array(
                    'modelo' => $modelo,
                    'ruta' => $ruta,
        ));
    }

    public function listarAction() {
        $militantes = $this->getGestor()->ObtenerTodosLosMilitantes();
        return $this->Grid('gestionarMilitantesTM', $militantes);
    }
```

Imagen 2.3- Código de la clase MilitanteController

- El patrón Decorador: responde a la necesidad de añadirle dinámicamente funcionalidades aun objeto determinado. En Symfony el archivo layout.php contiene al layout de la página (plantilla global), esta almacena el código html que es común a todas las páginas de la aplicación para no tener que repetirlo en cada una. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado “Decorator”.

```twig
<!DOCTYPE html>
<html>
    <!DOCTYPE html>
    <!--[if IE 8]> <html lang="en" class="ie8"> <![endif]-->
    <!--[if IE 9]> <html lang="en" class="ie9"> <![endif]-->
    <!--[if !IE]><!--> <html lang="en"> <!--<![endif]-->
        <head>
            <meta charset="UTF-8" />
            <title>
                {% block title %}Sistema para la Gestión de los Procesos de la UJC en la
                {% endblock %}
            </title>
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <meta content="width=device-width, initial-scale=1.0" name="viewport" />
            <meta content="" name="description" />
            <meta content="" name="author" />

            {% block JS_meta %}
```

Imagen 2.4- Código de la plantilla base del Sistema

- Los patrones Bajo Acoplamiento y Alta Cohesión están presentes en todo el sistema y son utilizados para permitir la asignación de responsabilidades entre clases estableciendo siempre una colaboración entre ellas, permitiendo dar paso a la reutilización. Para saber el porciento en el que se encuentra cada uno en el sistema se aplicó la métrica Relación entre clases (RC).

Relación entre clases (RC): La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman, 2002).

Para determinar el grado de afectación para los atributos de calidad que mide la métrica RC es necesario conocer la cantidad de relaciones de uso (CRU) que poseen las clases a medir. Una vez obtenida la CRU, se procede a calcular el promedio de las mismas. Teniendo en cuenta ambos valores, según los criterios expuestos en la tabla que se muestra a continuación, se determina la incidencia de los atributos de calidad en cada una de las clases. (Lorenz & Kidd, 1995)

Atributo	Categoría	Criterio
Responsabilidad	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
Complejidad implementación	Baja	CRU <= PO
	Media	PO < CRU < = 2* PO
	Alta	CRU > 2* PO
Reutilización	Baja	CRU > 2* PO
	Media	PO < CRU < = 2* PO
	Alta	CRU <= PO
Cantidad de Pruebas	Baja	CRU <= PO
	Media	PO < CRU < = 2* PO
	Alta	CRU > 2* PO

Tabla 2.7-. Atributos de calidad y el modo en que son afectados.

Como resultados de su aplicación se obtuvieron los siguientes resultados:

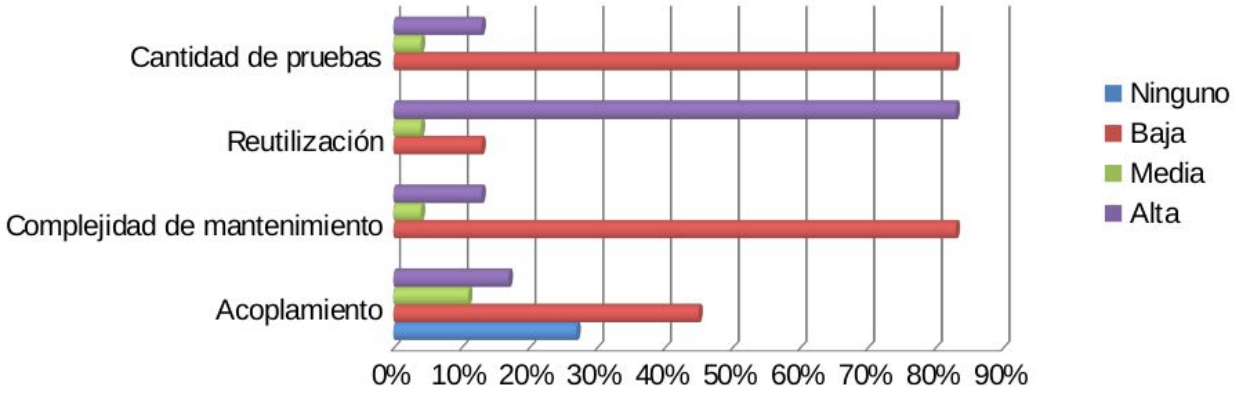

Imagen 2.5- Resultados de la métrica RC

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Luego de aplicarse la métrica de diseño RC y obtenidos los resultados, se puede concluir que el sistema presenta un bajo acoplamiento, complejidad de mantenimiento, cantidad de pruebas y alta reutilización todos con un 83 % de positividad.

- Para el trabajo con la base de datos, Doctrine 2 utiliza diferentes patrones como el Unit of Work que es un objeto encargado de realizar transacciones utilizado por el EntityManager (Administrador de Entidades de Doctrine 2) para llevar el conteo de las consultas que tiene que ejecutar una vez se invoque el método flush () del EntityManager, o sea, mantiene una lista de los objetos afectados por una transacción de negocios y coordina los cambios de redacción y la resolución de problemas de concurrencia.

```
/**
 * The EntityManager is the central access point to ORM functionality.
 */
/* final */class EntityManager implements EntityManagerInterface
{
    /**
     * The UnitOfWork used to coordinate object-level transactions.
     *
     * @var \Doctrine\ORM\UnitOfWork
     */
    private $unitOfWork;
```

Imagen 2.6- Código de la declaración del objeto unitOfWork en la clase EntityManager

```php
/**
 * Flushes all changes to objects that have been queued up to now to the database.
 * This effectively synchronizes the in-memory state of managed objects with the
 * database.
 *
 * If an entity is explicitly passed to this method only this entity and
 * the cascade-persist semantics + scheduled inserts/removals are synchronized.
 *
 * @param null|object|array $entity
 *
 * @return void
 *
 * @throws \Doctrine\ORM\OptimisticLockException If a version check on an entity that
 *         makes use of optimistic locking fails.
