UNIVERSISDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERÍA

DESARROLLO DEL PORTAL DEL CONTROL DE MIEMBROS Y ROTACIÓN AUTOMÁTICA DE CONTENIDOS VÍA WEB DE LA COMISIÓN MEXICANAN DE DERECHOS HUMANOS A.C.

TESIS PROFESIONAL
PARA OBTENER EL TITULO DE:

INGENIERÍA EN COMPUTACIÓN

PRESENTAN:

GUSTAVO SERRANO DIEZ
CONRADO SÁNCHEZ DEL VALLE

ASESOR DE TESIS:
ING. GABRIELA BETZABE LIZARRAGA RAMÍREZ

MEXICO,D.F. MAYO 2005

AGRADECIMIENTOS

A mi familia, maestros, amigos, compañeros, alumnos y toda la gente que me inspiró y ayudó en este trabajo. Todos ustedes son ángeles de Dios.

Conrado Sánchez del Valle

«¿Quién me dará reposar en ti, que vengas a mi corazón y lo embriagues hasta hacerme olvidar mis males y abrazarme a ti, mi único bien? ¿Qué eres tú para mí? Hazme la misericordia de que pueda decirlo. ¿Y quién soy yo para ti, pues me mandas que te ame; y si no lo hago te irritas contra mí y me amenazas con grandes miserias? ¡Pero, qué! ¿No es ya muchísima miseria simplemente el no amarte?» (San Agustín)

Hay tanto que agradecer que podría agotar libros enteros, y aún así me faltaría bastante; porque llevo una deuda con valor de eternidad a quien me ha librado de la muerte.

Dedico este trabajo, en primer lugar, a Cristo Rey, salvador, modelo, bien máximo, Verbo y Sabiduría del Padre; porque yo no lo buscaba y él me encontró. Reina, reina sobre nuestra patria y juventud, reina en la Universidad y en nuestra cultura, que al nombre de Jesús toda rodilla se doble.

¿Cómo hablar de Cristo sin mencionar a su Madre? ¿Cómo mencionar a la patria sin que me venga a la memoria el nombre de Guadalupe? Este trabajo también está dedicado a la Virgen de Guadalupe, Madre del verdadero Dios y patrona de Hispanoamérica, porque el alma de México, su identidad y su destino, son una mujer, que es Madre y Virgen a la vez, protectora de los pueblos de esta tierra.

También se la dedico a Juan Pablo II, modelo de juventud, de valor, justicia e idealismo; promotor de un nuevo humanismo, que contempla al hombre como persona, creado a *imagen y semejanza del Creador*; al hombre de Cracovia, que derribó el telón de acero, y que en mis horas de mayor desesperación siempre supo motivarme con el ejemplo de su vida: ¡No Tengas Miedo! *Glorifica Señor a tu siervo*.

Igualmente a todos los que desde la historia nos han dejado ejemplo de valor y esperanza, al cardenal Van Thuan (a quien conocí) a Anacleto González Flores (a quien no conocí), a G.K. Chesterton (a quien me hubiera gustado conocer), etc.

A aquellos con quienes comparto el tiempo de esta vida, a quienes les guardo un amor entrañable, gracias por su paciencia y su estima, ya saben que guardan un espacio privilegiado en mi alma. Los enumero como se me vayan ocurriendo, si me falta alguno no se ofenda, después le hago otra dedicatoria.

A mis padres, de quienes recibí el apoyo, la vida, el ejemplo y la fe. Al resto de mi familia, mis abuelas, primos y tíos, en especial Jorge, ¡se fiel hasta la muerte!

A todos los que nos asesoraron con la tesis, la Ing. Gabriela Lizárraga, el sr. Manuel Cota, Imelda Tamés, Pablo Hernández, etc.

A toda la banda, con quienes viví los mejores años de mi vida, con quienes dediqué mis mejores horas, codo con codo, para realizar un ideal, un proyecto, una meta: la Universidad. También tod@s l@s que tuvieron y tienen que aguantarme todavía.

Las chicas: Adelaida, Ana, Ana Karina, Claudia, Diana, Gabita, Gaby, Gloria, Judith, Karina, Lupita, Magda, Marcela, Rocío…

Los valedores: Beto, Carlos, César, Chucho, Estanislao, Gumer, Israel, Jacobo, Manuel, Marco, Miguelón, Pako, Poncho, Ramón, Rodolfo, Ruy, Tadeo… también el p. Toño, el p. Honorio, el p. Pepe Víctor, Carlos Caso-Rosendi, el p. Alvaro da Silva, Scott Hahn, etc.

