Universidad de las Ciencias Informáticas

Facultad 1

"Sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A."

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores: Lisbey Borroto Machín

Dayanis Palacio Rondón.

Tutores: Ing. Manuel Alejandro Gil Martín.

Ing. Susej Beovides Luis.

"Ciudad de la Habana, junio del 2010"

"El científico explora lo que existe, y el ingeniero crea lo que nunca ha existido".

Theodore Von Karman

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que hagan el uso que estime pertinente con el mismo.

Para que así conste firmamos el presente a los ___ días del mes de _____ del año _____.

_____	_____
Lisbey Borroto Machín	Dayanis Palacio Rondón
_____	_____
Ing. Manuel Alejandro Gil Martín	Ing. Susej Beovides Luis

Agradecimientos

Primeramente a la persona que es mi razón de ser, la persona que más quiero en el mundo, mi mamá. Te quiero mucho. Gracias por estar siempre ahí cuando necesito de tu ayuda, sin la cual no hubiese sido posible llegar hasta aquí. Por quererme siempre y por caminar junto a mí sin importarte el sacrificio.

Por enseñarme a vencer los obstáculos que se interponen en los caminos por los que atravesamos durante la vida, principalmente en los estudios; por su amor incondicional, por hacer de mí una persona digna y una profesional competente, por sus consejos, su apoyo moral y muy especial, por brindarme toda su confianza. Por ser la mejor madre del mundo. A ella que es la persona más importante en mi vida.

A mi "uelita" linda, que me enseñó que uno crece asimilando lo que deja por detrás, construyendo lo que tiene por delante y proyectando lo que puede ser el porvenir. Por su amor, entrega, dedicación y desvelos, los cuales me dieron el deseo y la fuerza para seguir adelante. Por educarme como lo hizo, por todo. Te quiero mucho.

A mis dos segundas madres, Adela e Iraida, no existe comparación alguna para lo mucho que representan en mi vida, por su cariño y apoyo, por su paciencia y sobre todo por su dedicación, las quiero mucho, gracias por ayudarme siempre en todo.

A mi papá, que ha sabido orientar y guiar mis pasos en el transcurso de mi vida, por su apoyo constante y su confianza en mí.

A mi hermana Yudi, que quiero mucho, por ser mi amiga y llenar mi vida de alegría. Por el enorme compromiso que contraigo con su persona y la confianza depositada que espero no defraudar, ya que ha sido la fuente de mi inspiración y a la que en todo momento he querido servirle de guía e inspiración en su vida.

A los miembros de mi familia, los que siempre han estado presentes en los aspectos fundamentales de mi vida, a los cuales estaré agradecida por guiarme siempre por el camino correcto, por el cariño y el amor que me han dado, por ser tan especiales. A todos los que me ayudaron, me apoyaron y estuvieron siempre a mi lado aun cuando la cima parecía estar más lejos, a aquellos que siempre creyeron en mí y no me abandonaron ni un momento.

A mi nueva familia Ávila, por permitirme formar parte de ustedes y alegrar mi vida con su presencia. Por haberme dado siempre el apoyo incondicional, por estar dispuestos a ayudarme, algo que no tiene precio, algo que solo se paga con el cariño, el respeto, la consideración y el amor, por estar siempre ahí cuando más los he necesitado. Les agradezco por ser el tesoro más grande que la vida me ha dado.

A mis compañeros por estos cinco años, los que han logrado hacer que mi vida en estos años haya sido placentera y alegre, por ser mi familia en la universidad. Por haberme ayudado en los momentos difíciles, por todas las cosas compartidas, gracias por compartir mis penas y duplicar mis alegrías. Y no menciono nombre pero crean que no deja de pasar ni una sola de esas personas en este momento por mi mente, como siempre lo estarán.

A mis amigos de toda la vida que son muy especiales para mí. Por ser partícipes de nuestros logros, desconciertos, malos y buenos momentos en general. Que han sido siempre mi mejor compañía.

A Lisbey, mi compañera de tesis que supo sobrellevar mis etapas de desesperación, por su desempeño, su preocupación, por la confianza que depositó en mí, por estar ahí a pesar de los problemas en el transcurso de la realización del trabajo y que sin ella, no hubiera sido posible culminarlo, sin importar las dificultades por las que atravesamos. Por compartir juntas nuestra última experiencia como estudiantes.

A nuestros tutores, quienes han sido los principales guías en el logro satisfactorio de este trabajo, a los cuales agradecemos las tantas horas de esfuerzo y sacrificio. Por habernos ayudado y estado junto a nosotros en todo momento a pesar de todo el trabajo que les ocupa.

A la UCI: Por permitirme encontrar nuevos amigos.

A todas estas personas de forma general que de una forma u otra han contribuido con su ayuda al desarrollo de este trabajo y a mi preparación como ingeniera y como profesional. Gracias.

Dayanis

Agradezco infinitamente a mi mami y a mi papi por confiar tanto en mí, por ser la inspiración de este sueño que hoy hago realidad, por ser los tutores de mi vida, por darme tanto amor y cariño, por apoyarme incondicionalmente, por ser mis ídolos, gracias de todo corazón, LOS AMO…

A mis abuelitas que aunque no están presentes, sé que anhelaban este sueño igual que yo, gracias por confiar siempre en mí, por adorarme y por estar siempre orgullosas de mi persona… NUNCA LAS OLVIDARÉ…

A mi hermanita linda, gracias por lograr que yo me esfuerce cada día para darte una meta a alcanzar. Gracias por ver mis triunfos como tuyos y tomarlos como desafíos. Te idolatro.

A Yosmel por ser mi tutor desde el inicio de mi carrera, por ser mi guía, mi apoyo, mi compañero, mi amigo, mi novio, gracias por estar siempre ahí cuando todo me parece imposible, gracias por dedicarme tanto tiempo, gracias por darme tanto amor, por amarme, por ser una persona tan especial para mí, por hacerme tan feliz. TE AMO MUCHÍSIMO…

A Melba por ser una amiga excepcional, por su apoyo constante e incondicional, sincero y desinteresado, por ser siempre y por siempre amiga, maestra, tutora y hermana. Gracias por ayudarme a ser hoy una profesional, por ayudarme cuando más lo necesité, gracias por cada segundo de alegría, mil gracias por regalarme esta bella amistad, NUNCA TE OLVIDARÉ.

A Zulema gracias por perfeccionarme tanto, por instruirme desde niña para que lograra ser una profesional, por dedicarme todo el tiempo que necesité, gracias por quererme como me quieres… yo te quiero MUCHO más…

A Neysis, gracias por ese sabio y oportuno consejo que definió mi vida profesional, gracias por confiar que podía alcanzar la meta, a ti, gracias infinitamente.

A mis tutores Susej y Manuel Alejandro (Choni) gracias por su ayuda incondicional, gracias por formar parte de este sueño hecho realidad, mil gracias Choni por dedicarnos parte de tu tiempo, gracias por todo tu apoyo…

A mi familia que adoro, tíos y tías, primos y primas, gracias por brindarme tanto apoyo y tanto amor en especial a mis tías Ada, Rosa, Gladys, Tere e Idania, a mi tío Raúl, a mis primos Mili, Mary, Alietys, Adita, Mirelis, Marialis y Rainel , gracias por esperar y recibir siempre lo mejor de ustedes… Los quiero mucho a todos…

A Yamilka , otra de mis amigas, gracias por ser tan especial, por quererme tanto…

A Narmis, otra gran amiga, gracias por confiar en mí, por darme tanto apoyo, mil gracias por nunca olvidarme…

A Yureilis, gracias por ser quien eres, una de las personas más especiales que he conocido, gracias por ser mi amiga, mi prima, mi hermana, gracias por confiar en mí, NUNCA TE OLVIDARÉ…

A una amiga diferente pero amiga, Ariadna, gracias por tanto cariño y tantos momentos de alegría, gracias por tu ayuda siempre incondicional…

A mi abuelita matancera, Ana Maida, a ti gracias por quererme como una nieta más, por confiar en mí desde tan pequeña, gracias por tus detalles… gracias también a Ana Daima por su apoyo, por su cariño…

A mis profesoras y amigas, Elsita y Dolores, gracias por tener un espacio para mí en su corazón, gracias por quererme como una hija…

A mi amigo Israel por enseñarme tanto del perfil que más disfruto, por ser estar siempre dispuesto a ayudarme… gracias de corazón…

A Danay, gracias por en tan poquito tiempo llegar y quedar para siempre, gracias por tu comprensión, tu sinceridad, por tus chalas tan amenas, por tu perfeccionismo, por tu cariño y tu apoyo incondicional…

A Geidys, gracias por tu inmenso querer, gracias por valorarme y por compartir momentos muy intensos, de alegría y de dolor, gracias primi…

A Dennis, a Mislenis y a Yadiel, gracias por tener la dicha de poder contar con ustedes siempre…

A Aberlardo, gracias por ese cariño, gracias por tener siempre la solución para todos los problemas, mil gracias…

A Yaliana, gracias de todo corazón por ayudarme en los momentos que más lo necesité… gracias por tu carisma que me animó en momentos tan difíciles para mí, gracias por todo…

A Roberto, gracias por tu ayuda tan oportuna y desinteresada…

A Laly, gracias por valorarme tanto y quererme como otra nieta y a Juany, mil gracias por tus lecciones tan sabias que contribuyen a superarme cada día más, por estar siempre dispuesto ayudarme…

A Dayanis, gracias por aceptar ser mi compañera de tesis, gracias por ayudarme a lograr este sueño, por confiar en mí desde el inicio, gracias por enseñarme a ser optimista y por compartir momentos tan intensos juntas.

A mi grupo de primero a tercer año que nunca olvidaré porque de todos aprendí algo, gracias por compartir tantos momentos y formar parte de mi vida universitaria, gracias especialmente a (Ana, Dariel, David, Chanel, Sergio, Nely, Norbert, Milay, Dayli, Yanis), a todos gracias…

A mi suegra Mary y mi abuelita Regla, gracias por poyarme incondicionalmente y quererme como una hija…

A todos los que no menciono pero aportaron su granito de arena en la realización de este trabajo, gracias…

A todos GRACIAS…

Lisbey

Dedicatoria

Dedico mi trabajo de diploma a mi mamá y a mi abuela, con todo mi amor por su apoyo incondicional, cariño y porque gracias a ellas pude hacer mi sueño realidad. Siempre han estado a mi lado, por ser mis amigas, confidentes y ser las personas que me inspiran a ser mejor cada día.

Dayanis

Dedico mi trabajo de diploma a las personas que más quiero en el mundo, a mi mamá, a mi papá, a mi hermana, a mi novio y a mis abuelitas que nunca olvidaré, a todos gracias infinitamente por ayudarme a convertirme hoy en una profesional, siempre estarán donde guardo mi oro líquido, LOS AMO…

Lisbey

Resumen

La necesidad de informatizar los procesos de relatoría en reuniones en Cuba y en el mundo, se torna una realidad y a la vez un reto. Debido al surgimiento de las nuevas Tecnologías de la Información y las Comunicaciones (TIC), el desafío que plantea el momento actual es transformar todas las entidades en centros digitalizados. Cuba, ha estado inmersa en este novedoso proceso de digitalización y en tales circunstancias surgió la Universidad de las Ciencias Informáticas (UCI). La UCI es el resultado legítimo de la cooperación entre varias entidades y un ejemplo es la empresa ALBET.S.A., cuyo origen y desarrollo está vinculado a dicha Universidad, donde ofrece productos y servicios destinados al sector de las TIC. En dicha institución los procesos de relatoría en reuniones son de vital importancia, pues la información que se manipula en estas debe mantener su integridad por las implicaciones en la producción de software que posee dicha empresa y por ende la misma se reúne con gran frecuencia. Este proceso se realiza actualmente de forma manual, lo cual se hace muy engorroso y conlleva a una pérdida sustancial de tiempo, genera gran cúmulo de información, lo que hace difícil la actualización y la búsqueda de la misma en momentos específicos. La presente investigación brinda una propuesta de sistema para optimizar dicho proceso, permitiendo la gestión de las reuniones y acuerdos, así como su seguimiento, la generación de actas automáticamente y la integración con la herramienta de gestión de tareas RedMine utilizada por la empresa.

Palabras clave
ALBET.S.A, Procesos, Relatoría, Reuniones, Sistema.

Índice

Introducción

En la actualidad, la humanidad vive una revolución tecnológica, debido en buena parte a los avances en las tecnologías de la información y las telecomunicaciones. Los grandes cambios que caracterizan a la sociedad de este siglo XXI son: la generalización del uso de las tecnologías, las redes de comunicación, el rápido desenvolvimiento tecnológico y científico y la globalización de la información. Su constante evolución ha sido un elemento importante para el progreso de la humanidad y ha impactado de tal forma las vidas de las personas, que ahora resulta difícil concebir el mundo sin televisión, sin teléfono celular o sin los servicios que brinda el Internet; se vive en una sociedad que sufre los efectos de la digitalización que penetra en todos los sectores de la población. Hoy en día, en la denominada “Era de la información”, se observa que el desarrollo de las TIC y la manera que la sociedad las ha acogido, constituyen factores que favorecen al bienestar socioeconómico de los países. Por su extraordinaria influencia en todos los ámbitos del quehacer humano y el papel que desempeñan dentro del desarrollo social, son un potencial vínculo para lograr mayores niveles de sustentabilidad. Esto convierte a las TIC en un campo multidisciplinario e interdisciplinario con grandes desafíos; pero a su vez, de grandes capacidades en beneficio del desarrollo armónico de las regiones, logrando impactar en la calidad de vida de la población.

Desafortunadamente, el crecimiento acelerado que experimentan las TIC no ha sido homogéneo en todos los procesos que se llevan a cabo en la actualidad. Las exigencias y necesidades de los usuarios crecen día a día, debido a que en un mundo interconectado, en el cual las fronteras y distancias geográficas se han reducido y donde buena parte de las transacciones económicas se realizan por medios electrónicos, las condiciones políticas y socioeconómicas del mundo afectan a las regiones y sus economías. Se puede decir que con el surgimiento de la digitalización de la sociedad es posible el almacenamiento de gran cantidad de información sin la utilización de papel y con poco trabajo manual humano. El desafío que plantea el momento actual es transformar las diferentes entidades en centros digitalizados, teniendo así una mayor seguridad e integridad de la información, garantizando el aumento de la productividad, que las respuestas sean rápidas y eficientes y que la población y el personal de las empresas tengan un alto desarrollo en su capacidad y conocimiento.

Cuba ha estado inmersa en el profundo y novedoso proceso de informatización del país y en tales circunstancias surge la idea de crear la Universidad de las Ciencias Informáticas (UCI), siendo esta protagonista de este proceso y teniendo como misión además, formar profesionales calificados en la rama de la Informática en cada sitio del país, a partir de un modelo pedagógico flexible, que vincula dinámica y coherentemente, el estudio con la producción y la investigación. Este proyecto UCI es un

resultado legítimo de la cooperación entre varias organizaciones del país, un ejemplo es la empresa ALBET, S.A. cuyo origen y desarrollo es vinculado estrechamente a dicha Universidad y con la participación de otras entidades, ofrece y comercializa soluciones integrales, productos y servicios asociados al sector de las tecnologías de la información y las comunicaciones.

Desde sus inicios en esta empresa, como en la gran mayoría de las entidades, existen procesos que se llevan manualmente y en los cuales la manipulación de la información es engorrosa y compleja, existiendo la posibilidad de intromisión de errores humanos en el uso y gestión de la misma. Uno de esos procesos es el levantamiento de actas en las reuniones, siendo de vital importancia por las implicaciones que tienen en la productividad. Las actas constituyen un documento elaborado en la mayoría de los casos, con fines legales, para testimoniar acuerdos y características del desarrollo de eventos tales como: asambleas, reuniones, juntas, consejos, etcétera. Generalmente las actas son realizadas por un secretario que tiene tal responsabilidad o alguno de los asistentes asignado para tal fin, a los cuales les resulta tedioso el proceso de relatoría de forma manual y en ocasiones no se toman los criterios o acuerdos con la precisión requerida. El personal encargado de la elaboración del acta, a veces, no conoce una forma estandarizada de realización.

Después de haber hecho un análisis de la situación problémica existente y ante la necesidad de encontrar una solución para dar respuesta a tales dificultades, se plantea resolver el siguiente **problema científico**: ¿Cómo gestionar el proceso de relatoría en reuniones de ALBET, S.A.?

En correspondencia con el problema planteado a resolver se define como **objeto de estudio** el proceso de relatoría en reuniones, restringiendo el **campo de acción** al proceso de relatoría en reuniones de la empresa ALBET, S.A.

Para dar solución al problema antes mencionado se define como **objetivo general** desarrollar un sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A. e integrarlo con el sistema de gestión de tareas de la empresa. Para dar cumplimiento al objetivo general se han trazado los siguientes **objetivos específicos**:

- Analizar la estructura y el funcionamiento de los procesos de relatoría en reuniones de ALBET, S.A.
- Realizar el diseño del sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A.
- Implementar el sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A.
- Integrar el sistema de apoyo al proceso de relatoría en reuniones con el sistema de gestión de tareas utilizado en la Empresa.
- Validar la solución propuesta con el usuario final.