 */
public function flush($entity = null)
{
    $this->errorIfClosed();

    $this->unitOfWork->commit($entity);
}
```

Imagen 2.7- Código del Método flush ()

- Symfony 2 utiliza el patrón Inyección de Dependencias para estandarizar y centralizar la forma en laque son construidos los objetos. El contenedor de inyección de dependencias de Symfony2 es el encargado de cargar todos los objetos e inyectarlos (inyección) posteriormente cuando sea requerido las clases que los necesiten (dependencia).

```php
class Container implements IntrospectableContainerInterface
{
    /**
     * @var ParameterBagInterface
     */
    protected $parameterBag;

    protected $services = array();
    protected $methodMap = array();
    protected $aliases = array();
    protected $scopes = array();
    protected $scopeChildren = array();
    protected $scopedServices = array();
    protected $scopeStacks = array();
    protected $loading = array();
```

Imagen 2.7- Código del Contenedor de Dependencias

2.2.2.3- Tarjetas CRC

Las tarjetas CRC son técnicas de modelado creadas para ayudar a los desarrolladores de software a crear diseños de clases orientados a responsabilidades. Dichas tarjetas constan de tres secciones, nombre de la clase, responsabilidades y colaboradores. La sección responsabilidades lista cada una de las funciones o tareas que debe ser capaz de cumplir un objeto de dicha clase mientras que la sección colaboradores contiene otras clases del diseño que pueden colaborar para proveer datos o funcionalidades.

La metodología XP estipula en toda la bibliografía consultada, el uso de las tarjetas CRC como un artefacto obligatorio durante el desarrollo de un proyecto debido a los beneficios que aporta en cuanto a la comunicación y transparencia del desarrollo. La mayoría de los autores recomiendan que se encuentren en un lugar público, a la vista de todo el equipo, donde todos los programadores puedan contribuir a mejorar y refinar el diseño resultante; así como verificar que diferentes clases no dupliquen responsabilidades o que una interfiera con las funcionalidades de la otra; problemas muy frecuentes a la hora de desarrollar una arquitectura bajo paradigmas de orientación a objetos.

Clase Militante	
Responsabilidad	**Colaboradores**
adicionarModificarMilitante gestionarCargos darBaja cargarComboOcupacion	

Tabla 2.3 – Tarjeta CRC de la HU Gestionar militante

2.2.2.4- Diseño de la Base de Datos

Una de las tareas más importantes a la hora de construir una aplicación Web es la base de datos ya que uno de sus objetivos fundamentales es brindar la persistencia al modelo que se describe. El modelo de datos del problema en cuestión posee un nivel de complejidad alto, producto a que la aplicación es compleja.

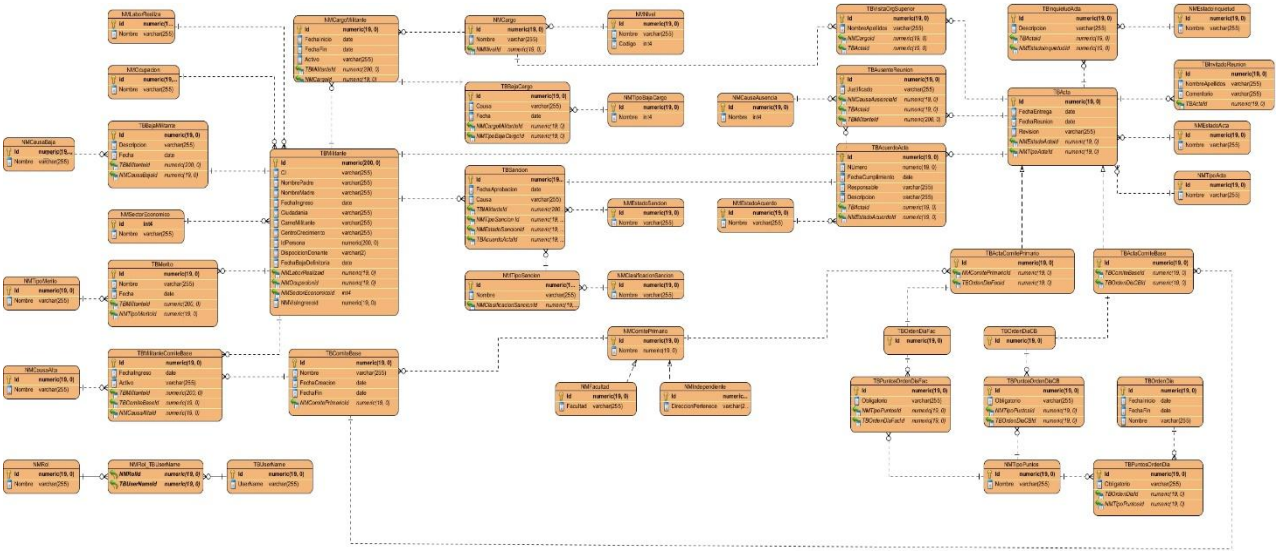

Imagen 2.8 – Diagrama entidad-relación

2.3- Conclusiones Parciales

En el presente capítulo se realizó una descripción de la solución general propuesta, se definieron y redactaron las historias de usuario así como las tareas de ingeniería correspondientes a cada una de ellas dando como resultado una estimación ideal de 100 días necesarios para implementar la herramienta. Además se generaron los diagramas UML considerados indispensables para asegurar la completa comprensión y claridad del sistema antes de comenzar a codificar las historias obtenidas y las tarjetas CRC necesarias para la descripción de las clases del sistema, así como el modelo de datos y la arquitectura del mismo.

Capítulo 3: Codificación y Pruebas

Introducción

En el presente capítulo se definirán los artefactos que define XP para las fases de implementación y prueba: casos de pruebas necesarios para poder ejecutar las pruebas al software y así validar la calidad del mismo, asegurándose el cumplimiento de los requisitos. Además se incluyen otros temas como el tratamiento de errores y la seguridad.

3.1- Estándares de Codificación

El estándar de codificación seleccionado para el desarrollo del sistema se define a continuación:

- Identación.

El contenido siempre se identará con tabs, nunca utilizando espacios en blanco.

- Comentarios en las funciones.

Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben ser suficientes para entender el código.

- Ubicación y denominación de archivos.