Gustavo Serrano Diez (fru)

ÍNDICE

INTRODUCCIÓN

Este capítulo trata de dos cosas: a) la institución a la que se le brindó el servicio y b) el fundamento teórico del proyecto.

1. INTRODUCCIÓN

1.1. ÁMBITO DEL NEGOCIO

1.1.1. ¿Qué es la Comisión Mexicana de Derechos Humanos?

La Comisión Mexicana de Derechos Humanos A. C. (CMDH) es una organización de la sociedad civil (OSC) que busca generar y promover una Cultura de los Derechos Humanos en México y el mundo procurando, a través de su estudio, difusión y defensa, el conocimiento, reconocimiento, vigencia y respeto de los mismos, basados en la dignidad de la persona humana[1].

La CMDH la forman los asociados, que integran la Asamblea General de asociados, la cual «es el órgano supremo de la asociación y su autoridad no tendrá más límites que los señalados por su Declaración Escrita de Principios Fundamentales, estos estatutos y la ley de la materia»[2].

La Asociación para su operación cuenta con diversos órganos de toma de decisiones según se muestra en el organigrama:

ORGANIGRAMA DE LA CMDH

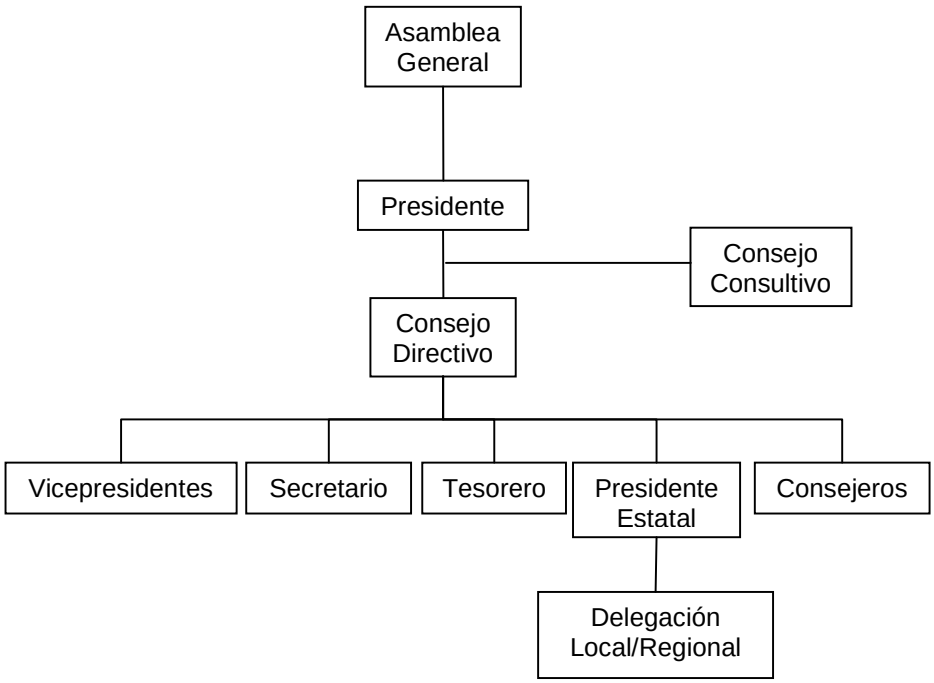

Fig. 1.1.1-1: Organigrama según estatutos de la CMDH

[1] Cf. Estatutos Sociales de la CMDH, Cláusula Segunda: Objeto Social.
[2] Cf. Estatutos Sociales…, Cláusula Décima Tercera: Supremacía.

La Asamblea General está formada por todos los miembros de la CMDH, mismos que cumplen con los siguientes requisitos:

1. Firmar en conformidad los estatutos y principios de la asociación.
2. Participar en las actividades de la asociación.
3. Pagar puntualmente las cuotas establecidas por la asociación[3].