En la presente investigación se asume la siguiente **hipótesis**: Si se desarrolla un sistema que gestione los procesos de relatoría en reuniones de ALBET, S.A. y este se integra con el sistema de gestión de tareas (RedMine), entonces se obtendrá un mayor nivel de gestión de los documentos, facilitando el proceso de relatoría en reuniones.

Variables:

- Independientes:
  - Sistema para la gestión de los procesos de relatoría en reuniones de ALBET, S.A.
  - Integración con el sistema de gestión de tareas (RedMine).
- Dependientes:
  - Nivel de gestión de los documentos.
  - Facilidad en el proceso de relatoría en las reuniones.

Operacionalización de las variables.

Variable Conceptual	**Dimensión**	**Indicadores**	**Subindicadores**
Independientes	Proceso de relatoría de reuniones en ALBET, S. A.	Tiempo de elaboración de acta.	Minutos
Sistema para la gestión de los procesos de relatoría en reuniones de ALBET, S.A.		Tiempo para tomar notas.	Minutos
		Cantidad de errores cometidos.	Unidades
Integración con el sistema de gestión de tareas (RedMine).		Tiempo de gestión.	Minutos
Dependientes	Proceso de relatoría de reuniones en ALBET, S. A.	Tiempo de gestión	Minutos
Nivel de gestión de los documentos.		Tiempo.	Minutos
Facilidad en el proceso de relatoría en las reuniones.			

Tabla 1: Operacionalización de las variables.

Para el cumplimiento de los objetivos trazados se han propuesto el conjunto de **tareas de la investigación** que se relacionan a continuación, las cuales ayudarán a que el trabajo fluya de forma eficiente:

- Caracterización de los procesos de negocios de la relatoría en reuniones de la Empresa.
- Caracterización de la metodología de desarrollo de software para el modelado del sistema.
- Definición de las tecnologías y herramientas que permitan obtener una solución acorde al problema planteado.
- Levantamiento de requisitos.
- Definición de la arquitectura a utilizar.
- Diseño de las funcionalidades del sistema.
- Implementación del sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A.
- Integración con la herramienta de gestión de tareas.
- Realización de pruebas a la aplicación.

Estrategia de investigación

La estrategia que se utiliza es la **descriptiva**, esta permite descubrir la esencia del proceso de relatoría en reuniones, así como comprender el valor científico de los resultados obtenidos a partir de la profundidad teórica de la investigación.

En la realización de la presente investigación se utilizaron los siguientes **métodos de investigación científica**:

Métodos Teóricos: Permiten comprender el fenómeno que se estudia, su evolución y proponer las mejoras respectivas a los problemas identificados, entre ellos están:

- **Histórico-lógico:** Este método permite realizar la primera parte de la investigación, haciendo un análisis del estado del arte de la problemática; se analizan las ventajas y desventajas de cada una de las herramientas y las tendencias en la resolución de esta problemática. Se utiliza para estudiar todo lo referente al proceso de relatoría de reuniones en ALBET, S.A. Su empleo permite el desarrollo evolutivo y coherente en el estudio de la metodología orientada a objetos, patrones de diseño, herramientas de desarrollo de software y sistemas.
- **Analítico-Sintético:** Se utiliza para resumir, sintetizar y procesar la información recopilada en el estudio realizado, centrándose en el análisis de las teorías; permitiendo la extracción de los elementos más importantes de manera que se procese la información y se elaboren conclusiones, para luego sintetizarlas en la solución propuesta. Facilita el entendimiento del proceso de relatoría de reuniones en ALBET, S.A.

- **Modelación:** Su utilización permite crear abstracciones que explican la realidad, por ejemplo: todos los modelos y diagramas presentados.

Métodos Empíricos: Permiten describir y explicar las características del fenómeno en estudio. Dentro de estos, se aplicaron métodos particulares con el objetivo de recolectar los datos necesarios para identificar la problemática y las causas de esta, así como determinar la magnitud de su influencia, estos son:

- **Observación:** Con el objetivo de ampliar la información obtenida a través de las entrevistas realizadas.
- **Revisión de documentos**: Se realiza para la determinación del estado del arte del objeto de investigación.
- **Entrevistas:** Se realizan diferentes entrevistas a los clientes para dar cumplimiento a las tareas y objetivos planteados como vía de obtención y elaboración de los datos, así como el levantamiento de requisitos, posibilitando que los conocimientos que puedan brindar las personas calificadas en el tema a tratar, sean de carácter imprescindible para cumplir con los objetivos planteados anteriormente.

El presente trabajo se encuentra estructurado en cuatro capítulos que abordan los siguientes contenidos:

Capítulo 1: "Fundamentación teórica": Se describirán los principales conceptos y términos abordados en la investigación. Se investigará el estado del arte del tema tratado a nivel nacional e internacional, tecnologías, metodologías y software que se utilizarán para darle solución al problema planteado.

Capítulo 2: "Características del sistema": Se describirá el flujo actual del proceso de relatoría en reuniones de ALBET.S.A. Se realizará el modelado de negocio y el del sistema, donde se definirán los requisitos funcionales y no funcionales. Además, se detallarán las características del sistema que se propone.

Capítulo 3: "Diseño del sistema": Se describirá la arquitectura a utilizar, así como el empleo de los patrones de diseño. Se delineará el sistema a través del diseño, enfocado a cómo este cumple sus objetivos teniendo en cuenta los requisitos funcionales y no funcionales. Se obtendrán los artefactos de dicho flujo.

Capítulo 4: "Implementación y validación": Se describirá el flujo de implementación del sistema, obteniéndose los artefactos necesarios para la realización del mismo. Además, de concebir y realizar la validación del sistema a través de pruebas a la aplicación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

Encontrar una solución eficiente a los problemas relacionados con el levantamiento de actas en eventos tales como asambleas, reuniones, juntas, consejos, etcétera, se ha convertido en un reto trazado por generaciones. El control y la manipulación de la información contenida en las actas se convierten en un factor importante a tener en cuenta en la totalidad de las instituciones, para lograr la organización y correcto funcionamiento de las mismas.

En el presente capítulo se brinda una breve panorámica sobre los sistemas informatizados existentes a escala nacional e internacional, haciendo un análisis comparativo entre los requerimientos del cliente y las soluciones ya existentes. Se describirán las tecnologías actuales de desarrollo, las herramientas y metodologías a utilizar abordando sobre su importancia y las ventajas de uso que brindan en el desarrollo del sistema.

1.2 Proceso de relatoría en reuniones.

1.2.1 ¿Qué es una reunión?

Se entiende por reunión a la agrupación de varias personas en un momento y espacio dado, voluntaria o accidentalmente. La reunión es una de las expresiones más características de todo ser vivo que se considere gregario y esto es especialmente importante en el caso del ser humano. La reunión de diferentes individuos en un lugar y momento específico puede llevarse a cabo de manera planificada, con un objetivo delimitado y con un tiempo de duración planeado, pero también puede darse de manera espontánea, por razones casuales y sin mayores propósitos. En ambos casos se habla del encuentro de dos o más personas que por lo general se conocen previamente, aunque no siempre sucede esto.

Uno de los casos más comunes de reunión es aquel que tiene que ver con cuestiones profesionales o laborales. En este ámbito, las personas reunidas se juntan para trabajar sobre elementos en común y delinear futuras acciones, para hacer un análisis del trabajo realizado o por realizar, para asignar tareas o para controlar diferentes aspectos de la labor de todos. En estos casos, las reuniones suelen ser formales, con un vocabulario específico y relativo a la temática a desempeñar (CECILIA, 2009).

La reunión es una técnica que permite que personas con objetivos comunes intercambien información y lleguen a compromisos colectivos para conseguir las metas.

1.2.2 ¿Qué es un acta?

La palabra "acta" viene del latín y significa "los hechos"; luego, un "acta" no es más que un testimonio escrito de los hechos ocurridos en cualquier circunstancia: una reunión de consorcio, una asamblea de miembros de una comisión directiva de cualquier entidad, una certificación del nacimiento de una persona, etcétera. Es decir, hechos que se asientan por escrito y que resulta importante registrar y conservar (JANI, 2009).

Un acta es una certificación o testimonio escrito en la cual se da cuenta de lo sucedido, tratado o pactado en oportunidad de cualquier circunstancia que lo amerite, como es el caso de: la elección de una persona para un cargo que puede ser público o privado, la reunión del directorio de una empresa u organización, la asamblea de miembros de una comisión directiva de cualquier entidad, una certificación del nacimiento de una persona o cualquier otro hecho que requiera o exija de la correspondiente certificación legal de que ocurrió (FLORENCIA, 2009).

La redacción de un acta es impersonal, de forma que sólo se recogen alusiones personales en las intervenciones, en las que el propio individuo pide expresamente que conste en el acta o bien cuando se trata de algo relevante o se considera necesario establecer con claridad, quién o quiénes sustentan determinadas opiniones o realizan las intervenciones. Un acta es un documento empleado por gran número de personas y profesionales con diferentes fines. Para un trabajador social es esencial en las reuniones profesionales o en aquellas que se tratan temas concretos para abordar tareas.

Como sucede con otros documentos públicos, el acta deberá contar con una serie de datos que serán determinantes a la hora de evaluar su validez, en tanto y generalmente, la misma está labrada por un profesional que se conoce como escribano y que como tal está facultado para proceder en la confección de la misma. Entre los datos que obligatoriamente deberán quedar asentados en este documento se encuentran: datos del lugar donde se elabora, fecha y hora de inicio y de culminación, una breve introducción sobre el hecho a que se refiere o el motivo que da lugar a su escritura y luego, en lo que se denomina como cuerpo, el escribano hará un detalle pormenorizado de lo que sucedió en el acto o reunión en cuestión y que motivaron la celebración de la pertinente acta (…). Si hay votaciones, se describe en detalle por lo que se vota y resultado de la votación. Para su cierre, se usa generalmente un párrafo "de estilo", señalando que al pie firmarán los presentes "prestando su conformidad" a lo actuado (FLORENCIA, 2009).

Existen varios formatos para la confección de actas, este lo determina la institución y el tipo de reunión o junta. La persona responsable de dicha confección debe conocer cuando utilizar cada uno.

1.2.3 ¿Cómo redactar un acta de reunión?

El proceso de redacción de un acta tiene tres partes diferenciadas. La más importante es la trascripción, que es la anotación escrita fiel a lo que se habla en la reunión. Hay que saber qué anotar, identificar a quién expone cada punto y tener claro cuál es el acuerdo. Esto en teoría es bastante simple, pero en pleno debate en una reunión, suele ser más engorroso. La trascripción es un documento en bruto, sobre el cual se va a redactar el acta, no tiene que estar ordenado necesariamente. Luego se le dará la forma y orden según el modelo establecido por la institución.

El acta es a la vez un documento de trabajo y de archivo, que ofrece la ventaja adicional de establecer las prioridades del discurso de forma visible. En este, aparecen las síntesis de cada punto debatido y acuerdo al que se ha llegado. Por esa razón son fundamentales la forma, el orden del desarrollo de los puntos, el desglose de los mismos y una redacción adecuada.

1.2.4 ¿En qué consiste el proceso de relatoría en reuniones?

El proceso de relatoría es el momento propio de escritura de todo lo que se considera citable (...) (JULIO, 2009).

El proceso de relatoría en reuniones es el proceso de tomar notas en las reuniones. Es muy habitual que tras unos cuantos días de lo sucedido en una reunión esté todo envuelto en una niebla y no se recuerde exactamente qué es lo que le interesaba al interlocutor o al cliente. Esto es especialmente cierto y relevante cuando estamos hablando de convenios, dado que en negociaciones complejas es muy recomendable incluir las notas de reuniones en actas, lo más detalladas posibles y que no sólo representen los aspectos objetivos, sino también los subjetivos como pueden ser: reacciones, posturas, entre otros.

Grabar las reuniones o tomar notas, para después hacer un resumen de los compromisos acordados es una buena opción, pero el tiempo a veces no lo permite. Idealmente y cuando se trate de reuniones vinculadas a un proyecto real, se debe procesar posteriormente las notas y reflejarlas en un acta, donde se registren principalmente los cambios, ya sean nuevas acciones, tareas a desempeñar, información adicional a suministrar o elementos que se han desestimado. El acta deberá ser firmada por el cliente y el proveedor, de esta forma no se corre el riesgo de perder información importante.

En general, las notas de una reunión deberían reflejar al menos los siguientes puntos:

- Tareas o acciones que se han acordado que se deben desarrollar. Se debe cumplir las 4 preguntas importantes (¿Qué?, ¿Quién?, ¿Dónde?, ¿Cuándo?). Es imprescindible que se defina la "Siguiente Acción", ya que no solamente se debe definir en abstracto lo que se debe hacer ("Qué"), sino que debe quedar claro cuál es la tarea exacta incluyendo la información de apoyo para desarrollarla. Así mismo, se debe dejar muy claro la fecha prevista de compromiso ("Cuándo"), el responsable de la tarea ("Quién") y si es necesario, el lugar ("Dónde").
- Información que no ha quedado clara o preguntas que no se le han dado respuesta y se deben ampliar.
- Información de soporte proporcionada por el interlocutor que pueda ser útil más adelante.
- Datos o información que se han solicitado.
- Aspectos que han preocupado al interlocutor o interlocutores o que se ha percibido que son importantes.
- Detalles en los que el interlocutor hace especial énfasis y no se le había dado relevancia.

En resumen, siempre se debe tomar notas, aún en las reuniones más prolijas es posible registrar lo hablado. Según lo anterior se puede decir que las principales razones para tomar actas en las reuniones son: cumplir los compromisos, conocer, recordar los temas tratados y asumir comentarios anteriores para tomar nuevas decisiones.

El encargado del proceso de relatoría sería el secretario o secretaria, elegida democráticamente para esta función. Esta persona no deberá estar haciendo al mismo tiempo otras funciones. Una opción sería que el encargado de este proceso no siempre fuera la misma persona, que esta responsabilidad puede rotar entre el grupo de participantes. El acta puede ser tomada exclusivamente de los acuerdos tomados, en tanto que suponen compromisos a asumir y legitiman las acciones a desarrollar por los miembros. Hay que tener precaución en la redacción para ser fieles a la decisión y no dar lugar a equívocos. Se puede señalar el grado de acuerdo, registrando el número de votos si no hubo consenso. Las actas deben estar al alcance de todos los miembros del grupo para poder consultarse en todo momento, pero con el privilegio de consultarlas no de modificarlas, por seguridad.

1.3 Sistemas de software existentes que facilitan el proceso de relatoría en reuniones.

- **Sistema de control de actas para reuniones y asambleas.**

ALFASA[1] ha preparado un sistema para el manejo de actas para todo tipo de organizaciones, sean estas del tipo empresas en general, asociaciones solidarias, municipalidades, cooperativas, instituciones estatales e instituciones autónomas. Su objetivo general como un sistema computarizado es llevar un control y seguimiento de actas de cualquier tipo de organización, con el fin de dar una mayor utilidad funcional a estas y a los elementos que la componen, de esta forma, optimizar la integridad de la información palpada en las mismas para generar una mayor credibilidad en su utilización. Este cuenta con un sistema de seguridad que ejecuta un manejo de usuarios y privilegios tanto para introducción de datos como para la solicitud y obtención de información mediante una autenticación al inicio de la aplicación. Dicho sistema incluye un módulo que permite la introducción de la información referente a cada acta con un formato previamente especificado y que contiene toda la información anexa de la junta o reunión. Permite el seguimiento del estado de los acuerdos tomados en cada sesión con el fin de controlar cuáles acuerdos se han cumplido en base a lo estipulado y cuáles están pendientes para indagar qué acciones se han presentado que justifiquen su incumplimiento. El sistema genera un manejo adecuado de consultas y reportes que permite solicitar datos adecuados de los documentos archivados en el formato y tiempo que se requiere con el fin de acelerar el proceso de obtención y análisis de la información. Genera además, un documento de cada acta con formatos variados, inicialmente tipo tabulado o carta (ALFASA, 2008).

Los programas de ALFASA pueden instalarse en una diversidad de equipos y plataformas e inclusive trabajar en ambientes distribuidos de una forma transparente, permitiendo al usuario seleccionar y/o utilizar sus recursos sin perder la cultura informática de su empresa. Este sistema de control de actas está basado en estándares Windows y desarrollado en la plataforma integral de desarrollo Visual Studio 2005. Con una base de datos versátil y confiable diseñada en SQL Server, que permite el acceso solo a los usuarios previamente registrados, teniendo en cuenta además que el uso de la tecnología .NET permite al sistema operar e interactuar tanto por si solo como por otros módulos según la necesidad del cliente. Al ser un software propietario la empresa garantiza la calidad bajo normas internacionales ISO-9001.

- **ActaJun**

[1] Empresa líder en el desarrollo e implantación de sistemas de información.

Es una aplicación informática creada por la empresa CISI S.L[2] la cual permite gestionar fácilmente juntas, reuniones, asambleas entre otros eventos. Envía a través del correo electrónico una vez concluido el encuentro la relación de asistentes a la misma, registra votaciones realizadas durante el evento calculándolas automáticamente y acuerdos tomados durante el mismo. Facilita el desarrollo de los puntos del orden del día para luego confeccionar el acta mediante la aplicación con todo lo antes mencionado, incluyendo además la fecha, la hora y el lugar donde fue realizado el evento, con un formato personalizado que brinda la aplicación. El software cumple con la normativa actual de protección de datos, el acceso al sistema está protegido por contraseña de usuario (L., 2008).