Se ubicarán los archivos según las convenciones establecidas por Symfony2. Para la denominación de los archivos se seguirán las convenciones establecidas por Symfony2. Ejemplo:

1. Para las clases de gestión del negocio se usará el sufijo Gtr: AcusadoGtr.
2. Para los table model definidos para los grid se usará el sufijo Tm: AcusadoTm.
3. Para las entidades de presentación se usará el sufijo Ep: AcusadoEp.
4. En las páginas, plantillas html.twig definidas para la vista el nombre del archivo debe seguir el estándar de la denominación de las clases. Ejemplo: Acusados.html.twig. En caso de estar formado por más de una palabra: AcusadosRebeldes.html.twig.
5. Para la denominación de los servicios a crear se tendrán en cuenta los siguientes elementos:
    - Para los parámetros se utilizará la siguiente nomenclatura (todo en minúscula utilizando como separador el carácter punto (.)): bundle.entidad.categoria.class:

- Para los servicios se utilizará la siguiente nomenclatura (todo en minúscula utilizando como separador el carácter punto (.)): bundle.entidad.categoria:
- Para las rutas se utilizará la siguiente nomenclatura (todo en minúscula utilizando como separador el carácter guión bajo (_)): bundle_ruta:

- Clases

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo del de la clase. Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad. Los nombres de las clases deben iniciar con letra mayúscula. Si un nombre de clase se comprende de más de una palabra, la primera letra de cada nueva palabra debe comenzar con letra mayúscula. No se permiten las letras mayúsculas sucesivas; por ejemplo, una clase "SymfonyPDF" no se permite, mientras " SymfonyPdf" es aceptable. Siempre utilizar las etiquetas <?php ?> para abrir un bloque de código. No utilizar el método de etiquetas cortas.

- Nombres de variables.

Los nombres deben ser descriptivos y concisos. No usar grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber qué hace una variable con solo conocer su nombre. Esto se aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

- Las definiciones de la función

Los nombres de las funciones pueden contener solo caracteres alfanuméricos y siempre deben empezar en letras minúsculas. Cuando el nombre de una función conste de más de una palabra, la primera letra de cada nueva palabra debe comenzar con mayúscula. Ejemplo: listarUsuariosAction ()

- Llamadas a funciones

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura y el primer parámetro; los espacios entre las comas y cada parámetro y ningún especial entre el último parámetro, el paréntesis del cierre y el punto y coma.

- Siempre incluir las llaves.

En todo momento a la hora de codificar un bloque de instrucciones, este debe ir encerrado entre llaves, aun cuando conste de una sola línea.

- Llaves. Dónde colocarlas.

Las llaves de apertura irán al final de la sentencia que delimitan y las de cierre, alineadas con el inicio de la sentencia en una nueva línea.

- Poner espacios entre signos.

Si se tiene un signo y operador binario, se colocan espacios a ambos lados. Si se tiene un signo unario, se colocan espacios a uno de sus lados. En términos más simples, se programa como si se escribiera bien en español. Este elemento es algo muy sencillo que ayuda a la legibilidad del código.

- Precedencia de operadores

Lo mejor es siempre usar paréntesis para estar seguro de la precedencia de los operadores

- No utilizar variables sin inicializar.

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada. Pero sólo se debe usar está opción cuando no se tenga el control o no se esté seguro del valor que pueda tener la variable (Como en variables que llegan por POST o por GET, etc.).

- Instrucción “switch”.

Cuando se utilice, se deberá seguir el siguiente estilo:

```
switch ($modo) {
        case 'modo1':
                // Instrucción 1
                break;
        default:
                // Código a ejecutar si todo falla
                break;
}
```

3.1.1- Codificación

Al principio de las iteraciones se lleva a cabo una revisión al plan de iteraciones y se modifica en caso de ser necesario. En el transcurso de las mismas, se realiza la implementación de las historias de usuario que fueron seleccionadas en cada una de ellas y se descomponen estas historias en tareas de ingeniería. Teniendo en cuenta la planificación realizada en el capítulo anterior, se llevaron a cabo tres iteraciones de desarrollo sobre el sistema, obteniéndose un producto con funcionalidad en cada una de ellas. A continuación se detallan cada una de las iteraciones.

Iteración 1

En esta iteración se implementaron las historias de usuario con una alta prioridad para el usuario, con el objetivo de obtener una versión del producto con algunas de las funcionalidades más importantes para ser mostrado al cliente.

Historias de Usuario	Tiempo de Implementación (Días)	
	Estimación	Tiempo Real
Gestionar usuario	8	5
Autenticar usuario	4	6
Gestionar militante	9	7
Gestionar traslado interno	2	4
Gestionar plantilla de control de la militancia	7	5
Gestionar roles	6	4

Tabla 3.1 - Estimación de la iteración 1

Iteración 2

En esta iteración se implementaron las historias de usuario con una prioridad media para el usuario.

Historias de Usuario	Tiempo de Implementación (Días)	
	Estimación	Tiempo Real
Gestionar estructura organizativa	7	4

Gestionar acta	14	15
Gestionar listado de militantes ausentes	4	3
Gestionar acuerdo	4	3
Gestionar orden del día	4	3
Gestionar cronograma	5	4

Tabla 3.2 - Estimación de la iteración 2

Iteración 3

En esta iteración se implementó la historia de usuario de prioridad baja para el cliente. Esta tiene la finalidad de proporcionarle un ambiente afable y cómodo para el usuario. Al finalizar se cuenta con un producto listo para poner en funcionamiento.

Historias de Usuario	Tiempo de Implementación (Días)	
	Estimación	Tiempo Real
Gestionar sanción	7	5
Gestionar méritos	5	3
Generar reportes	8	12

Tabla 3.3 - Estimación de la iteración 3

3.1.2- Diagrama de despliegue

Los diagramas de despliegue muestran nodos, conexiones, componentes y objetos. Los nodos representan objetos físicos con recursos computacionales como procesadores y periféricos; pueden mostrarse como una clase o una instancia, por lo que su nombre sigue la misma sintaxis establecida para clases y objetos. Las conexiones son asociaciones de comunicación entre los nodos y se etiquetan con un estereotipo que identifica el protocolo de comunicación o la red utilizada. Los componentes son archivos de código ejecutable, que residen y se ejecutan dentro de un nodo; se pueden representar relaciones de dependencia entre los componentes que, de manera similar a las dependencias entre paquetes, corresponden al uso de servicios. En este caso la aplicación se encuentra hospedada en un servidor Web y esta se comunica con un sistema de gestión de base de datos.