La Asamblea se reúne al menos una vez cada año con los siguientes objetivos:

1. Aprobación del informe anual de actividades.
2. Aprobación de planes y programas que deseen someter a su consideración el Consejo Directivo y/o el Consejo Consultivo.
3. Elección o reelección, en su caso de los miembros del consejo Directivo de la Asociación.
4. Resolución de las inconformidades sobre exclusión presentadas por asociados.[4]

El presidente lleva la responsabilidad de presidir las Asambleas de la Asociación y las sesiones del Consejo Directivo; también es el ejecutor de sus resoluciones; en caso de empate tiene voto de calidad.[5]

El Consejo Consultivo, que está formado por todos los ex presidentes y por aquellas personas que designe la Asamblea a propuesta del Consejo Directivo.[6]

Su función es asesorar al presidente y al Consejo Directivo; son invitados permanentes a las sesiones del Consejo Directivo y tienen derecho a estar informados de las decisiones de dicho consejo.

El Consejo Directivo tiene, entre otras, las siguientes atribuciones:

1. Designar a los funcionarios del propio consejo (excepto al presidente).
2. Aprobar el reglamento interno de la Asociación.
3. Elaborar el Programa de Trabajo Anual.
4. Aprobar el presupuesto y vigilar su ejercicio.
5. Establecer los comités, comisiones y grupos de trabajo o de estudio especializado que estime necesarios o convenientes.
6. Resolver sobre la admisión, suspensión y/o exclusión de asociados.[7]

El Consejo Directivo se reúne al menos trimestralmente para llevar a cabo sus funciones, cuenta con el apoyo de personal contratado para necesidades específicas según sea el caso, como es en la actualidad el staff central, que apoya al presidente.

El Consejo Directivo está formado a su vez, por uno o más vicepresidentes, un tesorero, un secretario, los presidentes estatales y otros consejeros que el mismo consejo o presidente a nombre de éste considere necesario nombrar.

[3] Cf. Estatutos Sociales…, Cláusula Octava: Derechos y Obligaciones de los Asociados.
[4] Cf. Estatutos Sociales…; Cláusula Décima Sexta: Periodicidad y competencia de las Asambleas Ordinarias.
[5] Cf. Estatutos Sociales…; Cláusula Vigésima Cuarta: Funcionarios del Consejo Directivo.
[6] Cf. Estatutos Sociales…; Cláusula Vigésima Octava: El Consejo Consultivo.
[7] Cf. Estatutos Sociales…; Cláusula Vigésima Segunda: Facultades del Consejo Directivo.

Los presidentes estatales representan a las delegaciones estatales, que son las representaciones de la CMDH en el interior de la república mexicana; según estatutos, la asociación podrá establecer Delegaciones Locales o Regionales dentro y fuera de los Estados Unidos Mexicanos.[8]

Las principales actividades de la CMDH son:[9]

- Estudio de los derechos humanos.
- Difusión de los mismos.
- Defensa de los derechos humanos.

A partir del año 2003, se estableció mediante planeación estratégica que esto se logra mediante las líneas de acción.

Una línea de acción es una coyuntura que está involucrada directamente con los objetivos.

Entre otras, podemos enumerar las siguientes:

- Derecho a la Seguridad Pública -contra la corrupción e impunidad.
- Democracia y Observación Electoral.
- Derechos de la Mujer.
- Derechos de la Familia.
- Derecho a la Vida.
- Jóvenes por los Derechos Humanos.

Igualmente, la presidencia, a partir del año 2003 decidió que la mejor manera de alcanzar los objetivos era nombrar vocalías, una para cada línea de acción; por vocalía se entiende una línea de acción a la cual se le ha asignado un programa y un responsable, para lograr determinadas metas en un determinado tiempo.

El responsable de la vocalía se llama vocal, puede tener personal (contratado o voluntario) a su cargo y es miembro del consejo directivo.

1.1.2. NECESIDADES EXPRESADAS POR EL CLIENTE

Las necesidades expresadas por el cliente se engloban en tres áreas específicas:

1. El fortalecimiento de la cohesión interna.
2. Hacer eficiente la comunicación.
3. La creación de un sitio Web oficial.

1.1.2.1. La cohesión interna

La CMDH es una organización de la sociedad civil (OSC)[10], formada por voluntarios, distribuidos por todo el país formando representaciones locales o estatales[11], cuyos asociados presentan una pluralidad de pensamiento, ocupaciones y problemas particulares.

Por tanto, la cohesión entre los miembros es siempre un problema de vital importancia y su solución no es trivial. El conocimiento mutuo y el contacto que tengan los miembros unos con otros, en especial los del consejo, es primordial.