ActaJun se vincula con Seguiges (Gestión de expedientes e incidencias) y Cfin H (Gestión de comunidades, propietarios y urbanizaciones) convirtiéndose esta en una herramienta clave en un despacho de administración de comunidades. Para su uso es necesario disponer del acceso a Internet y a correo. Es propietario y está disponible en versión mono-usuario y multi-usuario. CISI S.L. garantiza el buen funcionamiento de la misma con mantenimiento y actualización sistemática de la aplicación, respondiendo a las necesidades de gestión de sus clientes (profesionales de la administración de fincas y del sector inmobiliario en general). Brinda una interfaz ágil e intuitiva facilitando al usuario su comprensión y uso, permite emitir informes según el idioma de la comunidad y del copropietario, con una arquitectura diseñada para el óptimo rendimiento de las aplicaciones.

Independientemente de los beneficios o ventajas que puedan ofrecer los sistemas anteriormente mencionados y ante la situación existente que se presenta en dicha investigación, se deduce que estos software no satisfacen las necesidades del cliente, pues no se integran con la herramienta RedMine capaz de controlar y gestionar tareas, siendo tan eficaz para la gestión de proyectos y por ende tan imprescindible para la empresa. ALBET.S.A requiere que durante una reunión, junta o cualquier otro evento permita la toma de notas para la posterior confección del acta sin conexión de red, permitiendo a la vez almacenar dichos datos en una base datos embebida que debe poseer la aplicación; almacenando conjuntamente los acuerdos tomados en cada sesión, con el fin de controlar cuáles acuerdos se han cumplido y cuáles están pendientes. El cliente requiere que el software sea fácil de portar y además que posea un completamiento de frases que posibilite la toma de nota más eficiente y con mayor rapidez.

[2] Empresa de Comunicaciones y Sistemas Informáticos.

1.4 Metodologías de desarrollo de software.

En el ciclo de vida de un software se desarrollan una serie de tareas para obtener un producto de software. Cada tarea puede ser resuelta de varias formas, utilizando herramientas y técnicas diferentes. Las metodologías de desarrollo de software son las que definen cómo debe desarrollarse una tarea determinada y cuándo podemos darla por concluida, quién debe hacer qué, cuándo y cómo debe hacerlo, cuáles son las tareas que preceden o anteceden a una dada y qué documentación utilizar. Estas pueden ser pesadas o ágiles y posibilitan que todos los integrantes de un equipo sigan un criterio común a la hora de realizar las actividades.

Las metodologías ágiles son efectivas para modelar y documentar un proyecto de software y los clientes forman parte el equipo de desarrollo. Estas son conscientes de que el mundo del software está sometido a un constante cambio y avance, siendo lo suficientemente adaptable como para aplicarse en distintos proyectos, principalmente si estos son pequeños. Las más aceptadas y por ende, más utilizadas son: XP (Extreme Programing) y Scrum por sus características y ventajas en el progreso de desarrollo de un producto.

Las llamadas metodologías pesadas imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Entre las principales y más conocida de las metodologías pesadas está RUP (Rational Unified Process), siendo una de las más importantes para alcanzar un grado de certificación en el desarrollo del software.

- **Rational Unified Process (RUP)**

  El proceso unificado de desarrollo (RUP) es una metodología tradicional o pesada que para la ingeniería de software va más allá del mero análisis y diseño orientado a objetos, para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. Representa una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas, probadas y una arquitectura configurable. Estas prácticas constituyen un conjunto de procesos de ingeniería de software que permiten seleccionar fácilmente el conjunto de componentes de

procesos que se ajustan a las necesidades específicas del proyecto. Se podrán alcanzar resultados predecibles unificando el equipo con procesos comunes, que optimicen la comunicación y creen un entendimiento común para todas las tareas, responsabilidades y artefactos, de forma que cada miembro del equipo comprenda su contribución al proyecto. Esto ayuda a simplificar la comunicación, asegurando la asignación de recursos en forma eficiente, la entrega de los artefactos correctos y el cumplimiento de las tareas en el tiempo establecido.

Características principales de RUP

- **Guiado por los casos de uso**: Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- **Centrado en la arquitectura**: Los modelos son proyecciones del análisis y el diseño, constituye la arquitectura del producto a desarrollar, por lo que le da a los desarrolladores una mayor visibilidad del sistema.
- **Iterativo e incremental:** Durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo (SULLON, 2006).

Una particularidad de esta metodología es que en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

- **Extreme Programing (XP)**

Es una de las metodologías de desarrollo de software ágiles más exitosa de las utilizadas en la actualidad para proyectos de corto plazo y equipo pequeño. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP, la metodología se basa en:

1. Pruebas Unitarias: Son pruebas realizadas a los principales procesos, de tal manera que si se adelanta en algo hacia el futuro, se puedan realizar pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

2. Refabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
3. Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro desarrolla la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

¿Qué es lo que propone XP?

1. Empieza en pequeño y añade funcionalidad con retroalimentación continua.
2. El manejo del cambio se convierte en parte sustantiva del proceso.
3. El costo del cambio no depende de la fase o etapa.
4. No introduce funcionalidades antes que sean necesarias.
5. El cliente o el usuario se convierte en miembro del equipo.

Lo fundamental en este tipo de metodología es:

1. La comunicación, entre los usuarios y los desarrolladores.
2. La simplicidad al desarrollar y codificar los módulos del sistema.
3. La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales (MORGADE, 2009).

- **Scrum**

Es un proceso de desarrollo de software iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software, más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, de forma que los "jefes" puedan ver día a día cómo progresa el trabajo.

Un principio clave de Scrum es el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan. Por sus características no es válida para cualquier proyecto o equipo de personas, según muchos especialistas de esta metodología, es óptima para

equipos de trabajo de hasta 8 personas, aunque hay empresas que la han utilizado con éxito con equipos más grandes.

Las acciones de Scrum forman parte de un ciclo iterativo repetitivo, por lo que el mecanismo y la forma de trabajar tienen como objetivo, minimizar el esfuerzo y maximizar el rendimiento en el desarrollo.

1.5 Lenguaje de modelado unificado (UML: Unified Modeling Language)

El uso de los lenguajes de modelado facilita a los desarrolladores de software representar visualmente el sistema a construir. UML permite modelar, construir y documentar los elementos que forman un sistema software. Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa. Cuenta con una notación estándar y simétrica esencial para el modelado de un sistema orientado a objetos, teniendo como meta principal el avance en el estado de la integración institucional, proporcionando herramientas de interoperabilidad para el modelado visual de objetos. Tiene como objetivo permitir visualizar de una forma gráfica un sistema de tal manera que pueda ser entendido por todos, especifica cuáles son las características antes de que comience a ser construido. Todos estos elementos gráficos sirven como documentación del sistema y de igual manera servir para posteriores revisiones. Prácticamente todas las herramientas CASE y de desarrollo la han adaptado como lenguaje de modelado.

1.6 Tendencias y tecnologías actuales a considerar.

La informatización de la sociedad en los últimos años ha sido el motor impulsor del desarrollo de nuevas tecnologías, metodologías y herramientas para facilitar este propósito. En el ámbito mundial existen herramientas que permiten lograr en tiempo récord el desarrollo de grandes sistemas que manipulan un amplio volumen de información, además de posibilitar un alto grado de calidad gracias a las facilidades que implementan dichas herramientas.

En la actualidad para realizar el desarrollo de un software existen ciertos criterios a tener en cuenta como son: los lenguajes de programación a utilizar, los gestores de bases de datos, las herramientas CASE, los entornos de desarrollo, entre otros.

Un lenguaje de programación es el lenguaje artificial formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, utilizado para crear programas que controlen el comportamiento físico y lógico de una máquina, así como para expresar algoritmos con precisión, o como modo de comunicación humana (JUAN, 2007).

Las herramientas CASE son aquellas aplicaciones informáticas destinadas al desarrollo del software y sirven de apoyo durante todo el ciclo de vida del mismo, facilitando la realización del diseño del proyecto y aumentando la productividad y la calidad del software, ya que el costo de las mismas en términos de tiempo y dinero se reducen. Se puede diferenciar algunas por su funcionalidad y las más conocidas son los editores de UML (ERICK, 2006).

Los Sistemas de Gestión de Base de Datos (SGBD) son software dedicados a manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización y sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

A continuación se presenta el resumen realizado durante la investigación de las propuestas tecnológicas que más se adaptan a los requerimientos del sistema propuesto.

- **Visual Paradigm**

Es una Suite de herramientas que figura también dentro de las llamadas CASE. Es fácil de usar y soporta el ciclo de vida completo del desarrollo de software: negocio, análisis y diseño orientado a objetos, construcción, pruebas y despliegue (MARTÍNEZ, 2009).

Visual Paradigm permite realizar tanto la ingeniería directa como la inversa, generación de código, importación desde Rational Rose, exportación/importación, generador de informes, editor de figuras, integración con IDE Visual Studio, Eclipse, NetBeans y otros. La transición desde el análisis al diseño y después a la implementación es cuidadosamente integrada dentro de la herramienta CASE, de esta manera, se reduce significativamente el esfuerzo en todas las etapas del ciclo de vida del desarrollo del software. Está diseñada para distintos usuarios entre los que se incluyen ingenieros de software, analistas de sistemas, analistas de negocios, arquitectos y desarrolladores. Está orientada a la creación de diseños y se usa el paradigma de programación orientada a objetos.

- **Rational Rose Enterprise**

Rational Rose Enterprise es el producto más completo de la familia Rational Rose que proporciona un lenguaje común de modelado para el equipo y facilita la creación de software de calidad. Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. El navegador UML de

Rational Rose permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, vista lógica, vista de componentes y vista de despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Características adicionales:

- Capacidad de análisis de calidad de código.
- Soporte Enterprise Java Beans 2.0
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo (LUIS, 2007).

.NET

Es una nueva filosofía en cuanto a entornos de desarrollo, multi-lenguaje para la construcción, distribución y ejecución de Servicios Web y aplicaciones. Es una plataforma diseñada para simplificar el desarrollo de aplicaciones. El corazón de esta plataforma.NET es el CLR (Common Language Runtime), entorno de ejecución común, que es una aplicación similar a una máquina virtual que se encarga de gestionar la ejecución de las aplicaciones para ella escritas. A estas aplicaciones les ofrece numerosos servicios que facilitan su desarrollo, mantenimiento, favorecen su fiabilidad y seguridad. Entre ellos los principales son:

- Modelo de programación consistente y sencillo, completamente orientado a objetos.
- Ejecución multiplataforma
- Recolección de basura.
- Aísla la memoria entre procesos y brinda comprobaciones automáticas de seguridad de tipos en las conversiones.

- Soporte multi-hilo.
- Gestión del acceso a objetos remotos que permite el desarrollo de aplicaciones distribuidas de manera transparente a la ubicación real de cada uno de los objetos utilizados en las mismas.
- Seguridad avanzada, hasta el punto de que es posible limitar los permisos de ejecución del código en función de su procedencia (Internet, red local, CD-ROM, etc.), el usuario que lo ejecuta o la empresa que lo creó (MORGADE, 2009).

- **Microsoft Visual Studio 2008 Express**

Es la versión gratuita del entorno de programación creado por Microsoft para desarrollar aplicaciones y web para sistemas Windows. Ideal para principiantes, es compatible con todos los lenguajes de programación desarrollados por Microsoft: Visual Basic, C++, C#, SQL Server y J#. Cada uno de estos lenguajes viene con su propia interfaz personalizada, compartiendo librerías y herramientas como el autocompletado de código, asistentes de configuración y esquemas de organización que hacen de este entorno de programación bastante más productivo que sus predecesores. Con Visual Studio 2008 Express se logrará crear aplicaciones para Windows, desarrollar sitios y aplicaciones web innovadores.

- **NetBeans**

Es un entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java. Su aprendizaje es fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma, debido a que el funcionamiento de cualquier programa creado en este entorno será igual en cualquier sistema operativo en el que sea instalado.

Mediante NetBeans es posible diseñar aplicaciones con solo arrastrar y soltar objetos sobre la interfaz de un formulario y no solo es posible elaborar potentes aplicaciones de escritorio, también para la Web y para dispositivos portátiles, como móviles, sin que cambie la forma de programar.
La programación mediante NetBeans se realiza a través de componentes de software modulares, también llamados módulos los cuales se ponen a disposición del usuario a través de su página web. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

1.7 Lenguaje de programación

- **Java**

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (FILIBERTO, 2009).

Gracias a su versatilidad, eficiencia y portabilidad, Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital.

A pesar de sus grandes ventajas, Java también cuenta con un número de desventajas que deberán ser analizadas en el momento de tomar una decisión. Los programas en Java tienden a ser mucho más lentos que sus equivalentes escritos en otros lenguajes de programación.

- **C#**

C# es un lenguaje completamente orientado a objetos, con una gran cantidad de mejoras sobre sus predecesores. Es el lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Se dice que su competir más cercano es Java, lenguaje con el que guarda un enorme parecido, aunque C# incorpora muchos elementos de los que Java carece como: el rendimiento, el cual es mejor, soporta más tipos primitivos, incluyendo tipos numéricos sin signo, compilación condicional, aplicaciones multi-hilo simplificadas, entre otros. Las principales características que lo definen son:

- **Sencillez de uso:** Elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y compresión.

- **Modernidad:** Es un lenguaje de última generación, incorpora elementos que son muy útiles para el programador, como tipos decimales o booleanos.
- **Orientado a objetos:** Soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.
- **Orientado a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular.
- **Recolección de basura:** Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura.
- **Eficiente:** Todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros.
- **Compatible:** Para facilitar la migración de programadores de C++ o Java a C#, no sólo se mantiene una sintaxis muy similar a la de los dos anteriores lenguajes, sino que también ofrece la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos.
- **Unificación de tipos:** En C# todos los tipos derivan de una superclase común llamada System.Object, por lo que automáticamente heredarán todos los miembros definidos en esta clase. A diferencia de Java, en C# esta característica también se aplica para los tipos básicos.
- **Extensión de los operadores básicos:** Para facilitar la legibilidad de código y conseguir que los nuevos tipos de datos que se definan a través de las estructuras, estén al mismo nivel que los elementos predefinidos en el lenguaje, al igual que C++ pero a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores (incluidos el de la conversión) cuando se apliquen a diferentes tipos de objetos (FILIBERTO, 2009).

Algunas de estas características no son propias del lenguaje, sino de la plataforma .NET, aunque se listan aquí ya que tienen una implicación directa en el lenguaje.

1.8 Sistemas gestores de bases de datos.

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. La bibliografía especializada a menudo se refiere a estos sistemas como SGBD o DBMS, siendo ambos equivalentes y acrónimos de Sistema Gestor de Bases de Datos y DataBase Management

System, respectivamente. En la actualidad existe una gran variedad de SGBD, tanto de tipo comercial como libre.

- **MySQL**

Es un sistema de administración de base de datos relacional, licenciado bajo el GPL de la GNU, multi-hilo por lo que le permite soportar una gran carga de forma muy eficiente y además multi-usuario, con más de seis millones de instalaciones por todo el planeta. MySQL es propietaria y patrocinada por una sola firma, la compañía sueca MySQLAB que mantiene el derecho de la mayor parte del Código fuente del servidor SQL, así como también de la marca. Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL (ROBERTO, 2009).
Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. Entre las características disponibles en las últimas versiones se puede destacar:

  - Aprovecha la potencia de sistemas multi-procesador, gracias a su implementación multi-hilo.
  - Soporta gran cantidad de tipos de datos para las columnas.
  - Gran portabilidad entre sistemas.
  - Soporta hasta 32 índices por tabla.
  - Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

- **DB4O**

  Es un novedoso motor de base de datos orientada a objetos. Sus siglas se corresponden con la expresión "DataBase 4 (for) Objects", que a su vez es el nombre de la compañía que lo desarrolla. Las claves innovadoras de este producto son su alto rendimiento (sobre todo en modo embebido) y el modelo de desarrollo que proporciona a las aplicaciones para su capa de acceso a datos, el cual propugna un abandono completo del paradigma relacional de las bases de datos tradicionales. De este modo, tenemos las siguientes consecuencias directas resultantes de este nuevo paradigma:

- Deja de existir un lenguaje SQL de consultas/modificaciones para pasar a crearse sistemas de consulta por métodos delegados y actualización/creación/borrado automático de entidades mediante código compilable.
- Se elimina la necesidad de representar el modelo de datos de la aplicación en dos tipos de esquemas: modelo de objetos y modelo relacional. Ahora el esquema de datos del dominio viene representado por la implementación que se realice del diagrama de clases.

La mayor clave del éxito que está teniendo este motor de base de datos frente a otros competidores que han desarrollado tecnologías similares, es que se ha optado por un modelo de licenciamiento idéntico al utilizado por empresas como MySQL: licencia dual GPL/comercial. Es decir, si se quiere desarrollar software libre con esta librería, su uso no conlleva ningún coste por licencia; sin embargo si se desea aplicar a un software privativo, se aplica otro modelo de licenciamiento concreto.

Actualmente este producto funciona como una librería para dos tipos de plataformas de desarrollo: Java y.NET.