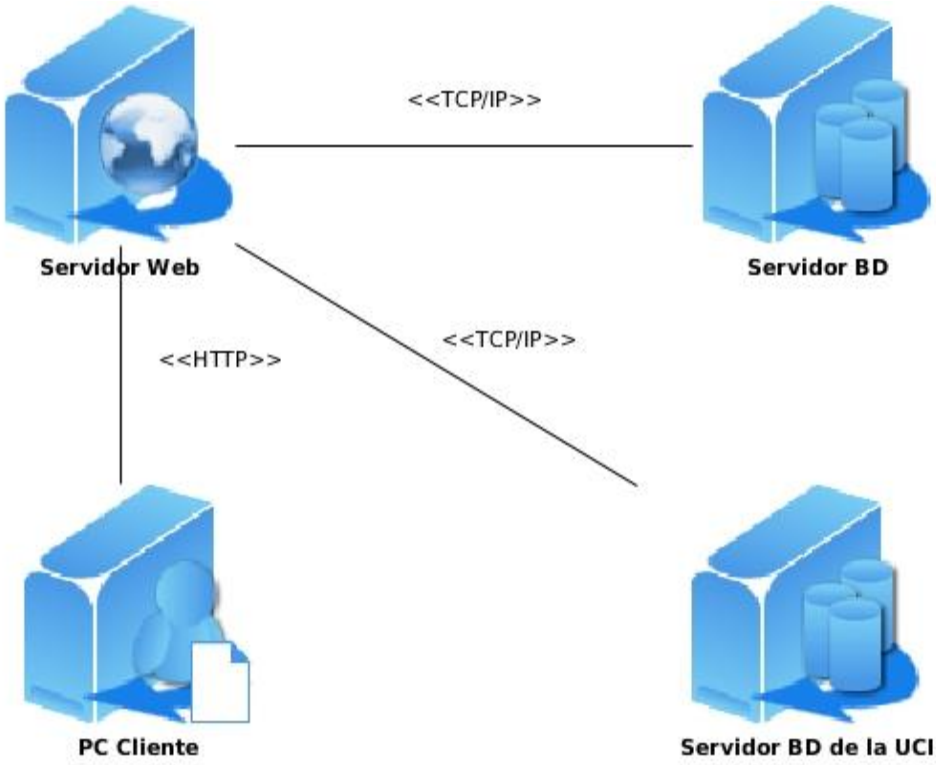

Imagen 3.1 - Diagrama de despliegue

3.2- Pruebas

La verificación y validación es el nombre dado a los procesos de análisis y prueba que tienen lugar en cada etapa del proceso de software, que comienza con las revisiones de los requisitos y continúa con revisiones del diseño e inspecciones de código, hasta las pruebas del producto. Estas validaciones en un software son de gran importancia ya que con estas se retroalimenta al usuario de datos importantes y se evitan errores. En el presente trabajo se utilizaran las pruebas de caja negra y caja blanca para llevar a cabo la validación de la solución.

3.2.1- Prueba de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta y que la integridad de la información externa se mantiene. De las diferentes técnicas que existen para desarrollar estas pruebas, se empleara la 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.

Los datos de los casos de prueba correspondientes a cada una de las historias de usuario desarrollados se presentan como uno de los artefactos generados de la presente investigación. Las pruebas fueron realizadas por el equipo de calidad del centro y durante la ejecución de las mismas se realizaron tres iteraciones. Entre los errores detectados se encuentran principalmente errores ortográficos, de validación de interfaces, de correspondencia con la documentación, de funcionalidad y de carga de datos, los cuales fueron corregidos y en una tercera iteración no se detectó ninguno. A continuación se presentan los resultados obtenidos al aplicar esta prueba al producto obtenido.

Iteración	No Conformidades
1	30
2	9
3	0

Tabla 3.6 - Cantidad de no conformidades por iteraciones

3.2.2- Prueba de caja blanca

Permiten inspeccionar la estructura interna del programa. Se diseñan casos de prueba para examinar la lógica del programa. En este caso se utilizará la prueba del camino básico la cual es una técnica de prueba de la caja blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

Para la aplicación de esta técnica se siguieron los siguientes pasos:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Se tomó como ejemplo el método crearActaAction (), perteneciente a la clase actaController como base para realizar el procedimiento anteriormente descrito. La selección del método se realizó teniendo en cuenta la complejidad del mismo, el cual da

lugar a una de las principales funcionalidades que responden al objetivo general de la investigación. A continuación se muestra parte del código del método.

```
public function crearActaAction() {
    $request = $this->getRequest();
    //        cambiar cuando se haga logueo
    $comiteBase = UtilRepository2::getCBUsuarioLogged(true);
    $ds = array();
    $acta = new TbActa();
    $presentes = $request->get('presentes');
    if (count($presentes) > 0) {
        $arregloPresentes = array();
        $contp = 0;
        foreach ($presentes as $p) {
            $arreglo = array();
            //obtener el id del militante ausente y adicionarlo al acta
            $presenteReunion = $this->getGestor()->ObtenerMilitanteXId($p[0]);
            $contp++;
            $arreglo[0] = $contp;
            $arreglo[1] = $presenteReunion->getNombreApellidos();

            $cargos = $presenteReunion->getCargoMilitante();
            foreach ($cargos as $cargo) {
                if ($cargo->getActivo()) {
                    $nombreCargo = $cargo->getCargo()->getNombre();
                    break;
                }
            }
            $arreglo[2] = $nombreCargo;
            $arregloPresentes[] = $arreglo;
        }
```

Imagen 3.2 - Parte del código del método crearActaAction ()

Para aplicar la técnica del camino básico se realizaron una serie de pasos que a continuación se describen:

1. Confeccionar el grafo de flujo: usando el código de la Imagen 3.2 se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.

- Regiones: son las áreas delimitadas por aristas y nodos.