---

[8] Cf. Estatutos Sociales…; Cláusula Séptima: Delegaciones.
[9] Cf. Estatutos Sociales…, Cláusula Segunda: Objeto Social.
[10] Sección *1.1.1.Ámbito del Negocio: ¿Qué es la CMDH?* del presente documento.
[11] *Ibid.*

En pláticas con el cliente se sugirió tener un directorio de consejeros que sea accesible a todos los miembros de los mismos consejos[12] donde se tengan los datos curriculares y la forma de contactar a cada uno de ellos; de esta forma cada miembro del consejo conocerá las capacidades del otro. Esto puede hacerse vía Web o de alguna otra forma.

Además, se vio la necesidad de tener una base de datos de los socios y simpatizantes y este punto es el enlace con la siguiente área: la comunicación, pues la comunicación es la base de la unión.

1.1.2.2. Formas de comunicación oficial dentro de la CMDH

Un aspecto crítico también, es encontrar la manera de hacer más eficiente la comunicación de la asociación, para reducir los costos y garantizar su eficacia; en conversación con el cliente se discutió lo siguiente:

Las formas de comunicación oficial en la Asociación son tres:

1. Servicio de noticias.
2. Avisos selectivos.
3. Flujo de documentos.

1.1.2.2.1. Servicio de noticias

El servicio de noticias consiste en una síntesis noticiera diaria con lo más destacado de la política nacional, la cual es enviada por correo electrónico a todos los socios.

1.1.2.2.2. Avisos selectivos

Se trata de avisos, informes, invitaciones, etc. que se hacen llegar a determinados grupos de contactos, que forman la Red de Relaciones Públicas.

La Red de Relaciones Públicas es una base de datos de grupos de interés con el fin de contactar a cierto tipo de personas para ciertas cosas.

La Red de Relaciones Públicas está formada por un conjunto de grupos de personas, donde cada grupo es tratado como una unidad para enviarle la invitación o el aviso que se desea.

TABLA 1.1.2-1: Ejemplo de grupos en la *Red de Relaciones Públicas*

Grupo1: AMNISTÍA INTERNACIONAL Contacto 1 Contacto 2	Grupo 2: PROTECTORES DE LA FAMILIA Contacto 1 Contacto 2
Grupo 3: PRENSA Contacto 1 Contacto 2	Grupo 4: ECOLOGISTAS Contacto 1 Contacto 2

[12] Consejo Directivo y Consejo Consultivo.

El esquema funciona bajo las siguientes propiedades:

1. Un grupo es un conjunto de personas.
2. Una persona puede pertenecer a varios grupos.
3. Un mensaje sólo puede ser leído por personas pertenecientes al grupo al que les fue enviado.
4. Un mensaje puede ser enviado a uno o más grupos simultáneamente.

El método puede variar, desde correo electrónico, hasta página Web; la decisión tomada se explica más adelante, en la *fase de inicio*.

1.1.2.2.3. Flujo de documentos

La documentación que generan las vocalías, el consejo consultivo, las delegaciones, etc. debe llevar un control para hacer las reuniones del consejo más eficientes, promover la integración de los miembros y que todos estén enterados de los avances en el trabajo de las diferentes áreas de la CMDH.

La comunicación se da en tres momentos:

1. Un consejero[13] envía un documento o un comunicado.
2. Pasa por un sistema de autorización.
3. Se "publica" a los receptores según las normas de seguridad.

El modelo del flujo de documentos y comunicados puede verse con mayor claridad en el siguiente diagrama.

MODELO DE COMUNICACIÓN INTERNA

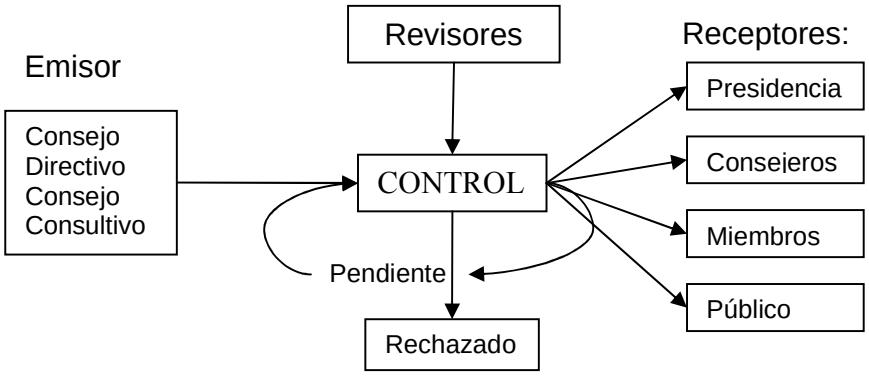

Fig. 1.1.2-1: Modelo de Comunicación Interna

Para la autorización se nombra una comisión revisora, que al momento de la entrevista se acordó como probable que fueran los vicepresidentes y el presidente o quien la presidencia designe para el caso.