- Nativa a Java y .NET.
- 100% orientada a objetos, sin mapeo objeto-relacional.
- Diseñada para uso embebido.
- De código abierto y libre bajo la GPL (QUINTANA, 2009).

Con los elementos expuestos anteriormente en cuanto a metodologías, tecnologías, lenguajes de programación, herramientas y gestores de bases de datos, se seleccionaron para la posterior confección del sistema de apoyo al proceso de relatoría en reuniones las más adecuadas a petición de ALBET.S.A.

- Como metodología de desarrollo de software RUP, pues el equipo de desarrollo cuenta con cierta experiencia en el uso de dicha metodología. Además, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.
- UML como lenguaje de modelado, que facilitará a los desarrolladores representar visualmente el sistema a construir. Se caracteriza por poseer una notación estándar y simétrica esencial para el modelado de un sistema orientado a objetos.

- Visual Paradigm como herramienta de modelado visual que soporta el ciclo de vida completo del desarrollo de software. Está orientada a la creación de diseños y utiliza el paradigma de programación orientada a objetos.
- .Net como tecnología de desarrollo de software a petición de ALBET, por contar con todas sus aplicaciones desarrolladas en dicha tecnología, es multi-lenguaje para la construcción, distribución y ejecución de aplicaciones. Es una plataforma diseñada para simplificar el desarrollo de aplicaciones.
- C# como lenguaje de programación diseñado por Microsoft para su plataforma .NET. Lenguaje de última generación, que simplifica la programación y minimiza la cantidad de errores.
- Microsoft Visual Studio Express como entorno de desarrollo, por ser la versión gratuita del entorno de programación creado por Microsoft para desarrollar aplicaciones. Es compatible con todos los lenguajes de programación desarrollados por Microsoft.
- DB4O como base de datos embebida, orientada a objetos, nativa a Java y .NET, de código abierto y libre.

1.9 Conclusiones

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la compresión de esta investigación. Se analizaron los sistemas existentes para los procesos de relatoría en reuniones y se determinó la necesidad de desarrollar una aplicación que satisfaga las necesidades de ALBET.S.A. Se estudiaron metodologías, tecnologías, lenguajes de programación, herramientas y gestores de bases de datos para la futura implementación del sistema. Se eligieron las más convenientes para desarrollar un software con características similares a los ya existes.

Capítulo 2: Características del sistema

2.1. Introducción

Un detallado modelado del negocio es la base para el éxito de un producto de software. Permite comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema, se comprenden los problemas actuales de la organización y se identifican las mejoras potenciales. Asegura además que los consumidores, usuarios finales y desarrolladores posean un entendimiento común de la organización. Es uno de los flujos de trabajo que tiene mayor peso durante la fase de inicio en el desarrollo de software.

En el presente capítulo se define una visión general del proyecto y su alcance en iteraciones mediante el modelado del negocio. Se hace un análisis de las características del sistema a desarrollar por la situación problémica existente. Se detallan las necesidades de los usuarios, describiéndose las funcionalidades que serán objeto de automatización. Se especifican los requerimientos funcionales y no funcionales que regirán el desarrollo del sistema.

2.2 Situación actual del proceso de relatoría en reuniones de la empresa.

Los trabajadores de ALBET.S.A se reúnen periódicamente con fines laborales, comerciales y de interés para dicha institución. Cada reunión constituye un hecho relevante, pues en ellas se asignan las tareas y responsabilidades que debe desempeñar cada rol. La citación a las reuniones es enviada a través del correo electrónico por quien la presidirá, el secretario o responsable de esta según el tipo de reunión que son tres: Consejo de dirección, Comité de contratación y aprobación de divisas (CAD) o Comité de negocios.

La activista de acta, una vez comenzada la reunión, transcribe todo lo expresado, lo acordado y programado en dicho encuentro. Al concluir esta, la secretaria conforma el acta con un modelo o plantilla aprobado para este tipo de documento. Los acuerdos, además de dejarse plasmados en el acta se recogen en un documento Microsoft Excel para su control y gestión. Cuando el acta se aprueba por la persona que preside la reunión y es firmada por esta, se circulan los acuerdos tomados a través del correo electrónico para recordar a los responsables su cumplimiento. Las actas se archivan en formato duro y posteriormente se transcriben a formato digital. ALBET.S.A. cuenta con una serie de sistemas informáticos que ayudan al control del trabajo en la misma, pero carece de un sistema automatizado con la capacidad para procesar, almacenar y recuperar (de manera ágil, confiable y oportuna) la información necesaria para soportar las operaciones y toma de decisiones de negocios. Por esta razón se propone construir dicha aplicación.

2.3 Objeto de automatización

Durante el ciclo de desarrollo existen varios procesos que deben ser automatizados, puesto que su ejecución sin la ayuda de un sistema informático resulta más engorrosa y propensa a errores, además, con la aplicación propuesta se le ahorra tiempo a la persona encargada de redactar el acta, haciéndole más ágil y eficiente la confección y gestión de las mismas. Se desea desarrollar un sistema que tiene como propósito agilizar y facilitar el proceso de relatoría en reuniones de la empresa ALBET S.A. y la integración de este con un sistema de gestión de proyecto utilizado por la empresa.

2.4 Propuesta del sistema

En el presente trabajo se propone la implementación de una aplicación de escritorio que pone a disposición del usuario un grupo de funcionalidades que facilitarán la gestión de actas. El mismo estará programado en C# usando como plataforma de desarrollo el Microsoft Visual Studio 2008 Express y DB4O para la base de datos embebida con que contará el sistema. Teniendo en cuenta los resultados arrojados de investigaciones desarrolladas sobre los diversos sistemas existentes para el apoyo al proceso de relatoría, se determinaron las principales funcionalidades y características que poseerá la aplicación. El software debe permitir la autenticación de usuarios por el dominio para el control de acceso. Debe de ser capaz de:

- Posibilitar una vía rápida e intuitiva para la persona que toma nota en la reunión, de forma que le sea cómodo almacenar los planteamientos de los asistentes.
- Integrarse con el RedMine, herramienta de gestión que se utiliza en la Empresa. Dicha integración debe ser en dos cuestiones principales: leer los usuarios definidos y sincronizar los acuerdos.
- Almacenar la información localmente, mientras se desarrolla la reunión, puesto que es probable que en el local donde se desarrolla la misma, no exista conectividad. Al finalizar la reunión la persona encargada podrá conectar la herramienta a la red y sincronizar los acuerdos con la herramienta RedMine, estos serán accesibles por la web, para los trabajadores de la Empresa.
- Debe ser capaz de generar el acta de la reunión, basándose en una plantilla configurada. Además, debe permitir extraer los datos de una reunión específica, para generar su acta; esta podrá ser generada en formato Microsoft Word.

2.5 Definición del dominio.

No se cuenta con la información suficiente para establecer procesos de negocios, se hará la descripción a través del modelo de dominio. Este constituye un modelo conceptual que muestra gráficamente los conceptos (clases de objetos), los atributos y las asociaciones más importantes que se manejan en el entorno donde estará el sistema. Los objetos o conceptos incluidos en este modelo no describen clases u objetos del software, sino que representan clases conceptuales del mundo real que están asociadas al problema en cuestión, las cuales son importantes para entender lo que se representa, es decir, podrán ser utilizadas como una base de las abstracciones relevantes en el proceso de construcción del sistema.
Esto proporcionará a los usuarios, clientes, desarrolladores y otros interesados, la ventaja de utilizar un vocabulario claro y relacionado con el sistema en cuestión y facilitará la captura de los requerimientos para darle solución al problema.

2.5.1 Definición de las entidades y los conceptos principales.

A continuación se brinda una descripción de los principales conceptos del modelo de dominio para un mejor entendimiento del contexto que se está describiendo:

- **Reunión:** Concurrencia concertada y temporal de un conjunto de personas que tienen los mismos intereses o aficiones y se reúnen en un momento y espacio dado para discutir, aclarar o plantear algunos acuerdos. Las reuniones pueden ser ordinarias, que son aquellas que corresponden con la planificación o pueden ser extraordinarias, que son las que se citan para analizar temas que no estén en el plan temático u otros que hayan quedado pendientes de reuniones anteriores, en estas no se chequean los acuerdos. Existen 3 tipos de reunión:
- **Consejo de dirección:** Se reúnen todos los miembros del Consejo de Dirección y se debaten temas estratégicos y organizativos relativos a la empresa y a su funcionamiento. Se aprueban informes, documentos y se toman decisiones.
- **Comité de contratación y aprobación de divisas (CAD):** Se aprueban las contrataciones y pagos a ejecutar en divisas, este comité está formado por trabajadores y miembros del Consejo de Dirección de la entidad.
- **Comité de negocios:** Se realizan análisis vinculando la producción con la realización de negocios relativos a la empresa.
- **Plan temático:** Es la guía de temas a tratar, por cada una de las reuniones ordinarias que se citen, se usa para el Consejo de Dirección.

- **Acta:** Constituye una certificación o testimonio escrito en la cual se da cuenta de lo sucedido, tratado o pactado en oportunidad de cualquier circunstancia que requiera o exija de la correspondiente certificación legal de que ocurrió.
- **Acuerdo:** Es un compromiso en común por dos o más personas, por una junta, asamblea o tribunal. También se denomina así a un pacto o tratado por organizaciones, instituciones, empresas, entre otros.
- **Plantilla:** Esquema predefinido. Es un medio o un instrumento que permite guiar, portar o construir un diseño.
- **Persona:** Individuo de la especie humana, con derechos y obligaciones.
- **Participante:** Personas que asisten algún acto o lugar que debía participar.
- **Invitado:** Comunicar a alguien que asista o participe en una celebración, acontecimiento o junta.
- **Ausente:** Persona que no está presente donde debía estar.

2.5.1 Representación del modelo de dominio.

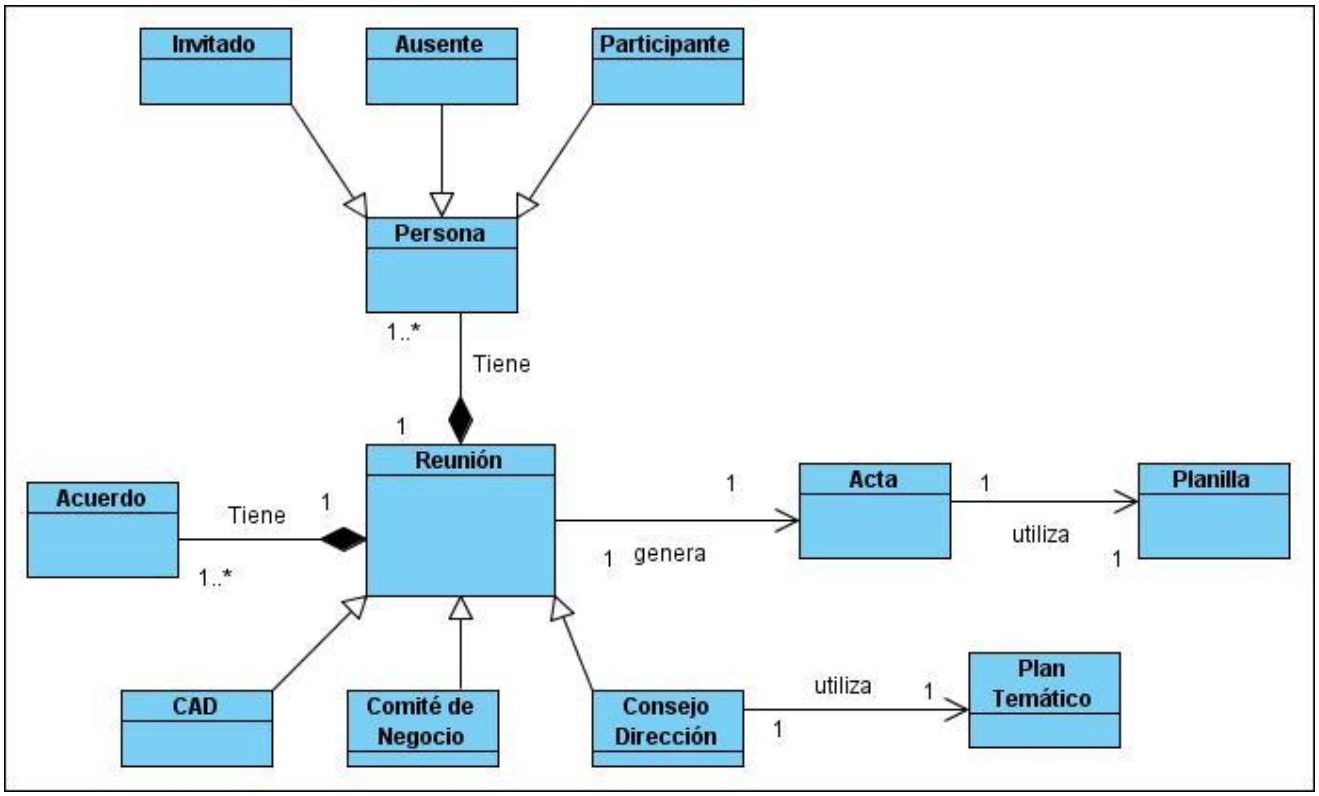

Figura 1: Modelo de dominio.

2.6 Captura e identificación de los requerimientos del sistema.

Este proceso cumple un esencial papel en el proceso de desarrollo de un producto de software, se orienta a la definición de lo que se desea producir, describiendo con claridad, sin imprecisiones, en forma estable y compacta. Se define el comportamiento que deberá tener el sistema, minimizando la posibilidad de errores que puedan ocurrir relacionados al desarrollo del mismo, pues se tiene especificado de forma clara lo que el cliente desea. Los requerimientos por lo general se dividen en dos grupos: requerimientos funcionales (capacidades o condiciones que el sistema debe cumplir) y requerimientos no funcionales (son propiedades o cualidades que el producto debe tener), y estos deben ser posibles de probar o verificar.

2.6.1 Requerimientos funcionales

Los requisitos funcionales, desde el punto de vista de las necesidades del cliente, son aquellas capacidades o condiciones que el sistema debe cumplir y que están fuertemente ligadas a las opciones del programa. Ellos deben de ser comprensibles por los clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en forma medible y verificable. Una vez conocido los conceptos que encierran al objeto de estudio, se puede analizar qué debe hacer el sistema para que se cumplan los objetivos planteados. Estos se enumeran a través de requerimientos funcionales, las prestaciones que el sistema será capaz de brindar. De acuerdo a los objetivos planteados, el sistema debe ser capaz de:

RF1: Autenticar usuarios en el sistema por el dominio.

RF2: Gestionar reunión.

- RF2.1 Crear una reunión.
- RF2.2: Facilitar la redacción de la minuta de la reunión a través del uso de combinaciones de teclas para insertar frases y nombres de participantes.
- RF2.3: Tomar la fecha y la hora inicio-fin de la reunión automáticamente de la PC o manualmente.
- RF2.4: Eliminar reunión.
- RF2.5: Modificar reunión.
- RF2.6: Filtrar reuniones por tipos de reunión.

RF3: Gestionar tipos de reunión.

- RF3.1: Crear tipos de reunión.
- RF3.2: Eliminar tipos de reunión.

RF4: Gestionar participantes por tipo de reunión.

RF4.1: Crear participantes por tipo de reunión.

RF4.2: Modificar participantes por tipo de reunión.

RF4.3: Mostrar participantes por tipo de reunión.

RF5: Gestionar invitados a la reunión.

RF5.1: Crear invitados a la reunión.

RF5.2: Eliminar invitados a la reunión.

RF6: Gestionar acuerdos.

RF6.1: Crear acuerdos por reunión.

RF6.2: Obtener acuerdos con incumplimientos dada una fecha que se seleccione a través de un calendario.

RF6.3: Mostrar acuerdos por reunión.

RF6.4: Revisar estado de los acuerdos de una reunión.

RF6.5: Modificar acuerdos de una reunión.

RF6.6: Eliminar acuerdos de una reunión.

RF6.7: Buscar acuerdos por reunión, por fecha y por estado.

RF7: Consultar plan temático.

RF7.1: Actualizar plan temático.

RF7.1.2: Mostrar actividades del plan temático de acuerdo al avance de los acuerdos tomados en reuniones pasadas.

RF8: Gestionar actas.

RF8.1: Generar el acta de una reunión en formato Microsoft Word.

RF8.2: Permitir adjuntar anexos.

RF8.3: Eliminar actas.

RF8.4: Buscar actas por tipo reunión asociada.

RF8.5: Mostrar acta de reunión.

RF9: Integrar con herramienta de gestión de tareas RedMine.

RF9.1: Obtener los usuarios definidos en el RedMine.

RF9.2: Mantener los usuarios sincronizados.

RF10: Subir acuerdos al RedMine.

2.6.2 Requerimientos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable,

rápido o confiable. Son restricciones en los servicios o funciones ofrecidas por el sistema, de mucha importancia para que clientes y usuarios puedan valorar las características no funcionales del producto. Normalmente, sirven como complemento a los requisitos funcionales que debe cumplir el sistema, es decir, una vez que se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Con el propósito de responder a las necesidades de la empresa, se definió un conjunto de propiedades o cualidades que debe cumplir la aplicación, las cuales se describen a continuación:

Usabilidad

Estos requerimientos describen los niveles apropiados de usabilidad según los usuarios finales del producto, por tanto, teniendo en cuenta que el sistema será usado por personas con conocimientos informáticos y asumiendo que sus niveles de experiencia, sean acordes con la tarea a desempeñar, se considera que:

- El sistema proporciona un acceso fácil y rápido al usuario, con el objetivo de proveer familiarización con el uso de la misma.
- Un usuario con pequeños conocimientos de informática debe ser capaz de aprender a utilizar la herramienta en poco tiempo.