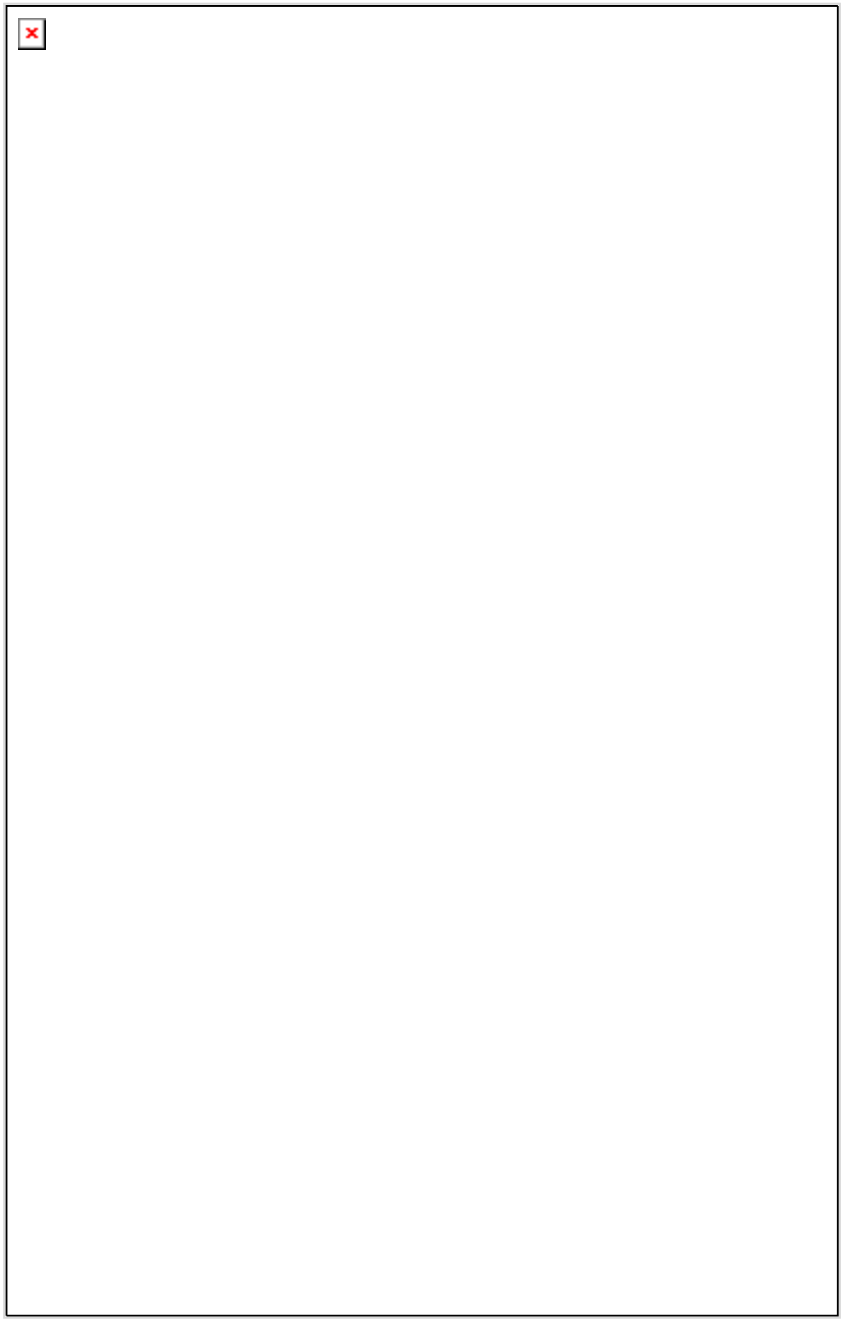

Imagen 3.3- Grafo del método crearActa ()

2. Calcular la complejidad ciclomática: proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa y da un límite inferior para el número de casos de prueba que se deben

realizar para asegurar que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática se calcula de tres formas:

- El número de regiones del grafo de flujo. V (G) = 18
- V (G)= A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo. V (G) = A-N+2 = 71 – 55 + 2 = 18
- V (G)= P+ 1, donde P es el número de nodos predicados. V (G) = P+1 = 17 + 1 = 18

3. Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) da el número de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 18 caminos:

Camino básico #1: 1-2-3-4-5-6-7-8-9-4

Camino básico #2: 1-2-3-4-5-6-8-9-4

Camino básico #3: 1-2-3-4-10-11-12-13-14-15-14

Camino básico #4: 1-2-3-4-10-11-12-13-14-16-17-18-19-20-21-20

Camino básico #5: 1-2-3-4-10-11-12-13-14-16-17-18-19-20-22-23-24-25-26-27-26

Y así sucesivamente hasta el 18.

4. Obtención de los casos de prueba: cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 18 caminos básicos, por tanto se hace necesario la confección de igual número de casos de prueba, para aplicar las pruebas a este método. A continuación se brinda un ejemplo de uno de ellos:

Caso de prueba: Camino básico #1	
Entrada	El acta debe tener al menos un militante presente y a cada uno de ellos se les añade a la tabla de presentes con sus respectivos nombres y apellidos y el cargo que tiene en el CB.
Resultados esperados	Se listan todos los militantes presentes del CB con todos los datos correctos.

Condiciones	count($presentes) > 0 $cargo->getActivo()

Tabla 3.7- Caso de prueba de caja blanca para el camino básico #1.

Una vez aplicadas las pruebas de caja blanca mediante la técnica del camino básico al método crearActa (), se detectó en una primera iteración varios errores en el código. Dichos errores fueron erradicados y se procedió a aplicar una segunda iteración de estas pruebas, de las cuales se obtuvieron resultados satisfactorios. De igual forma se le realizaron estas pruebas al resto de las funcionalidades de la herramienta, en un total de tres iteraciones.

3.3- Tratamiento de errores

Las excepciones son errores que se producen en tiempo de ejecución de una aplicación y detienen el flujo normal de esta. Para evitar este tipo de errores es recomendable capturarlas y tratarlas, para esto se han utilizado la excepción nativa de PHP “Exception” para el manejo de errores inesperados, el try Catch para la captura de las mismas y la excepción que brinda Symfony2 “NotFoundException” para el manejo de elementos que nos son encontrados al realizar búsquedas.