La información que puede ser generada es la siguiente: noticias, eventos y documentos. Entre ellos puede haber relación, por ejemplo, cuando en un evento se publica un informe (documento) y tiene repercusión en prensa (noticia).

[13] Miembro del Consejo Directivo o del Consejo Consultivo (ver la sección 1.1.1. *¿Qué es la CMDH?*)

Los receptores están catalogados según el nivel de acceso a la información en un esquema de círculos concéntricos, de tal forma que las personas ubicadas en un determinado círculo tienen acceso a la información de su círculo y de los exteriores pero no de los interiores según se muestra.

CLASIFICACIÓN DE RECEPTORES

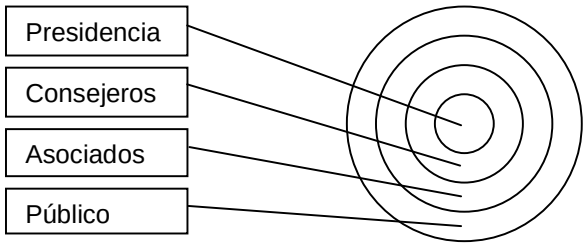

Fig. 1.1.2-2: Clasificación de receptores en círculos concéntricos

Sitio Web:

La motivación de realizar un sitio Web para la CMDH, es, según el cliente, dar una imagen ante la opinión pública, en la que se de a conocer las distintas actividades y opiniones relacionadas con la misma.

El tipo de audiencia se puede clasificar según los grupos de interés, que podemos resumir en los siguientes:

- Prensa
- Políticos
- Iniciativa privada
- Líderes sociales
- Organizaciones de la sociedad civil
- Organismos internacionales
- Instituciones educativas
- Jóvenes
- Mujeres

Los cuales buscan determinado tipo de información en el sitio y de determinada forma.

Las características del sitio Web deben ser, en cuanto a contenido, el mostrar las actividades y documentos que la CMDH decida publicar en un orden y formato adecuado para que sean de fácil acceso.

Además, esta página deberá rotar de manera automática estos contenidos, poniendo más cerca de los visitantes los contenidos más nuevos y desplazando los anteriores hasta retirarlos de la vista de los usuarios.

El sistema presentado en este trabajo resuelve el problema del *flujo de documentos* y el del *sitio Web* de manera conjunta.

1.2. MARCO TEÓRICO

1.2.1. Métodos tradicionales de la Ingeniería de Software: ventajas y desventajas

A continuación presentamos una breve descripción de los métodos tradicionales utilizados en la Ingeniería del Software, seguida de un comentario acerca de sus ventajas y desventajas, así como de la naturaleza de los proyectos donde conviene utilizarlos.

Cuando se hable del *Proceso Unificado* mostraremos en qué forma se incorporan la mayoría de estos métodos tradicionales para ofrecer mejores rendimientos.

1.2.1.1. Principio General

Roger S. Pressman[14] citando a Raccoon[15] sugiere un *«Modelo del Caos»* que puede describirse como un ciclo en el que se parte de un estado del problema, se define el problema, se realiza un desarrollo técnico, se integran soluciones y esto nos lleva a un nuevo estado del problema.

MODELO DEL CAOS BÁSICO

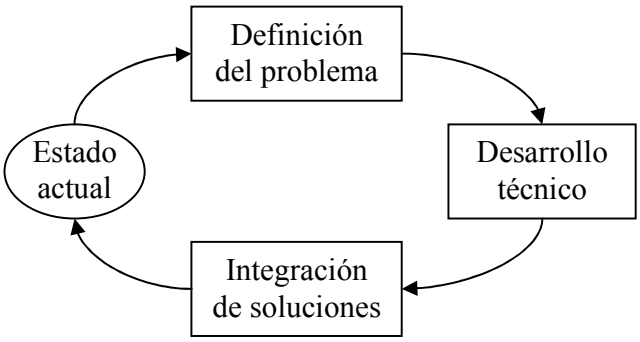

Fig. 1.2.1-1: Modelo del Caos Básico

Pudiéramos entender estos términos de la manera siguiente:

1. Existe el problema en la realidad.
2. Lo entendemos.
3. Proponemos e implantamos una solución.
4. Alteramos esa *realidad* solucionando parcial o totalmente ese problema.