Apariencia e interfaz externa

- El sistema deberá mostrar una interfaz sencilla, cómoda, amigable, intuitiva y de fácil navegación para cualquier tipo de usuario.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.
- La interfaz debe ser agradable para el usuario, que combine correctamente los colores, tipo de letra y tamaño, que los iconos estén en correspondencia con lo que representan no debe contener muchas imágenes que demoren las respuestas al usuario o lo distraigan de su trabajo.
- Debe cumplir con las pautas definidas en el Manual de Identidad de la Empresa

Rendimiento

- Los tiempos de respuesta y velocidad no deben de ser mayores de cinco segundos para las actualizaciones y las recuperaciones.
- En el proceso de relatoría, el sistema debe responder instantáneamente a las acciones del operador.

Seguridad

Dado que es uno de los requerimientos que puede provocar la mayor cantidad de riesgos, en la aplicación se hace necesario manejar de forma adecuada la seguridad, tratando de forma simultánea la confidencialidad, la integridad y la disponibilidad de los datos que serán manejados por los distintos usuarios y que pudiera afectar la integridad de los datos. Para lograrlo:

- Los usuarios se autenticarán con el dominio para poder tener acceso a la aplicación.
- Prevenir posibles fallos y recuperarse ante ellos totalmente.

Portabilidad

- El sistema podrá ejecutarse en entornos basados en Windows, sin dependencias a bases de datos o sistemas propietarios, previendo una futura migración a software libre.
- No debe requerir un proceso de instalación en las PCs donde se utilice.

Soporte

- Se requiere que el producto reciba mantenimiento y configuración ante cualquier fallo que ocurra durante el período de prueba.
- El sistema debe ser diseñado e implantado de manera que permita extensiones, modificaciones y un mejoramiento progresivo de sus funcionalidades.

Políticos-Culturales

- El producto no debe contener palabras en otros idiomas distintos al español.
- El producto debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.
- El sistema debe tener una interfaz que esté acorde con el lugar donde se implantará, es decir, que refleje los ideales de la organización y que sea amena, cumpliendo con las pautas de diseño.

Legales

- Reconocido y autorizado por instancias superiores tales como la dirección de producción de la UCI (facultad 1).
- Documentación legal de uso como Declaración de Autoría.
- Debe cumplir con la política de Gestión Documental de la Empresa.

Software

- Deberá implementarse sobre la plataforma .NET.

- Debe ser capaz de trabajar sin conexión en el momento de la reunión.

Hardware

- Para garantizar el correcto funcionamiento de la aplicación se necesitan como requerimientos mínimos una computadora con un procesador Pentium II o superior, una memoria RAM de 256 MB ó más y un Disco Duro de 10 GB ó más.

Confiabilidad

- Deben establecerse los mecanismos necesarios para el restablecimiento del sistema ante fallos de comunicación u otros.
- Garantiza un control estricto sobre el tráfico de información. La información debe de ser transmitida sobre canales seguros.
- Chequeo constante de la integridad y consistencia en los datos.
- Deberá prevenir los posibles fallos y/o errores que pudieran presentarse y posibilitar una rápida recuperación en dichos casos.

Restricciones en el diseño y en la implementación

- Utilizar los estándares de diseño establecidos.
- El sistema debe ser capaz de conectarse con PostgreSQL por ser la base de datos utilizada por el RedMine.

2.7 Modelación del sistema.

De acuerdo a lo explicado anteriormente en la descripción y conceptualización del sistema que se propone, se definen las principales funcionalidades de este en casos de uso (CUS), que no son más que las descripciones narrativas en lenguaje natural de los procesos del dominio, definiendo verdaderamente lo que se quiere.

Siendo así se definieron como CUS:

CUS 1 Autenticar usuarios en el sistema por el dominio.
CUS 2 Sincronizar usuarios con el RedMine.
CUS 3 Gestionar reunión.
CUS 4 Gestionar tipos de reunión.
CUS 5 Gestionar participantes por tipo de reunión.
CUS 6 Gestionar invitados.
CUS 7 Crear combinaciones de teclas.
CUS 8 Tomar notas.

CUS 9 Filtrar por tipos de reuniones.

CUS 10 Gestionar acuerdos.

CUS 11 Subir acuerdos al RedMine.

CUS 12 Consultar plan temático.

CUS 13 Gestionar actas.

CUS 14 Subir anexos al acta de una reunión.

2.7.1 Actor del sistema.

El actor del sistema no es otro sino un tercero fuera del sistema que interactúa con él, para este caso se define:

Personas relacionadas con el sistema	Justificación
Secretario de acta	Es la persona que interactúa con el sistema. Se encarga del manejo de toda la información y realiza todas las funcionalidades dentro de la aplicación.

Tabla 2: Actor del sistema.

2.7.2 Diagrama CUS.

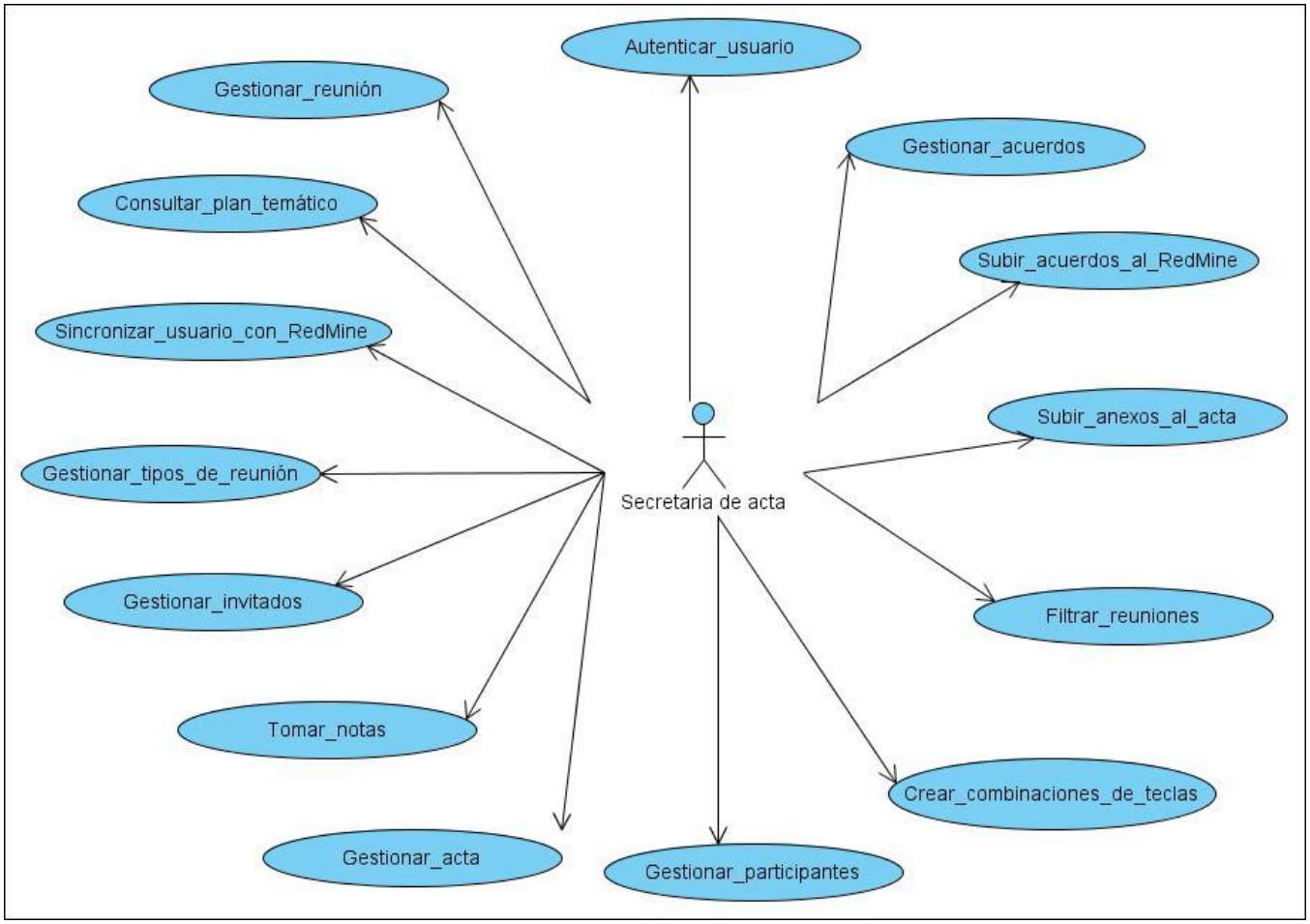

Figura 2: Diagrama de caso de uso del sistema.

2.7.3 Descripción de CUS.

Descripción del caso de uso	
Nombre del caso de uso	Gestionar reunión.
Objetivo	Permitir al secretario de acta gestionar las reuniones que se realizan en la empresa (crearlas, modificarlas, mostrarlas, eliminarlas).
Actores	Secretario de acta (inicia)
Resumen	El CUS se inicia cuando secretario de acta accede al sistema para gestionar las reuniones de la empresa y finaliza con la adición, modificación o eliminación de una reunión.
Precondiciones	Para poder gestionar reuniones el secretario de acta debe estar autenticado

	en el sistema.
Poscondiciones	Los datos de la reunión se guardan en el sistema una vez que se crea por primera vez, en caso de ser modificada conserva los cambios respectivos, si es eliminada desaparecen de la base de datos y se actualiza el listado de estas que se muestra en el sistema.
Referencias	RF2
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1. El secretario de acta se dirige al menú Reuniones y selecciona la opción Gestión de Reuniones.	2. El sistema muestra la interfaz Gestión de Reuniones.
Sección “Nueva reunión”	
3. El secretario de acta selecciona la opción Nueva. 3.2 El secretario de acta introduce los datos y selecciona la opción Salvar.	3.1 Muestra en la interfaz los campos para la nueva reunión, la fecha de inicio y fin de la reunión, el lugar. 3.3 Guarda los datos insertados.
Sección “Modificar reunión”	
4. El secretario de acta selecciona la reunión que desea modificar de la lista de Reuniones. 4.2 El secretario de acta introduce los nuevos datos de la reunión y selecciona la opción Salvar.	4.1 El sistema muestra los datos de la reunión que se desea modificar dando la opción de hacer cambios en los mismos. 4.3 Guarda las modificaciones realizadas.
Sección “Eliminar reunión”	
5. El secretario de acta selecciona la reunión que desea eliminar de la lista de reuniones y selecciona la opción Eliminar.	7.1 El sistema elimina la reunión.
Prioridad	Crítico.

Tabla 3: Descripción de caso de uso gestionar reunión.

El resto de las descripciones de los casos de uso se encuentran en los Anexos.

2.8 Conclusiones.

En el desarrollo del capítulo se detallaron los procesos actuales de ALBET.S.A, obteniéndose la propuesta de solución de un sistema que deberá con su implementación resolver los problemas existentes en los procesos de relatoría en reuniones. En los procesos de negocios no se determinaron límites bien establecidos, por lo que se realizó un modelo de dominio, definiéndose los principales conceptos para un mejor entendimiento del negocio. Se fundamentó el objetivo general del mismo,

desarrollándose la propuesta de solución a través de los requerimientos que debe tener el sistema, creando los casos de uso necesarios para satisfacer los mismos y por ende una breve descripción de estos. Al culminar este análisis se puede dar comienzo al desarrollo del sistema propuesto.

Capítulo 3: Diseño del sistema

3.1 Introducción

En el presente capítulo se desarrolla el modelado de un conjunto de artefactos vinculados a la construcción de la aplicación para una mejor comprensión de la solución que se propone. El diseño pone de relieve la solución lógica: como el sistema cumple con los requerimientos. De manera que se encuentran reflejados los diagramas de clases y de secuencias referentes al diseño de cada uno de los casos de usos. Se describe la arquitectura a utilizar, se presentan las pautas seguidas en el diseño y el modelo de clases persistentes. Para la realización del mismo se utilizó como punto de partida las especificaciones de los casos de uso del sistema y los requisitos asociados a estos.

3.2 Definición de la arquitectura

Estilo arquitectónico

Según la recomendación de la IEEE-1471 la arquitectura de software se entenderá como: la organización fundamental de un sistema encarnada en sus componentes, las relaciones de los componentes con cada uno de los otros y con el entorno, y los principios que orientan su diseño y evolución (IEEE, 2000).

Dentro de la amplia clasificación de estilos arquitectónicos utilizaremos arquitectura en capas debido a que organiza el modelo de diseño.

Arquitectura en capas:

Las arquitecturas en capas son muy utilizadas para el desarrollo de aplicaciones en la actualidad por las grandes ventajas que proporcionan. El principal objetivo que persigue es reducir dependencias entre artefactos, situándolos en capas lógicas, donde cada capa depende del servicio prestado por la inferior y presta un servicio a la superior, proporcionando a los desarrolladores ventajas en cuanto al mantenimiento y reutilización de componentes o artefactos. El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Además, el estilo admite optimizaciones, refinamientos y proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la

posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

Cada capa se ocupa de un nivel del problema y debe tener poco acoplamiento con las demás de manera que el cambio en una capa, no altera en gran medida los cambios en la otra capa. Si se desea en un futuro incorporar una capa de presentación distinta, no debe alterar a la capa operacional ya desarrollada, es decir, cambiar su implementación, debe introducir los mínimos efectos en el resto de la aplicación.

La arquitectura a utilizar constará de las siguientes capas:

Capa de presentación:

Esta es la capa encargada de la interacción con los diferentes tipos de usuarios, se encarga de modelar la forma en que se mostrarán y se recogerán los datos entrados por los usuarios, así como de la apariencia visual que tendrá la aplicación. Se comunica con la capa de lógica de negocio a la cual envía todas las solicitudes de los usuarios y recibe la respuesta después de que hayan sido procesadas cada una de estas solicitudes. En el sistema a desarrollar esta capa se llamará Acuerdos.

Capa de lógica de negocio:

La capa de lógica representa las clases controladoras del negocio. Es la encargada de modelar la lógica de negocio que dará solución a cada uno de los casos de usos de la aplicación, es en esta donde se establecen las reglas o restricciones que debe cumplir la aplicación. Se comunica con la capa de presentación para recibir las solicitudes de los usuarios y enviar las respuestas después del procesamiento. En el sistema a desarrollar esta capa se llamará Negocio.

Capa de acceso a datos:

En el sistema que se desarrollará esta capa estará dividida en dos subcapas: una subcapa llamada Local que representará la base de datos embebida de la aplicación donde estará almacenada la información del sistema y será la encargada del manejo de datos persistentes dentro del mismo; la otra subcapa llamada Remoto representará la conexión de la aplicación con el servidor de base de datos del RedMine como encargado de la sincronización de datos del sistema con la herramienta de gestión

de tareas. Esta capa en su conjunto se comunicará con la capa de lógica de negocio desde donde recibe las solicitudes de almacenar o recuperar información.

Capa entidad:

En esta capa aparecerán las clases entidades, las cuales definen algún tipo de información perdurable del sistema. Es una capa horizontal que puede ser utilizada por cualquier otra capa anterior.

3.3 Patrones

Los GRASP son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (ARIZACA, 2008).

- **Creador:** Permite decidir cuáles serán las clases creadoras de otras clases. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. Este patrón se utiliza en las clases entidades, cuando estas se instancian y crean instancias de clases contenidas en las mismas, ejemplo: Reunion.cs.
- **Alta Cohesión:** Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Una clase cohesionada facilita el cambio. Al realizar un cambio en una clase muy cohesionada, todos los métodos que pueden verse afectados, toda la información que necesitamos controlar, estará a la vista, en el mismo fichero. Se pone de manifiesto en la implementación de las clases controladoras como: PersonaCtrl.cs.
- **Bajo Acoplamiento:** Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Debe haber pocas dependencias entre las clases. Uno de los principios para proteger al software frente al cambio es mantener bajo el acoplamiento entre clases, cuanto menor sea el acoplamiento entre clases, menor influencia tendrán los cambios. Se aplica en una herencia no profunda entre las clases entidades, (Participante.cs -> Persona.cs), además en la separación de capas lógicas.

3.3.1 Patrones de diseño

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, Christopher Alexander considera que: Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma (HELM, 2006).

Los patrones de diseño pretenden:

Proporcionar catálogos de elementos reusables en el diseño de sistemas software.

- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Patrones de creación

- **Singleton:** Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. Dicho patrón se utiliza en la capa de acceso a datos Local, para inicializar una sola vez la base de datos antes de leer los datos: Local.PersonaCtrl.cs, Local.ReunionCtrl.cs

Patrones estructurales

- **Composite:** Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos. Ejemplo: la clase Acuerdo.cs contiene una referencia a Persona.cs.

Patrones de comportamiento

- **Iterator:** Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna. Se evidencia a través de las colecciones de Items de los ListBox.

3.4 Modelo de diseño

El diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos, traduce los requerimientos funcionales y no funcionales en una representación del software, desarrollando así la arquitectura y constituyendo el primer paso para el desarrollo de cualquier sistema.