3.4- Seguridad

Para garantizar la seguridad de la información que se maneja en la aplicación, el control de acceso se encuentra garantizado mediante un nivel de accesos de usuarios y contraseñas, realizando solo los accesos que el rol tenga establecido. Para ello, se agregó al sistema el FOSUserBundle, que es un bundle de Symfony que más que proporcionar seguridad, añade varias características comunes necesarias para aprovechar el sistema de seguridad de Symfony tales como la inclusión de un formulario de acceso, un formulario de inscripción, la funcionalidad de olvido de la contraseña, la integración con la base de datos y mucho más. Permite además el uso de un subsistema de administración de usuarios, añadiendo acciones como registrar nuevos usuarios, enviar un mensaje de confirmación a los usuarios recién creados, editar el perfil de usuario, encriptación de contraseñas y agrega un campo en la base de datos para poder guardar roles.

3.5- Validación de la investigación

Para la validación de la investigación se realizó una prueba piloto al sistema con el objetivo de evaluar las variables de la misma: calidad de la información y la toma de decisiones y luego comparar

los resultados con los valores obtenidos antes de la puesta en marcha del sistema. Como resultado se obtuvieron los siguientes datos:

Antes	Después
El tiempo promedio de recibo de toda la información (actas) es de 20 días	El tiempo promedio de recibo de toda la información (actas) es de 10 días
La probabilidad de errores en cuanto al formato de la información es Alta	La probabilidad de errores en cuanto al formato de la información es Baja
El tiempo promedio de confección de un acta es de 2 horas	El tiempo promedio de confección de un acta es de 30 minutos
El tiempo promedio de análisis y consolidación de la información (reportes) es de 3 días	El tiempo promedio de análisis y consolidación de la información (reportes) es de 15 segundos

Con el sistema se logró garantizar además:

- La veracidad, unicidad y flexibilidad de los datos así como su integridad al ser validados desde el proceso de captura.
- La seguridad de la información impidiendo su manipulación sin que quede registro de esta modificación para luego, en caso que sea necesario, se pueda recurrir a la información base.
- La existencia de información adicional que permita identificar de cuándo es la información almacenada, a qué períodos corresponde para poderla usar en análisis de tendencias y comparativa. El número por sí solo no sirve mucho si no se sabe a qué corresponde y de cuándo es.
- El establecimiento de seguridades a diferentes niveles y la auditabilidad de las actividades, específicamente identificando quién hizo qué, cuándo y desde dónde.
- Mayor velocidad de respuesta en la consulta de datos.
- Una alta facilidad de uso del sistema para el usuario.
- Mayor volumen de información disponible para la toma de decisiones.

3.6- Conclusiones

El empleo del estándar de codificación seleccionado contribuyó al entendimiento y estandarización del código de la herramienta obtenida, facilitando la futura extensión y mantenimiento de la misma y la realización de pruebas de caja blanca y caja negra a la herramienta obtenida permitió comprobar el correcto funcionamiento de la misma, evaluar las clases diseñadas y descubrir errores una vez implementadas.

Conclusiones

Con el desarrollo del Sistema para la Gestión de la Información de la UJC en la Facultad 3 se dio respuesta al problema a resolver planteado, obteniendo las siguientes conclusiones:

- Se realizó un estudio detallado que permitió tener un conocimiento de la situación actual y las tendencias de los sistemas de gestión de la información en las organizaciones políticas, demostrando la necesidad de desarrollar un sistema que fuese capaz de automatizar los procesos que se llevan a cabo en la UJC.
- Al realizar las entrevistas con los clientes se obtuvieron datos relevantes sobre los procesos de negocio así como los requisitos funcionales deseados por este que sirvieron de base para realizar una propuesta de solución basada en las necesidades de gestión de información de la UJC en la Facultad 3.
- Las tecnologías y herramientas definidas posibilitaron la creación de un sistema adaptable a diferentes entornos de despliegue y capaz de gestionar información referente a la UJC tales como: la gestión de acta, sanciones y méritos, militantes y organizaciones de base. El sistema además genera reportes de estadísticas como el cómputo de funcionamiento, el parte semanal, el listado de ausentes injustificados, entre otros.
- Se aplicaron las pruebas internas, caja negra y caja blanca, utilizando la técnica partición de equivalencia y camino básico; validando tanto la interfaz como el adecuado funcionamiento interno del software.

Recomendaciones

Como resultado del proceso de investigación y realización de la aplicación han surgido ideas que serían recomendables tener en cuenta para un futuro perfeccionamiento del sistema, a continuación se listan las mismas:

- Agregar un módulo para la gestión de las evaluaciones de los militantes.
- Agregar un módulo para la gestión del universo juvenil.
- Agregar un módulo para la gestión de las actividades de la organización.
- Agregar un módulo para el control y recogida de la cotización.
- Agregar un módulo para la gestión del proceso de crecimiento a la UJC y pase al partido.

Referencias Bibliográficas

1. González-Vallés Saco, G. (2014). *Ciberaula.* Obtenido de http://www.ciberaula.com/articulo/ventajas_poo/
2. Abrahamsson, P. S. (2002). Agile Software Development Methods. Review and Analysis. *VTT Electronics, Juhani Warsta, University of Oulu*, 112.
3. Alexander, C. (1997). *A Pattern Language vol (II).*
4. Álvarez, M. A. (24 de Julio de 2001). *desarrolloweb.com.* Obtenido de http://www.desarrolloweb.com/articulos/499.php
5. Álvarez, M. Á. (25 de Marzo de 2009). *Introducción a jQuery.* Obtenido de http://www.desarrolloweb.com/articulos/introduccion-jquery.html
6. Álvarez, S. (31 de Julio de 2007). *desarrolloweb.com*. Obtenido de http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html
7. *Apache Subversion*. (s.f.). Obtenido de https://subversion.apache.org/
8. *BaseDatosOfimaticas*. (2015). Obtenido de https://basedatosofimaticas.wikispaces.com/3+-+Características+de+los+Sistemas+Gestores+de+Bases+de+Datos
9. Ben Collins-Sussman, B. W. (s.f.). *Control de Versiones con Subversion.* Obtenido de http://svnbook.red-bean.com/en/1.6/svn.intro.whatis.html
10. *Control de versiones con Subversion*. (s.f.). Obtenido de http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html
11. *Doctrine*| *Marco de Desarrollo de la Junta de Andalucía.* (s.f.). Obtenido de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/260
12. Eguiluz, J. (2011). *Desarrollo web ágil con Symfony2.*
13. Eguiluz, J. (s.f.). *Introducción a JavaScript.*
14. *Emagister Engineering.* (20 de Julio de 2012). Obtenido de http://engineering.emagister.com/2012/07/20/formacion-de-twig-con-symfony2-1/

REFERENCIAS BIBLIOGRÁFICAS