Eso nos lleva a un nuevo estado en el que surgen nuevos problemas que podemos decidir resolver o no según sea el caso.

El modelo va más allá, pues cada etapa (figura *1.2.1-1*) puede ser compleja y constituir en sí misma un problema a resolver, y «*recursivamente*» se expande tomando la forma de un «*fractal*» como se muestra en la figura *1.2.1-2*.

[14] Pressman, Roger S.; Ingeniería del Software 5ª; Ed. Mc Graw Hill 2001; p. 19.
[15] Raccoon, L.B.S.; The Chaos Model and the Chaos Model Life Cycle; ACM Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA. 1993. La referencia en Pressman R.S; op. cit. loc.cit.

MODELO DEL CAOS EN FORMA DE FRACTAL

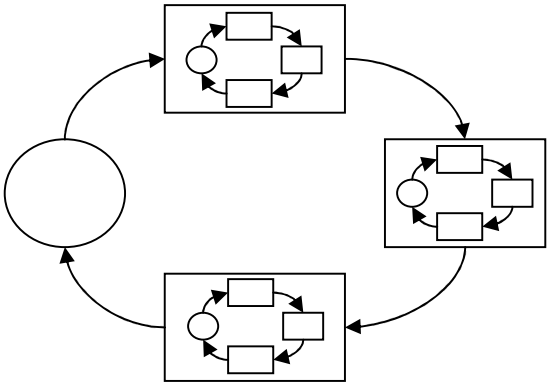

Fig. 1.2.1-2: Modelo del Caos en forma de fractal

Este modelo representa un patrón que aparece en todos los modelos clásicos del desarrollo del software de una u otra forma.

1.2.1.2. Modelo Secuencial o «en Cascada»

Propuesto originalmente por Winston Royce[16] en 1970, es un método secuencial cuyas etapas podemos resumir en cuatro:

1. Análisis.
2. Diseño.
3. Generación de código.
4. Pruebas.

Donde cada etapa significa:

- **Análisis:** En esta etapa se busca comprender los requisitos y la naturaleza del problema a resolver.
- **Diseño:** Durante esta actividad se proponen alternativas de solución y se elige la más adecuada.
- **Generación de código:** una vez elegida la solución, ésta se construye, es decir se hace el programa; el resultado es un ejecutable o fuente para ser compilado posteriormente.
- **Pruebas:** Finalmente se realizan las pruebas y correcciones al sistema para que funcione correctamente.

Por ser el más antiguo, también es el más utilizado de los métodos; pero ofrece serias dificultades, la experiencia personal muestra que un elevado porcentaje de proyectos se colapsan en la etapa de las pruebas, cuando salen a la luz los riesgos que no eran evidentes pues no se había probado el software.

Varios autores[17] han realizado diversas críticas a este método y ponen en duda su eficacia; nuestra apreciación personal es que el método debe ubicarse en su contexto, i.e. que funciona para pequeñas tareas cuyos requisitos y dominio de la solución son bien conocidos.

Como se verá, este *paradigma*, como también se le llama, es utilizado como base para otros métodos de la Ingeniería del Software, probando así su vigencia si se le sabe adaptar.

---

[16] Royce, W.W.; Managing the Development of Large Software Systems: Concepts and Techiques; Proc. WESCON, Agosto 1970; mencionado en Pressman, R. .S; op. cit.; p. 20.

[17] Así Pressman; op. cit., loc. cit.; lo mismo que Jacobson Ivar; El Proceso Unificado del Desarrollo de Software; Addison Wesley; 1a edición en español, Madrid 2000; en el prefacio y especialmente en el capítulo 5.

1.2.1.3. Modelo de Construcción de Prototipos

El principio de este modelo es el de «*prueba y error*». *Con frecuencia el cliente identifica los objetivos de una manera vaga o imprecisa*[18] o bien se requieren investigar determinadas características del sistema que son novedosas para los desarrolladores.

Cuando se presentan estas circunstancias, una solución conveniente es la construcción de un prototipo; entendemos por prototipo un «*modelo a escala*» del sistema que se desea construir; i.e. un sistema con un mínimo de características necesarias para poder investigar los requisitos del cliente o aquello que se desconoce.