Un modelo de diseño es una abstracción del modelo de implementación y su código fuente, el cual se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. Representa a los casos de uso en el dominio de la solución (DÉBORA, 2009).

Para el modelado del diseño se realizó un diagrama de clases por cada caso de uso y un diagrama de secuencia por cada escenario del caso de uso.

3.4.1 Diagrama de clases del diseño.

En el diagrama de clases del diseño se muestran los atributos y métodos de cada clase y se representa de una forma sencilla la colaboración y las responsabilidades de las distintas clases que forman el sistema.

Los diagramas de clases se utilizan para modelar la vista del diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados: los diagramas de componentes y los diagramas de despliegue. Estos son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

A continuación se muestra un diagrama de clases del diseño, el resto se encuentra en los anexos.

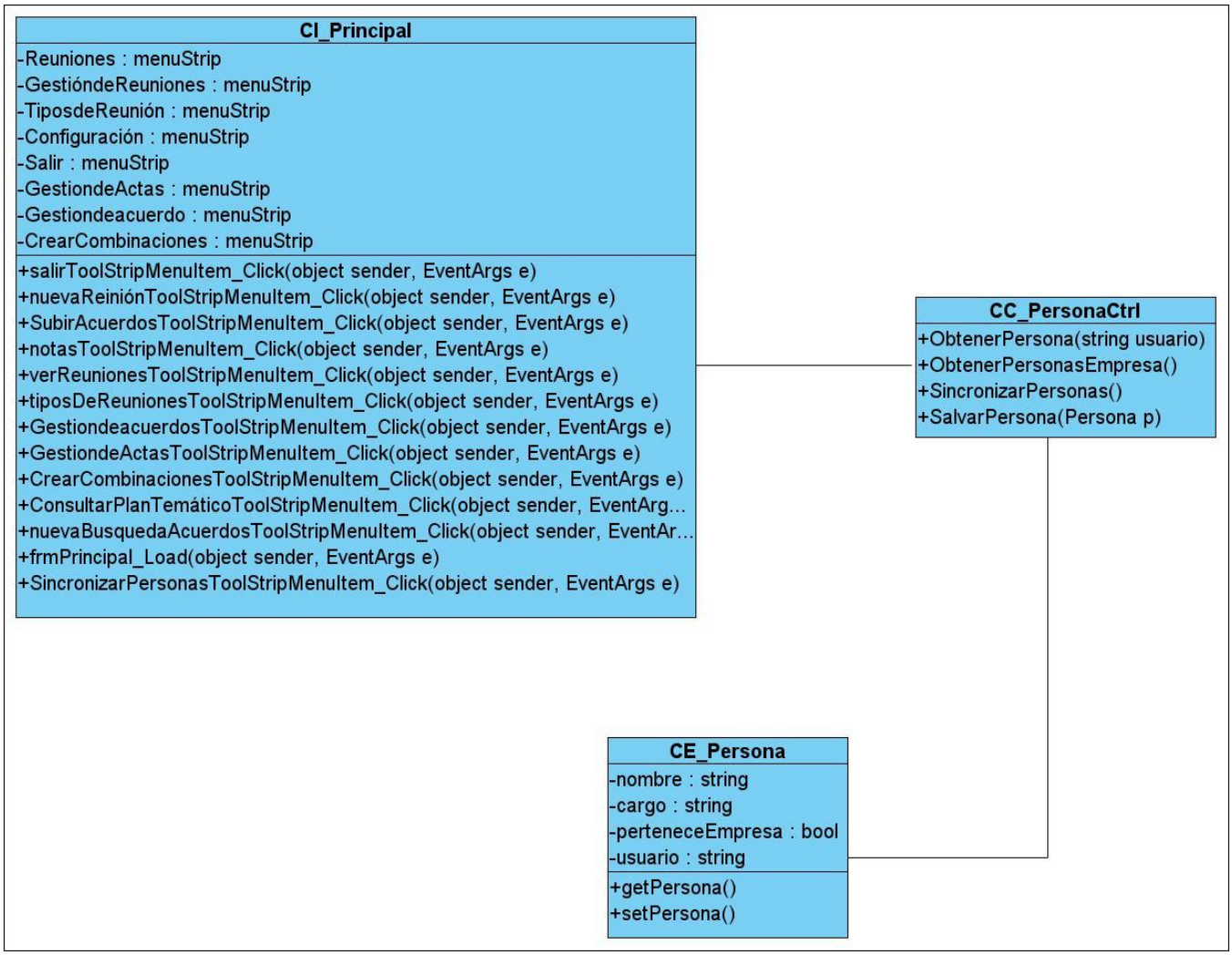

Figura 3: DC_Sincronizar_usuario_con_RedMine.

3.4.2 Diagramas de interacción

Los diagramas de interacción son utilizados para la modelación de los aspectos dinámicos de un sistema, por lo que se modelan instancias concretas, componentes y nodos juntos con los mensajes enviados entre ellos, que representan su interacción. Los diagramas de interacción tienen dos formas de manifestarse:

- Diagramas de secuencia.
- Diagramas de colaboración.

Un diagrama de secuencia destaca la ordenación temporal de los mensajes y el diagrama de colaboración a su vez destaca la organización estructural de los objetos que envían y reciben mensajes.

A continuación se presenta un diagrama de secuencia del diseño, el resto se encuentra en los anexos:

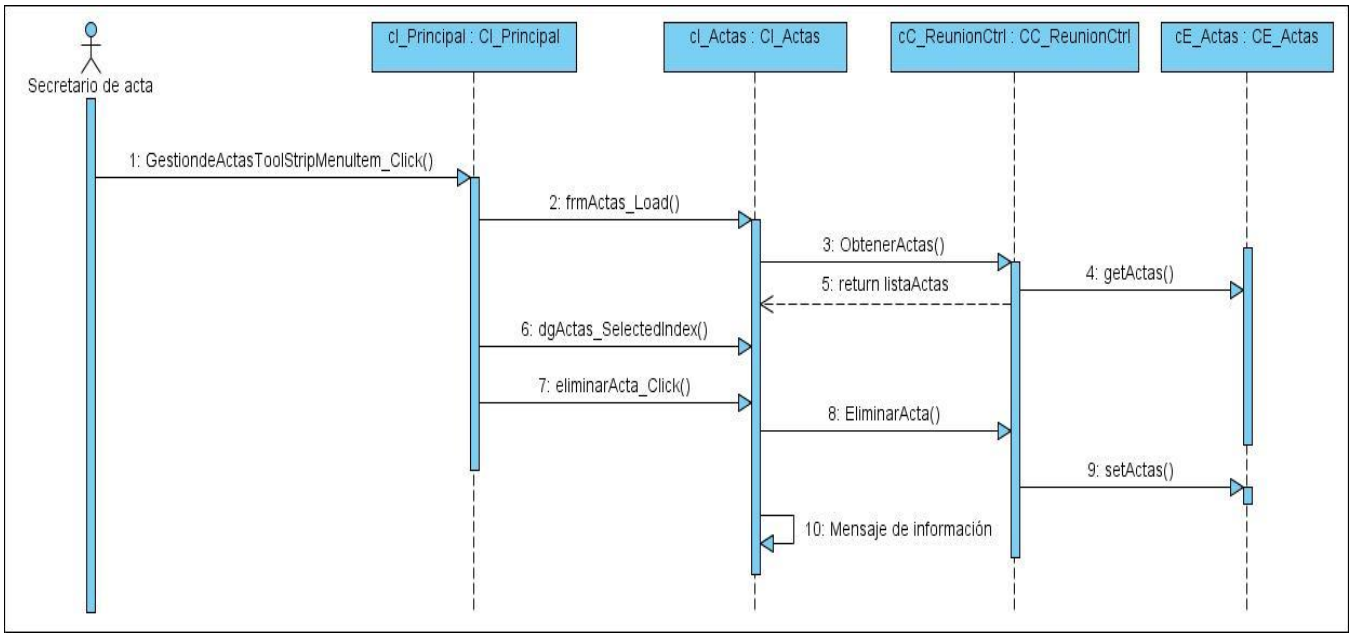

Figura 4: DS_Gestionar_actas (Eliminar).

3.5 Diseño de la base de datos.

Para el desarrollo del Sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A., se utilizará una base de datos orientada a objetos (DB4O) como se detalla en epígrafes anteriores. La misma no cuenta con un modelo de datos como las aplicaciones que utilizan bases de datos relacionales, pues se utiliza en estas un modelo de objetos.

El diseño de la base de datos está dado por su diagrama de clases persistentes. El modelo diseñado representa una base de datos OO (Orientada a Objetos), manteniendo información acerca de configuraciones del sistema.

A diferencia del modelado Entidad-Relación de las bases de datos relacionales, el diagrama de una base de datos OO no necesita pasar por el modelo conceptual, lógico y físico, lo que representa una ventaja a la hora de la mantención del modelo, puesto que significa menor esfuerzo.

A continuación, se presenta el modelo de objetos:

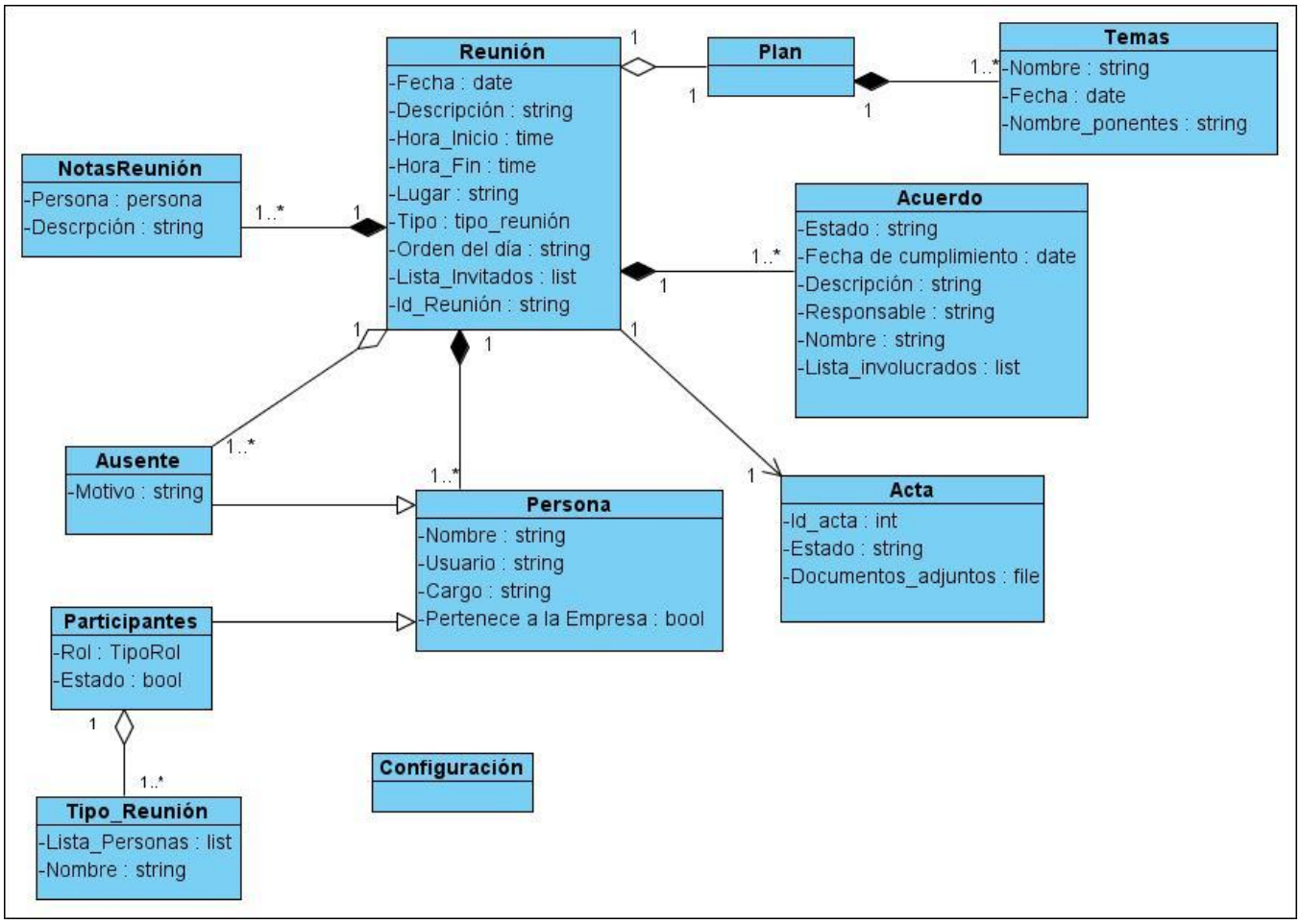

Figura 5: Modelo de Objetos.

En toda base de datos es muy importante la integridad de los datos, garantizando la corrección y completitud de estos. Al realizar diversas operaciones sobre los datos, como es el caso de una actualización, un insertado o un borrado, se podrían estar modificando datos existentes por valores incorrectos. DB4O tiene una forma especial de tratar la integridad de estos; la misma se implementa en la configuración de la base de datos antes de abrir la conexión. Uno de los aspectos fundamentales de la integridad consiste en evitar la duplicidad de objetos, es decir, que un mismo objeto no esté almacenado más de una vez, en este motor de base de datos se evita la duplicidad de objetos mediante la indexación de los campos identificadores de las clases persistentes.

3.6 Descripción de las clases persistentes

Al utilizarse un motor de base de datos como DB4O, el diseño de la base de datos consiste, como se detalla anteriormente, en las clases persistentes del sistema. Al ser una base de datos Orientada a Objetos, dejan de existir las tablas y relaciones entre ellas. A continuación en lugar de presentarse la

descripción de las tablas, se documentan las clases persistentes, con una breve explicación sobre cada uno de los atributos que poseen, logrando un mejor entendimiento del modelo de objetos.

Nombre: CP_Reunión		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Fecha	date	Almacena la fecha de la reunión.
Descripción	String	Almacena la descripción de la reunión.
Hora-Inicio	time	Almacena la hora que comienza la reunión.
Hora -Fin	time	Almacena la hora que finaliza la reunión.
Lista_notas	list	Almacena todas las notas tomadas en la reunión.
Lista_acuerdos	list	Almacena todos los acuerdos de la reunión.
Lugar	String	Almacena el lugar donde se realiza la reunión.
Tipo	Tipo-Reunión	Almacena el tipo de reunión.
Orden del día	String	Almacena el orden del día de la reunión.
Acta	Acta	Almacena el acta de la reunión.
Lista_ausentes	list	Almacena todos los ausentes de la reunión.
Lista_invitados	list	Almacena todos los invitados de la reunión.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 4: Descripción de la CP_Reunión.

Nombre: CP_Acuerdo		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Estado	String	Almacena el estado del acuerdo.
Id del acta	int	Almacena el identificador del acta.
Fecha de cumplimiento	date	Almacena la fecha en que se debe cumplir el acuerdo.
Descripción	String	Almacena la descripción del acuerdo.
Responsable	String	Almacena el responsable del acuerdo.
Nombre	String	Almacena el nombre del acuerdo.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 5: Descripción de la CP_Acuerdo.

Nombre: CP_Persona		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Nombre	String	Almacena el nombre de la persona.
Usuario	String	Almacena el usuario de la persona.
Cargo	String	Almacena el cargo de la persona.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 6: Descripción de la CP_Persona.

Nombre: CP_Acta		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Id del acta	int	Almacena el identificador del acta.
Estado	String	Almacena el estado del acta.
Documentos adjuntos	file	Almacena los documentos adjuntos del acta.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 7: Descripción de la CP_Acta.

Nombre: CP_Tipo_reunión		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Lista_Personas	list	Almacena las personas de la empresa que pertenecen a un tipo de reunión.
Nombre	String	Almacena el nombre del tipo de reunión.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 8: Descripción de la CP_Tipo_reunión.

Nombre: CP_Nota		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Persona	Persona	Almacena una persona de la empresa que pertenece a un tipo de reunión.

Descripción	String	Almacena la descripción de lo que habla una persona en la reunión.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 9: Descripción de la CP_Nota.

Nombre: CP_Ausente		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Motivo	String	Almacena el motivo por el cual se ausentó la persona a la reunión.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 10: Descripción de la CP_Ausente.

Nombre: CP_Plan		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Temas	Temas	Almacena el tema.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 11: Descripción de la CP_Plan.

Nombre: CP_Temas		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**

Nombre	String	Almacena el nombre del tema.
Fecha	date	Almacena la fecha del tema.
Nombre Ponentes	String	Almacena el nombre del ponente del tema.
Para cada responsabilidad: En esta clase solo se definieron métodos de acceso, es decir, operaciones que solamente se limitan a devolver o cambiar atributos. Estos tipos de operaciones son conocidas como propiedades.		

Tabla 12: Descripción de la CP_Temas.

Nombre: CP_Confifuración		
Tipo de clase: Clase Entidad		
Atributo	**Tipo**	**Descripción**
Para cada responsabilidad:		

Tabla 13: Descripción de la CP_Configuración.

3.7 Conclusiones

En este capítulo se realizó un modelo detallado de la solución propuesta, a través del modelado de los diagramas de clases del diseño. Además, se modelaron los diagramas de interacción por cada realización de caso de uso del sistema. Se diseñó también el modelo de clases persistentes que satisfacen las necesidades de la base de datos que requiere la implementación de la aplicación.