15. Enríquez, O. Y., & Gracia del Busto, H. (2011). Mapeo Objeto / Relacional (ORM). *Revista Telem@tica*, 7.

16. Facultad de Derecho. Universidad de Murcia. (13 de Octubre de 2011). *Universidad de Murcia.* Obtenido de http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-habituales.html

17. Facultad de Informática. (s.f.). *Patrones del "Gang of Four".* Universidad Politécnica de Madrid.

18. fergarciac. (25 de Enero de 2013). Obtenido de https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/

19. G. Figueroa, R., Solís, C. J., & Cabrera, A. A. (s.f.). METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. *Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación*, 9.

20. Gamma E., H. R., Software, D. P.-O., & Addison-Wesley. (1995). *Diseño de patrones. Traducción al castellano (2002).* Pearson Educación.

21. Gecko Script Software. (2015). *ExtMX.* Obtenido de http://www.extjs.mx/

22. *Genbeta:dev, desarrollo y software.* (9 de Enero de 2014). Obtenido de http://www.genbetadev.com/herramientas/netbeans-1

23. Group, T. P. (2 de Diciembre de 2012). *¿Qué es PHP?* Obtenido de http://php.net/manual/es/intro-whatis.php

24. Guardado, I. (5 de Mayo de 2010). *web.Ontuts. Tutoriales y Recursos Web.* Obtenido de http://web.ontuts.com/tutoriales/introduccion-a-object-relational-mapping-orm/

25. International, V. P. (2013). *Visual Paradigm for UML product overview.* http://www.visual-paradigm.com/support/documents/vpumluserguide/12/13/5963_visualparadi.html.

26. Kabir, M. J. (2003). *LA BIBLIA DE SERVIDOR APACHE 2.* ANAYA MULTIMEDIA.

27. Labs, S. (2010-2012). *Twig- The flexible, fast and secure template engine for PHP*. Obtenido de http://twig.sensiolabs.org/

28. Luján-Mora, S. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web.* Alicante: Club Universitario.

29. *Oracle*. (s.f.). Obtenido de http://www.oracle.com/technetwork/developer-tools/netbeans/overview/index.html

30. Pacheco, N. (18 de Junio de 2013). *Twig - Manual de Twig*. Obtenido de http://gitnacho.github.io/Twig/

31. Ponjuán, G. (1998). *Gestión de la información en las organizaciones: principios, conceptos y aplicaciones.* Santiago de Chile.

32. *PostgreSQL-es*. (2 de Octubre de 2010). Obtenido de http://www.postgresql.org.es/sobre_postgresql

33. Pressman, R. S. (2005). *La Ingeniería del Software, un enfoque práctico.*

34. Reynoso, C., & Kicillof, N. (Marzo de 2004). Estilos y Patrones en la Estrategia de Arquitectura. *UNIVERSIDAD DE BUENOS AIRES*, 73.

35. Sánchez Castro, A. A. (30 de Agosto de 2014). *michelletorres.mx- Consultoría integral en tecnologías de la información.* Obtenido de http://michelletorres.mx/que-es-un-framework-y-cuales-son-los-principales-en-php/

36. Santa Fe, A. (1998 - 2015). *ALEGSA. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA*. Obtenido de http://www.alegsa.com.ar/Dic/framework.php

37. Sensio Labs. (2010-2012). *Twig - The flexible, fast and secure template engine for PHP*. Obtenido de http://twig.sensiolabs.org/

38. Sensio Labs. (2015). *Sencha.* Obtenido de http://www.sencha.com/products/extjs/up-and-running

39. Software, I. d. (2015). *XP-Extreme-Programing*. Obtenido de http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

40. Sonora, I. T. (s.f.). *ITSON| Introducción a los sistemas de información*. Obtenido de http://biblioteca.itson.mx/oa/dip_ago/introduccion_sistemas/p3.htm

41. Tejeda, C. (s.f.). Metodología de desarrollo de aplicaciones basada en PMBOK y metodologías ágiles de desarrollo. *Universidad Don Bosco, El Salvador*.

42. Redmine. (27 de Marzo de 2015). Redmine. Obtenido de http://www.redmine.org

43. SensioLabs. (2012). Symfony, The Book.

44. Beck, K. (1999). Extreme Programming Explained (1ra edición.). Recuperado el 20 de abril de 2015, de http://software2012team23.googlecode.com/git-history/5127389d21813c2bd955c53999f66cede994578b/docs/literature/Extreme_Programming_Explained_Kent_Beck_1999.pdf

45. Lutz, Mark. 2009. Learning Python, 4th Edition. 4ta. s.l.: O'Reilly Media, 2009. pág. 1216.

46. F. D'Souza, D., & Cameron Wills, A. (1998). Objects, components, and frameworks with UML: the catalysis approach. Boston: Addison-Wesley Longman Publishing Co.

47. Pressman, R. S. (2002). Ingeniería del Software. Un Enfoque Práctico. Quinta Edición. España: MacGraw-Hill.

48. Lorenz, M., & Kidd, J. (1995). Object-Oriented Software Metrics (1ra ed., Vol. 20). (W. Tracz, Ed.) New York, Estados Unidos: ACM New York.

Bibliografía

- 2008. AJAX un nuevo acercamiento a aplicaciones web. [En línea] http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php.
- Andrei, Zeev, Rasmus, Jim y otros. Manual de PHP.
- Bakken, Sæther, Schmid, Stig, Egon. Manual de Php. PHP Documentation Group. [En línea] http://www.php.net/docs.php.
- Pérez, Javier Eguíluz. Introducción a JavaScript.
- 2012. Pérez, Javier Eguíluz. Desarrollo Web Ágil con Symfony2.