El paradigma de la construcción de prototipos puede resumirse a las siguientes actividades:

1. Escuchar al cliente.
2. Construcción del prototipo.
3. Evaluación por parte del cliente.

CONSTRUCCIÓN DE PROTOTIPOS

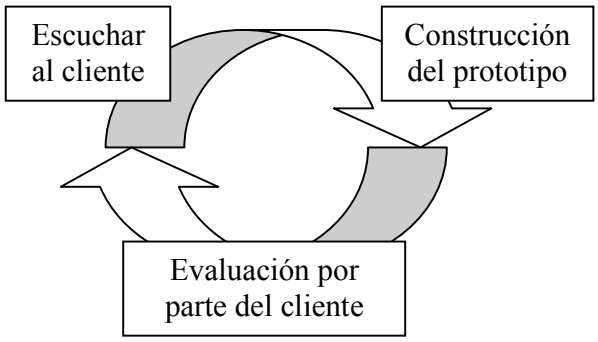

Fig. 1.2.1-3: Flujo de actividades en el paradigma de construcción de prototipos

Estas actividades forman un ciclo que se repite hasta que los requisitos o aquello que se quiere investigar se ha averiguado satisfactoriamente.

Aunque este modelo resulta útil para investigar requisitos desconocidos y para probar determinadas soluciones también desconocidas, resulta impráctico para la construcción de grandes sistemas, pues no sigue ningún camino excepto la intuición del cliente y del desarrollador.

El riesgo es el colapso debido a la incapacidad de integrar los diversos prototipos que forman el sistema. Otro problema generado es la falsa expectativa al cliente de que el «*sistema está funcionando*» cuando en realidad solamente algunas características aisladas funcionan bien.

Brooks[19] propone un esquema de «*desechables*», en el que primero se construye el prototipo de pruebas y una vez logrados los objetivos de las mismas empezar de nuevo el sistema verdadero, i.e. el sistema que se le va a entregar al cliente; así el problema para la gestión no es si tirar o no el prototipo a la basura sino cómo decírselo al cliente.

[18] Pressman, R. S.; op. cit. p. 21.
[19] Brooks, F.; The Mytical Man-Month; Addison-Wesley, 1975; citado por Pressman R.S. op. cit. loc.cit.

1.2.1.4. El modelo basado en componentes

Propuesto y desarrollado por Yourdon[20] en 1975, este modelo tiene por principio dividir el programa de software en pequeñas partes independientes llamadas *componentes* que se ensamblan para realizar la tarea solicitada.

Por tanto, el proceso de desarrollo se divide en pequeños desarrollos para cada componente.

Para que un sistema basado en componentes funcione, debe respetar dos principios: *alta cohesión* y *bajo acoplamiento*.

El principio de *alta cohesión*[21] consiste en que cada componente debe tener una solidez interna, entendida ésta como la capacidad del componente para realizar por sí mismo la tarea.

Por tanto, si un módulo se modifica internamente esto no alterará las dependencias externas.

El principio de *bajo acoplamiento*[22] señala que mientras menos conexiones y dependencias haya entre los componentes el sistema será más estable.

En términos de proceso de desarrollo de software esto permite entregar un producto funcionando parcialmente al que se le van agregando componentes o módulos gradualmente o bien realizar un sistema mediante equipos de desarrollo independientes y acelerar de esta forma el proceso.

Este modelo tiene una gran importancia, de él se derivan otros modelos como RAD, el modelo orientado a objetos y los modelos iterativos. Un ejemplo de la vigencia e importancia que tiene es que los sistemas UNIX están basados en componentes y son muy eficientes.

1.2.1.5. El Modelo RAD

El Desarrollo Rápido de Aplicaciones (RAD por sus siglas en inglés) es una adaptación del modelo en cascada con la característica de lograr un tiempo de desarrollo muy reducido. Si se ajustan los requisitos y el ámbito del sistema puede hablarse de períodos de 60 a 90 días.[23]

La base de este desarrollo rápido está en el desarrollo basado en componentes, la reutilización y la limitación de los requisitos y el ámbito del sistema adecuadamente.

Kerr[24] propone las siguientes fases:

1. Modelado de gestión.
2. Modelado de datos.
3. Modelado de procesos.
4. Generación de aplicaciones.
5. Pruebas y entrega.

La estrategia es dividir el proyecto en varios subproyectos independientes que siguen este camino cada uno, si se logran sincronizar todos entonces se logrará entregar el sistema completo en el tiempo esperado.