Capítulo 4: Implementación y validación

4.1 Introducción

El flujo de trabajo implementación se comienza con el resultado del diseño, por lo que se realiza la implementación de las clases obtenidas en el diseño del sistema, con el objetivo de realizar la construcción del software en términos de componentes, ficheros de código fuente, ejecutables, entre otros.

En el presente capítulo se desarrolla el diagrama de componentes y el diagrama de despliegue conformando lo que se conoce como modelo de implementación, al describir los componentes, construir su organización y dependencia entre los nodos físicos en que funciona el sistema. Se realizan los casos de pruebas de caja negra con el objetivo de validar el correcto funcionamiento de la aplicación.

4.2 Diagrama de componentes

Los Diagramas de componentes ilustran las piezas del software que conformarán un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, un componente eventualmente puede comprender una gran porción de un sistema (PTY, 2009).

El diagrama de componentes describe como los elementos del modelo de diseño se implementan en términos de componentes. Además, como se organizan de acuerdo con los mecanismos de estructuración y modulación disponibles en el entorno de implementación y lenguaje de programación utilizado (OSCAR, 2009).

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. En el mismo, se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Dicho diagrama define cómo las clases, artefactos y otros elementos de bajo nivel, se unen para formar componentes de alto nivel y las conexiones entre ellos. Los componentes son artefactos de software compilados que trabajan acoplados, para brindar el comportamiento requerido dentro de las restricciones definidas en el proceso de captura de requisitos.

A continuación se muestra el diagrama de componentes de la aplicación de software en cuestión:

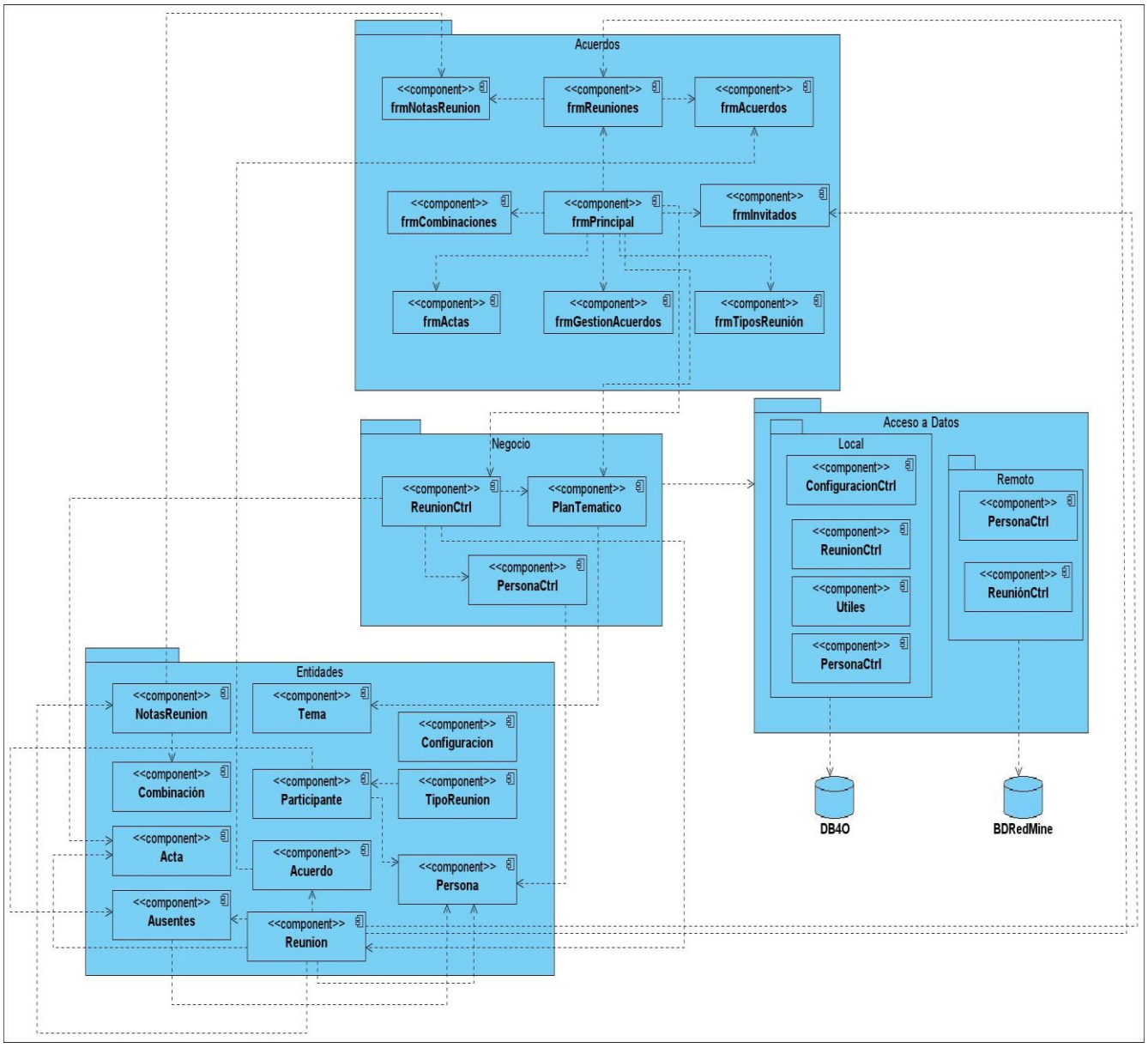

Figura 6: Diagrama de componentes.

4.3 Diagrama de despliegue.

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones que son las relaciones entre los nodos y componentes (PTY, 2009).

Un diagrama de despliegue muestra la disposición física de los recursos de ejecución computacional. Durante la ejecución, los nodos pueden contener componentes y objetos. La asignación de componentes y de objetos a los nodos puede ser estática o pueden migrar entre ellos.

Los diagramas de despliegue muestran la configuración en funcionamiento del sistema, incluyendo su hardware y su software. Estos permiten describir la plataforma sobre la que se ejecuta el sistema al representar la topología de procesadores y dispositivos, lo que facilita la definición y manejo de la frontera entre el hardware y el software. La distribución física del sistema se expresa en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo y dónde el sistema será puesto en funcionamiento. Las estaciones de trabajo, dispositivos y procesadores son reflejados como nodos y su estructura interna puede ser representada adicionando otros nodos o artefactos. A continuación se muestra el diagrama de despliegue de la aplicación de software en cuestión:

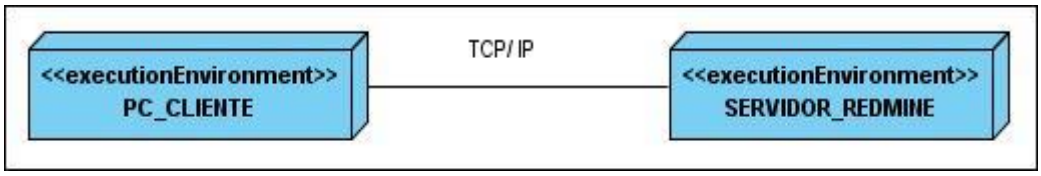

Figura 7: Diagrama de despliegue.

4.4 Modelo de prueba.

Las pruebas del software son un elemento crítico para la garantía de calidad del producto y representan una revisión final de las especificaciones del diseño y de la codificación. Demuestran hasta qué punto las funciones del software cumplen con los requisitos funcionales. Además los datos recogidos en el desarrollo de la prueba proporcionan un buen indicador de la fiabilidad del software y de alguna manera, la calidad del mismo, aunque no asegura la ausencia de defectos si puede demostrar la existencia de errores(PRESSMAN, 2002).

Las pruebas son una actividad en la cual un sistema o componente es utilizado bajo condiciones o requerimientos especificados, los resultados son observados, registrados y se hace una evaluación de algún aspecto del sistema o componente.

Las pruebas pueden clasificarse siguiendo varios criterios, uno de estos es según los métodos que se utilicen. Existen dos métodos de pruebas para probar cualquier producto de software, ellos son: el método de caja blanca y el método de caja negra.

Las pruebas de caja blanca del software se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado (PRESSMAN, 2002).

Las pruebas de caja negra se refieren a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de pruebas pretenden demostrar que las funciones del producto son operativas, que la entrada se realiza de forma adecuada, que se produce un resultado correcto y que la integridad de la información externa se mantiene. Las pruebas de caja negra examinan algunos aspectos del modelo fundamental del sistema sin valorar demasiado la estructura lógica interna del software (PRESSMAN, 2002).

Se centran principalmente en los requisitos funcionales del software. Estas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, agrupándose en los requisitos funcionales del sistema y ejercitándolos. Muchos autores consideran que estas pruebas permiten encontrar:

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

Siendo estas las usadas para la realización de las pruebas al Sistema de apoyo al proceso de relatoría en reuniones de ALBET.S.A.

4.4.1 Casos de pruebas

A continuación se muestran los casos de pruebas de algunos de los casos de uso, más significativos del sistema, como se define en la plantilla Diseño de caso de pruebas funcionales basados en CU.doc publicada en la sección de Producción, en el Expediente de proyecto versión 2.0 (040_expediente-proyecto v2.02.zip), del sitio de calidad de la universidad (disponible en http://calisoft.uci.cu/). Dicha plantilla está basada en la estrategia seguida para el Programa de Mejoras de los procesos basado en el modelo CMMI (Capability Madurity Model Integration) que se lleva a cabo en la universidad para alcanzar el nivel dos del mismo.

Para confeccionar los casos de prueba de caja negra existen distintas técnicas. Algunas de ellas son:

- Particiones de equivalencia.
- Análisis de valores límites.
- Métodos basados en grafos.
- Pruebas de comparación.

En las pruebas realizadas al sistema se utilizó la técnica de Partición de equivalencia.

La partición equivalente es una técnica de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (PRESSMAN, 2002).

- **Caso de prueba para el caso de uso "Gestionar reunión".**

Descripción del caso de uso: El CUS se inicia cuando secretario de acta accede al sistema para gestionar las reuniones de la empresa y tomar la fecha, la hora inicio y la hora de fin de la reunión. El CUS finaliza con la adición, modificación o eliminación de una reunión.

Condiciones de ejecución: Que el secretario de acta se haya autentificado para poder tener acceso a la interfaz Gestión de reuniones y se haya insertado una reunión al sistema.

Secciones a probar en el caso de uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Nueva reunión	EC 1.1: Adicionar una nueva reunión introduciendo datos válidos.	Se registran los datos y se adiciona la reunión presionando el botón Salvar. Se presiona el botón Aceptar del mensaje de confirmación.
	EC 1.2: Adicionar una nueva reunión dejando campos requeridos en blanco.	Se introducen los datos dejando campos requeridos en blanco y se presiona el botón Salvar. Se muestra un mensaje indicando que faltan datos obligatorios por llenar.

	EC 1.3: Adicionar una nueva reunión introduciendo errores en los datos.	Se introducen datos erróneos y se presiona el botón Salvar. Se muestra un mensaje que indica que se han introducido datos inválidos.
	EC 1.4: Cerrar	Introduciendo o no datos, se cancela la operación y se cierra la ventana.
SC 2: Modificar reunión	EC 2.1: Modificar una reunión introduciendo datos válidos.	Se selecciona la reunión a modificar, se cambian los datos necesarios y se modifica la reunión presionando el botón Salvar. Se presiona el botón Aceptar del mensaje de confirmación.
	EC 2.2: Modificar una reunión dejando campos requeridos en blanco.	Se selecciona la reunión a modificar, se cambian los datos necesarios, dejando campos requeridos en blanco y se presiona el botón Salvar. Se muestra un mensaje indicando que faltan datos obligatorios por llenar.
	EC 2.3: Modificar una reunión introduciendo errores en los datos.	Se selecciona la reunión a modificar, se cambian los datos necesarios introduciendo errores y se presiona el botón Salvar. Se muestra un mensaje que indica que se han introducido datos inválidos.
	EC 2.4: Cerrar	Modificando o no, los datos mostrados se cancelan la operación y se cierra la ventana.
SC 3: Eliminar reunión	EC 3.1: Eliminar	Se selecciona la reunión a eliminar y se presiona el botón Eliminar. Se presiona el botón Aceptar.

Tabla 14: Secciones a probar en el caso de uso Gestionar reunión.

.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

[1]	Tipo de reunión	Lista desplegable	No	Es una combinación de letras que no puede dejarse vacío.
[2]	Lugar	Texto	No	Es una combinación de letras que no puede dejarse vacío.
[3]	Hora de inicio-Fecha	Fecha	No	Es una combinación de caracteres y números que deben cumplir el siguiente formato: dd/mm/aaaa hh:mm.
[4]	Hora de fin-Fecha	Fecha	No	Es una combinación de caracteres y números que deben cumplir el siguiente
[5]	Presentes	Selección múltiple	No	Es una combinación de letras, donde al menos se debe seleccionar un presente.
[6]	Añadir invitado	Texto	Si	Es una combinación de letras que puede dejarse en blanco.
[7]	Cargo	Texto	Si	Es una combinación de letras que puede dejarse en blanco.

Tabla 15: Descripción de las variables.

El resto de los casos de pruebas y cada escenario de estos se encuentran en Anexos.

4.5 Estudio de la factibilidad

La realización de un análisis de los costos y beneficios del proyecto es imprescindible a la hora de asumir una tarea, pues se realiza con el objetivo de demostrar la viabilidad o factibilidad del desarrollo del sistema propuesto, planteándose los beneficios tangibles e intangibles que reportaría la aplicación.
A partir de la información obtenida como resultado de la fase de inicio se tienen los conocimientos necesarios para tener una idea de lo que debe hacer el producto. Por lo tanto, se puede definir qué es lo que realmente hace falta en cuanto a recursos humanos y materiales, lo que permite hacer una estimación del tiempo de duración del proyecto a través del Análisis por puntos de casos de uso.

4.5.1 Análisis por puntos de casos de uso

Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, contabilizar el tiempo total

estimado para el proyecto a partir de esos factores. Existe una posibilidad de predecir el tamaño de un sistema a partir de las características de sus requisitos, expresados en los casos de uso.

Cálculo de puntos de casos de uso sin ajustar

Para lograr una buena estimación mediante este método se debe calcular, como primer paso, los puntos de casos de uso sin ajustar. Este valor se calcula a partir de la siguiente ecuación:

UUCP = UAW + UUCW

Donde se tiene que**:**

UUCP: Puntos de casos de uso sin ajustar

UAW: Factor de peso de los actores sin ajustar

UUCW: Factor de peso de los casos de uso sin ajustar

Factor de peso de los actores sin ajustar

Este valor se calcula mediante un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. Los criterios a tener en cuenta se detallan a continuación.

Descripción de los tipos de actores y sus respectivos factores de peso.

Tipo de actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación.	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

Tabla 16: Descripción de los tipos de actores y sus respectivos factores de peso.

Factor de peso de los actores del sistema

Actor del Sistema	Tipo de Actor	Factor de Peso
Secretario de acta	Complejo	3

Tabla 17: Factor de peso de los actores del sistema.

En la presente investigación existe un solo actor en el sistema, el cual constituye una persona que interactúa con el sistema mediante una interfaz gráfica, por lo que su factor de peso sería 3. Por tanto:

UAW = Cantidad de actores * Factor de peso

UAW = 1 * 3

UAW = 3

Factor de peso de los casos de uso sin ajustar

Después de haber obtenido el valor del factor de peso de los actores del sistema se calcula el factor de peso de los casos de uso sin ajustar (UUCW). Este valor se calcula mediante un análisis de la cantidad de casos de usos presentes en el sistema y la complejidad de cada uno de ellos.

Descripción de los tipos de casos de uso y sus respectivos factores de pesos

Tipo de Caso de Uso	Descripción	Factor de peso
Simple	El caso de Uso contiene de 1 a 3 transacciones.	5
Medio	El caso de Uso contiene de 4 a 7 transacciones.	10
Complejo	El caso de Uso contiene más de 8 transacciones.	15

Tabla 18: Descripción de los tipos de casos de uso y sus respectivos factores de pesos.

Factor de peso de los casos de uso del sistema

Caso de Uso	Tipo de Caso de Uso	Factor de Peso
Autenticar usuario	Simple	5
Gestionar reunión	Complejo	15
Gestionar acuerdos	Complejo	15
Consultar plan temático	Medio	10
Gestionar acta	Complejo	15
Subir acuerdos al RedMine	Medio	10
Sincronizar usuarios con el RedMine	Medio	10

Gestionar tipos de reunión	Medio	10
Gestionar participantes por tipos de reunión	Medio	10
Gestionar invitados	Medio	10
Tomar notas	Medio	10
Subir anexos	Medio	10
Filtrar reuniones	Medio	10
Crear combinaciones de teclas	Medio	10

Tabla 19: Factor de peso de los casos de uso del sistema.