- 2011. Traducido por Pacheco, Nacho. Manual de Twig Release 1.1.1
- 2000. Evitts, Paul. A UML Pattern Language
- Rumbaugh, James, Jacobson, Ivar, Booch, Grady. El Lenguaje Unificado de Modelado. Manual de Referencia.
- Pressman, Roger S. Ingeniería del Software. Un enfoque Práctico. Séptima Edición.
- 2013. Romer, M. PHP Data Persistence with Doctrine 2 ORM
- 2014. Reglamento de la Unión de Jóvenes Comunistas para las Organizaciones de Base.

Glosario de Términos

- UJC: Unión de Jóvenes Comunistas.
- AJAX: Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).
- CASE: (Computer Aided Software Engineering), Ingeniería de Software Asistida por Ordenador.
- CCS: Las hojas de estilo en cascada (en inglés Cascading Style Sheets) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).
- Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. 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.
- HTML: Siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.
- Plugin: “Parche” para un programa que le añade características nuevas.
- Release: Nueva versión de una aplicación informática.
- ORM: (Object Relational Mapper) Mapeo de Objeto-Relacional.
- SGBD: (en inglés Database Management System, abreviado DBMS). Sistema Gestor de Bases de datos
- Software: Es un programa o aplicación de que permite a los usuarios el control o realización de varias tareas y que hacen que el trabajo sea más cómodo, rápido y eficiente.
- UML: (Unified Modeling Language) Lenguaje Unificado de Modelado.

Anexos

Anexo 1 - Historias de Usuario

Historia de Usuario	
Número: 2	**Nombre**: Autenticar usuario
Usuario: Militante	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados:** 1
Desarrollador Encargado: Rolando Quintana Safont	
Descripción: Permitirá la entrada al sistema del usuario autenticado y el acceso solo a las acciones que puede hacer según su rol.	
Observaciones:	

Historia de Usuario	
Número: 3	**Nombre**: Gestionar estructura organizativa
Usuario: Documentador	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados**: 1
Desarrollador Encargado: Thais Sánchez Martínez	
Descripción: Debe permitir crear, modificar o eliminar un CB, comités primarios y comités de base independiente.	
Observaciones: Cuando el usuario desee crear una estructura el sistema debe dar la posibilidad de listar las estructuras existentes para que el usuario seleccione cual estará un nivel superior. Solo se mostrará al usuario la posibilidad de crear comités primarios si ya existen comités de base y no debe permitir salir de esta interfaz sin que se le asignen a este comité primario al menos dos comités de base existentes según lo estipulado en las reglas del negocio.	

Historia de Usuario	
Número: 4	**Nombre**: Modificar expediente de un militante
Usuario: Documentador	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados:** 1
Desarrollador Encargado: Rolando Quintana Safont	
Descripción: Debe permitir modificar los datos generales del expediente del militante según las reglas del negocio establecidas.	
Observaciones:	

Historia de Usuario	
Número: 5	**Nombre**: Gestionar sanción a un militante
Usuario: Organizador	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados:** 1
Desarrollador Encargado: Thais Sánchez Martínez	

Descripción: Debe permitir: Insertar sanción a un militante Especificando: - Tipo de sanción - Causa de la sanción. Modificar sanciones a un militante Parámetros a modificar: - Tipo de sanción. - Causa de la sanción. Eliminar la Sanción a un militante
Observaciones: Debe listar Sanciones que ha tenido un militante.

Historia de Usuario	
Número: 9	**Nombre**: Gestionar militante
Usuario: Documentador	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados**: 1
Desarrollador Encargado: Rolando Quintana Safont	
Descripción: Debe permitir Dar alta a militante Modificar militante Dar baja a militante Buscar militante	
Observaciones:	

Historia de Usuario	
Número: 11	**Nombre**: Gestionar Acta
Usuario: Organizador	
Prioridad en Negocio(Alta/Media/Baja): Alta	**Puntos Estimados:** 1
Desarrollador Encargado: Rolando Quintana Safont	
Descripción: Debe permitir Crear acta Modificar acta Eliminar acta Buscar acta Generar listado de acuerdos.	
Observaciones:	

Anexo 2 - Tareas de Ingeniería por HU.

Tarea de Ingeniería	
Número de la Tarea: 2	**Número de la HU: 3**
Nombre de la Tarea: Definición y desarrollo de la interfaz modificar un CB, comités primarios y comités de base independiente.	
Tipo de la Tarea: Desarrollo	
Programador Responsable: Thais Sánchez Martínez	
Descripción: Se realiza el diseño de la pantalla modificar un CB, comités primarios y comités de base independiente. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.	

Tarea de Ingeniería	
Número de la Tarea:3	**Número de la HU: 3**
Nombre de la Tarea: Definición y desarrollo de la interfaz eliminar un CB, comités primarios y comités de base independiente.	
Tipo de la Tarea: Desarrollo	
Programador Responsable: Thais Sánchez Martínez	
Descripción: Se realiza el diseño de la pantalla eliminar un CB, comités primarios y comités de base independiente.	

Tarea de Ingeniería	
Número de la Tarea: 1	**Número de la HU: 4**
Nombre de la Tarea: Definición y desarrollo de la interfaz modificar los datos generales del expediente del militante.	

Tipo de la Tarea: Desarrollo
Programador Responsable: Rolando Quintana Safont
Descripción: Se realiza el diseño de la pantalla modificar los datos generales del expediente del militante. Se crea el diseño realizando una correspondencia entre los tipos de datos a entrar, los datos soportados y los componentes visuales que más se adapten a estos.

Anexo 3 - Tarjetas CRC

Clase Acta	
Responsabilidad	**Colaboradores**
Crear acta Modificar acta Eliminar acta Buscar acta	Acuerdo OrdenDelDía

Clase Sancion	
Responsabilidad	**Colaboradores**
Crear sanción a un militante Modificar sanciones a un militante Eliminar la sanción a un militante Listar las sanciones que ha tenido un militante	Militante EstructuraOrganizativa

Clase Usuario	
Responsabilidad	**Colaboradores**
Crear usuario Modificar usuario Eliminar usuario Buscar usuario	Roles

Clase Militante	
Responsabilidad	**Colaboradores**
Adicionar militante Modificar militante Eliminar militante Buscar militante	Expediente