UUCW=∑ (cant cu * peso)

Simples: 1*5 = 5

Medios: 10*10 = 100

Complejos: 3*15 = 45

UUCW = 5 + 100 + 45 = 150

Finalmente, los puntos de casos de uso sin ajustar sería:

UUCP = UAW + UUCW

UUCP = 3 + 150

UUCP = 153

Cálculo de puntos de casos de uso ajustados

Se necesita ajustar el valor de los puntos de casos de uso y para ello se emplea la siguiente ecuación:

UCP = UUCP x TCF x EF donde:

UCP: Puntos de casos de uso ajustados

UUCP: Puntos de casos de uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

Para el cálculo del **factor de complejidad técnica (TCF)** se tiene en cuenta la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5 donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la siguiente tabla se muestra el peso de cada uno de estos factores acompañado de una breve descripción:

Factor de complejidad técnica

Factor	Descripción	Peso	Valor Asignado	Pesoi * Valor asignadoi
T1	Sistema Distribuido	2	5	10
T2	Objetivos de *performance* o tiempo de respuesta	1	3	3
T3	Eficiencia del usuario final	1	4	4
T4	Procesamiento interno complejo	1	2	2
T5	El código debe ser reutilizable	1	5	5
T6	Facilidad de instalación	0.5	4	2
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	1	1
T11	Incluye objetivos especiales de seguridad	1	4	4
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento	1	3	3

Tabla 20: Factor de complejidad técnica.

TCF = 0.6 + 0.01 * Σ (Pesoi x Valor asignadoi)

TCF = 0.6 + 0.01 * 50.5

TCF = 0.6 + 0.505

TCF = 1.105

Para el cálculo del **factor ambiente** se tienen en cuenta las habilidades y el entrenamiento del grupo involucrado en el desarrollo del sistema, los cuales constituyen otros factores que tienen gran impacto en las estimaciones de tiempo. El cálculo del mismo es similar al cálculo del factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5. En la siguiente tabla se asignan los valores correspondientes al sistema:

Factor de ambiente

Factor	Descripción	Peso	Valor asignado	Pesoi * Valor asignadoi
E1	Familiaridad con el modelo de proyecto utilizado	1.5	4	6
E2	Experiencia en la aplicación	0.5	5	2.5
E3	Experiencia en orientación a objetos	1	5	5
E4	Capacidad del analista líder	0.5	3	1.5
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	5	10
E7	Personal *part-time*	-1	5	-5
E8	Dificultad del lenguaje de programación	-1	3	-3

Tabla 21: Factor de ambiente.

EF = 1.4 - 0.03 * Σ (Pesoi * Valor asignadoi)

EF = 1.4 - 0.03 * 22

EF = 1.4 – 0.66

EF = 0.74

Finalmente, calculando los puntos de caso de uso ajustados quedaría:

UCP = UUCP * TCF * EF

UCP = 153 * 1.105 * 0.74

UCP = 169.065 * 0.74

UCP = 125.1081

Estimación del esfuerzo a través de los puntos de casos de uso

Está dado por la siguiente ecuación:

E = UCP *CF donde:

E: Esfuerzo estimado en horas hombres.
UCP: Punto de casos de usos ajustados.
CF: Factor de conversión.
Para calcular el **Factor de conversión** se tiene en cuenta:

- Se contabilizan cuántos factores de los que afectan al factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

- Se contabilizan cuántos factores de los que afectan al factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

No existe ningún valor menor que 3 entre los factores E1 y E6 y uno mayor que 3 entre los factores E7 y E8.
Total EF = 0+1
Total EF = 1
Como el total contabilizado es 1 se utiliza el **factor de conversión 20 horas-hombre/punto de casos de uso.** Entonces:
E = UCP * CF
E = 125.1081* 20
E = 2502.162 horas-hombre

Este método contempla solamente el desarrollo de la funcionalidad especificada en los casos de uso para proporcionar una estimación del esfuerzo obtenida por los puntos de casos de uso.

Para finalmente obtener una estimación más completa de la duración total del proyecto hay que agregar las estimaciones del resto de las actividades relacionadas con el desarrollo de software. Con ese objetivo se toma en cuenta un criterio reconocido como aceptable estadísticamente. Es planteada la distribución del esfuerzo entre las diferentes actividades de un proyecto. Utilizando estos datos junto con los cálculos realizados anteriormente se calcula el resto de las estimaciones y se obtiene la estimación total del proyecto:

Distribución del esfuerzo entre las actividades del proyecto

Actividad	Porcentaje	Esfuerzo
Modelado	10.00 %	250.2162
Diseño	20.00%	500.4324
Implementación	40.00 %	1000.8648
Pruebas	15.00 %	375.3243
Sobrecarga(otras actividades)	15.00%	375.3243
Total	100.00 %	2502.162

Tabla 22: Distribución del esfuerzo entre las actividades del proyecto.

El proyecto requiere de 2502.162 horas-hombre para su desarrollo. Suponiendo que se trabaje 8 horas diarias, se obtiene aproximadamente 312.77 días para el desarrollo del proyecto y suponiendo que las dos personas que conforman el equipo de desarrollo trabajen 30 días al mes, el proyecto tiene una duración de 5.21 meses aproximadamente.

Esto quiere decir que dos personas pueden realizar el sistema propuesto en más o menos cinco meses y medio (5.21 meses).

4.5.2 Beneficios tangibles e intangibles

Para la empresa ALBET.S.A, la aplicación que se propone ofrece beneficios a tener en cuenta para su utilización en los procesos de relatoría en reuniones. Dicho sistema brinda diversas mejoras, entre las que se encuentran el seguimiento de los acuerdos planteados en cada reunión y un control más eficiente de los mismos, actualizando la herramienta de gestión de tareas (RedMine) con dichos datos una vez concluido el evento. El software posibilita una mejor redacción de las actas a través de la casi exacta transcripción de lo que se dice o se plantea; resultando fácil e intuitiva la utilización del mismo a las personas involucradas en el proceso. Proporciona factibilidad en la gestión de las reuniones, permitiendo clasificarlas según el personal que debe participar en las mismas y ofrece un mejor control

de los asistentes e invitados. Por las razones antes mencionadas el sistema de apoyo al proceso de relatoría en reuniones viabiliza la gestión de los documentos generados en las reuniones de la empresa.

4.5.3 Análisis de costos y beneficios

El desarrollo del sistema no supone un gasto considerable debido al uso, en su mayoría, de herramientas gratis. Los gastos por concepto de tecnologías son mínimos, pues ya la Universidad cuenta con una infraestructura productiva, que favorece la realización de un proyecto con estas características. Debido a que el sistema brinda una interfaz intuitiva y amigable para los usuarios, los posibles gastos serían casi nulos. Por lo antes planteado se concluye que es factible desarrollar un sistema que reporte estos beneficios.

4.5 Conclusiones

En este capítulo se trató el modelo de implementación como artefacto de más importancia en el flujo de trabajo actual, teniendo en cuenta las bases creadas por el flujo de trabajo de diseño anteriormente desarrollado. Se describieron en detalle las pruebas que se le realizaron al sistema y como evidencia de ello, se tuvo en cuenta esencialmente los casos de pruebas de caja negra comprobándose que no existen errores en las funciones operativas del software.

Conclusiones generales

El trabajo manual y la carencia de un sistema informatizado en los procesos de relatoría en reuniones en la empresa ALBET.S.A, fue el comienzo en la presente investigación. La acertada descripción de los procesos actuales en dicha empresa, permitió obtener una mejor comprensión del problema e identificó las principales necesidades a resolver. La posterior definición de los requerimientos fue la iniciación del proceso de desarrollo del sistema propuesto; al lograr un entendimiento efectivo con el cliente sobre las funcionalidades que la aplicación debe brindar. Como resultado de las etapas de diseño e implementación desarrolladas, se ha concebido una aplicación de escritorio y la integración de la misma con la herramienta de gestión de tareas RedMine, utilizada por la Empresa. Se validó mediante un análisis realizado a través de las pruebas, arrojando resultados que permitieron el perfeccionamiento de la solución, contribuyendo a mejorar la funcionalidad requerida por el cliente. Dicho sistema facilita la gestión de las actas, la gestión de acuerdos y la toma de notas durante el evento; su valor fundamental se formula en la contribución a simplificar el trabajo y por ende el tiempo que origina el procesamiento manual de la información. Con el desarrollo del presente trabajo, se alcanzó satisfactoriamente, el objetivo propuesto: desarrollar un sistema de apoyo al proceso de relatoría en reuniones de ALBET, S.A. e integrarlo con el sistema de gestión de tareas de la empresa.

Recomendaciones

A partir de las experiencias obtenidas en el desarrollo del trabajo y con vistas de lograr un aprovechamiento óptimo del resultado alcanzado se recomienda:

- Continuar el desarrollo de la herramienta con el objetivo de extender el propósito de la misma con otras funcionalidades de interés, que se desean para versiones superiores, adecuándolo a la demanda creciente de exigencias de los usuarios.
- Proponer, tras corroborar un desempeño exitoso, la utilización y generalización de este sistema en otras áreas de la Universidad de las Ciencias Informáticas, en otras empresas o entidades para un mejor control de los acuerdos tomados en las reuniones.
- Implementar la ayuda de la aplicación con el objetivo de informarle al usuario cómo trabajar con la aplicación desarrollada para un mejor entendimiento de sus funcionalidades.
- Integrarlo con ALFRESCO, que es la alternativa de código abierto para la gestión documental utilizada por la Universidad, esta integración será específicamente para el control de las actas generadas por la aplicación.

Referencia bibliográfica

1. ALFASA. Disponible en: http://www.alfasa.com/.
2. ARIZACA, E. Patrones. 2008.
3. CECILIA. Reunión.Características. 2009.
4. DÉBORA. Diseño. 2009.
5. ERICK. Las herramientas CASE. 2006.
6. FILIBERTO. Implementación de un componente de software para la visualización tridimensional de Neuroimágenes 2009.
7. FLORENCIA. Actas.Características. 2009.
8. HELM, G. Patrones de Diseño. 2006.
9. IEEE. Recommended Practice for Architectural Description of Software Intensive Systems. 2000.
10. JANI. Actas.Características. 2009.
11. JUAN. Los lenguajes de programación. 2007.
12. JULIO. Relatoría. Características.Guía para su elaboración. 2009.
13. L., C. S. Disponible en: http://www.cisicom.com.
14. LUIS, J. Herramientas CASE.Rational Rose. 2007.
15. MARTÍNEZ, Y. Módulo Citas del Sistema de Información Hospitalaria alas HIS 2009.
16. MORGADE, G. ClioBD. Sistema de control de versiones para bases de datos 2009.
17. OSCAR, F. Diagramas del Diseño. 2009.
18. PRESSMAN, R. S. Ingeniería del software, Un Enfoque Práctico, McGrawHill, quinta edición. 2002.
19. PTY, S. S. Diagramas de Componentes. 2009, Disponible en: http://www.sparxsystems.com.ar.
20. QUINTANA, A. Implementación del Módulo de Búsqueda del Sistema de Monitoreo y Análisis de Noticias 2009.
21. ROBERTO. MySQL. 2009.
22. SULLON, M. Metodologías de desarrollo. 2006.

Bibliografía

1. Disponible en: http://www.corbinball.com/.
2. Disponible en: http://office.microsoft.com.
3. URIBE, R**.** Disponible en: http://www.elraul.com/
4. Disponible en: http://www.freedownloadmanager.org
5. Disponible en: http://www.diariodecuyo.com.
6. Disponible en: http://documentospoliciales.blogspot.com.
7. Disponible en: http://www.mundodescargas.com.
8. Disponible en: http://grupos.emagister.com.
9. Disponible en: http://www.definicionabc.com.
10. Disponible en: http://www.itbuilder.com.
11. Disponible en: http://www.chuidiang.com
12. Reynoso, Carlos and Kicillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [Online] Universidad de Buenos Aires, 2004. [Cited: febrero 22, 2009.] Disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#10.
13. García, Joaquín. UML: Casos de Uso.Desarrollo de software orientado a objetos. [Online] septiembre 27, 2003. Disponible en: http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.
14. Chaves, Michael Arias. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [Online] julio 7, 2006. Disponible en: http://www.intersedes.ucr.ac.cr/10-art_11.html.
15. Marqués, María Mercedes. Modelo de Datos. [Online] febrero 12, 2005-2006. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node32.html.
16. I. Jacobson, G. Booch, J. Rumbaugh. El Proceso Unificado de Desarrollo de software. s.l. : Addison-Wesley, 2000.
17. Lovelle, Juan Manuel Cueva. Introducción a UML. Universidad de Oviedo, España : s.n., 2003-2004. 34. Valencia, María Eugenia. Diagrama de Clases de Diseño. Escuela de Ingeniería de Sistemas y Computación. [Online] 2009. [Cited: enero 15, 2009.] Disponible en: http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/DISCLASES_A12.pdf.
18. Grau, Xavier Ferré. 2004. Tutorial UML, Desarrollo Orientado a Objetos con UML. [En línea] 2004. [Citado el: 23 de Enero de 2009.] Disponible en: http://www.clikear.com/manuales/uml/introduccion.aspx.

19. Mayoral, Antonio Gutiérrez. 2005. Características principales de C#. [En línea] 11 de Febrero de 2005. [Citado el: 23 de Enero de 2009.] Disponible en: http://gsyc.es/~agutierr/pfc-tecnica-html/node22.html.
20. Microsoft. 2008. Microsoft Visual Studio 2008 Standard Edition. [En línea] 2008. [Citado el: 2009 de Enero de 27.] Disponible en:http://www.microsoft.com/visualstudio/en-us/products/standard/default.mspx.
21. Oracle. Oracle Database 11g Standard Edition. [En línea] Oracle. [Citado el: 10 de Febrero de 2009.] Disponible en: http://www.oracle.com/database/
22. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. s.l.: PEARSON EDUCACIÓN S.A, 2000. [En línea] [Citado el: 12 de Diciembre de 2008] Disponible en: http://bibliodoc.uci.cu/pdf/reg00060.pdf

Glosario de términos

Aplicación: Programa con el que el usuario final interactúa a través de una interfaz, realizando tareas útiles para éste.

Arquitectura software: Conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Establece los fundamentos para que los desarrolladores trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema.

CASE: (Computer Aided Software Engineering). Bajo el término de Ingeniería de Software Asistida por Ordenador se incluyen una serie de herramientas, lenguajes y técnicas de programación que permiten la generación de aplicaciones de manera semiautomática. Las herramientas CASE liberan al programador de parte de su trabajo y aumentan la calidad del programa a la vez que disminuyen sus posibles errores.

Casos de uso: Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.

Cliente: Cualquier persona o elemento de un sistema de información que requiere un servicio.

Componente: Parte física y reemplazable del sistema que cumple y proporciona la realización de un conjunto de interfaces, ejemplo: ficheros de código fuente, ficheros de código binario, ejecutables y similares.

Despliegue: Cuando varios trabajos relativamente independientes (flujos de control, procesos) se distribuyen entre diferentes dispositivos hardware (procesadores).

Digitalizar: Acción de convertir en digital cualquier tipo de información, ya sea gráfica, de audio, vídeo, vídeo en movimiento, otros.

Embebida: Sumergida, metida adentro.

Gregario: Un animal o persona gregaria es la que sigue una tendencia a agruparse en manadas o colonias, en el caso de los animales, o en grupos sociales, en el caso de las personas.

Hardware: Conjunto de componentes materiales de un sistema informático. Cada una de las partes físicas que forman un ordenador, incluidos sus periféricos. Maquinaria y equipos (CPU, discos, cintas, cables, otros).

Integridad: Capacidad del sistema de información para garantizar, la excelencia en la calidad de todos los datos almacenados en el sistema y la garantía de que los datos e informaciones del sistema estén protegidos contra su eventual pérdida por la presencia de posibles contingencias.

Internet: Red de ordenadores mundial que permite comunicación y transferencia de datos, noticias y opiniones entre personas y usuarios conectadas a ella.

Microsoft: (Microsoft Corporation, Redmond, WA) Compañía de software más grande del mundo. Microsoft fue fundada en 1975 por Paul Allen y Bill Gates, dos estudiantes universitarios que escribieron el primer interprete BASIC para el microprocesador 8080 de Intel. Aunque también se conoce por sus lenguajes de programación y aplicaciones para computadores personales, el éxito sobresaliente de Microsoft se debe a sus sistemas operativos DOS y Windows.

Oracle: Empresa especializada en la fabricación de programas de bases de datos (en ordenadores).

Plataforma de desarrollo: Entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo, sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación.

Proceso de desarrollo: Definición del conjunto completo de tareas, actividades o acciones interrelacionadas entre sí, necesarias para transformar los requisitos de un usuario en un producto.

Producto: Artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables y documentación. Es cualquier cosa que puede ser ofrecida al mercado para su compra.

Redes de la comunicación: Intercomunicación entre ordenadores que permite no solo el intercambio de datos, sino también compartir recursos de todo tipo, optimizando así elevadas inversiones. Las redes son el soporte para estas conexiones y (aparte la diferenciación más genérica entre las redes públicas y privadas), según el objeto de definición, la tecnología es variada.

Servicios: es un conjunto de actividades que buscan responder a una o más necesidades de un cliente.

Sistema informático: Un sistema informático como todo sistema, es el conjunto de partes interrelacionadas, hardware, software y de recueros humanos (humanware). Un sistema informático típico emplea una computadora que usa dispositivos programables para capturar, almacenar y procesar datos.

Software: Los componentes intangibles de una computadora, es decir, el conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

SQL: (Structure query language) Lenguaje de preguntas estructurado, lenguaje que utiliza bases de datos para pedir información de las mismas.

TIC: Tecnologías de la Información y Comunicaciones.

.NET: Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.