---

[20] Yourdon, Edward; Techniques of program structure and design; Prentice Hall 1975.
[21] Yourdon, E.; op. cit. cap. 7.
[22] Yourdon, E.; op. cit. cap. 6.
[23] Martin, J.; Rapid Application Development; Prentice-Hall 1991; citado por Pressman R.S. op. cit. p.22.
[24] Kerr J., R. Hunter; Inside RAD; McGraw-Hill, 1994; la referencia en Pressman R.S. op. cit. loc. cit.

EL MODELO RAD

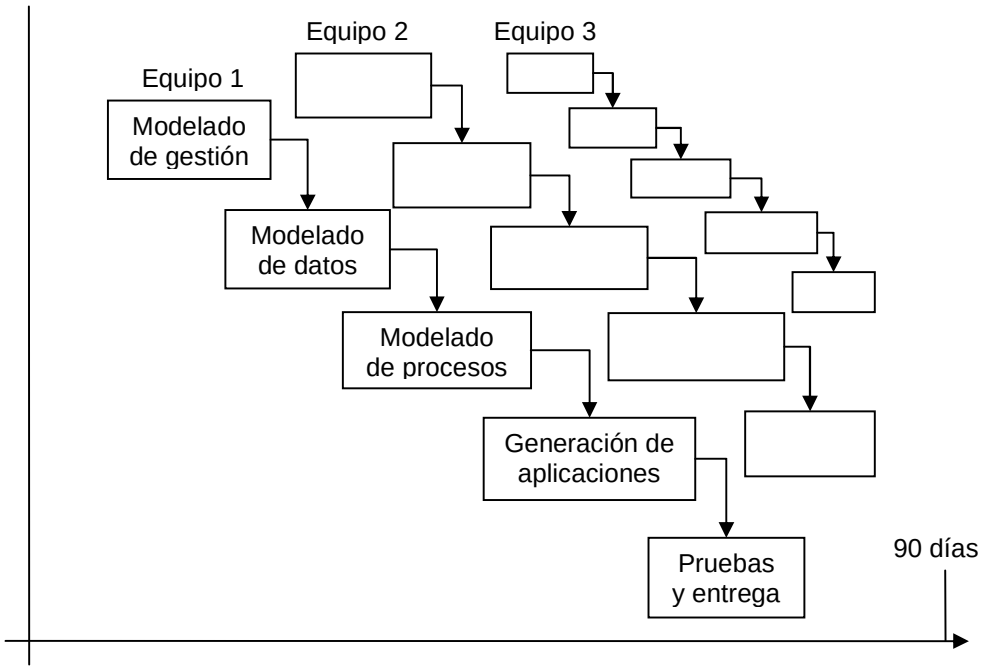

Fig. 1.2.1-4: El modelo RAD

Otra ventaja que ofrece es la reducción del colapso en las pruebas pues se utilizan componentes previamente probados así como generadores de código (técnicas de cuarta generación); todo esto reduce significativamente el tiempo del desarrollo del sistema.

Aun así, este modelo ofrece serios inconvenientes; el primero de ellos es la inflexibilidad al cambio de los requisitos, pues sólo se gestionan en la primera parte del desarrollo del sistema.

No todos los proyectos son susceptibles de ser resueltos por este método, aquellos que incorporen conceptos o tecnologías novedosas o bien aquellos cuyos requisitos no estén claros no son buenos candidatos.

Existen proyectos que difícilmente se pueden modularizar, éstos tampoco conviene resolverlos por este medio, pues el colapso en las pruebas no vendrá porque los componentes no estén probados por separados, sino por su difícil integración.

Los proyectos grandes o complejos tampoco son buenos candidatos por la cantidad de recursos humanos o económicos que demandan para su solución.

Por otra parte, no todos los equipos de desarrollo son aptos para este modelo; si entendemos por «equipo» a los desarrolladores y a los clientes; si unos u otros (o ambos) no tienen el suficiente grado de compromiso (o no tienen tiempo) para soportar un desarrollo a alta velocidad (e.g. reuniones diarias o varias veces al día) el proyecto fracasará.

Si el equipo tiene pocos integrantes (imposible hacer RAD personal) tampoco puede lograr este desarrollo, pues no tiene personal suficiente para encargarse de cada módulo por separado y tenerlos a tiempo.

1.2.1.6. Modelo Incremental

El modelo incremental combina elementos del modelo secuencial con la filosofía iterativa del modelo de construcción de prototipos.[25] Este modelo aplica una serie de secuencias lineales (pequeños modelos en cascada) de forma escalonada; digamos que sigue el modelo de Henry Ford, las «*pipe-lines*» usadas en la producción de automóviles.

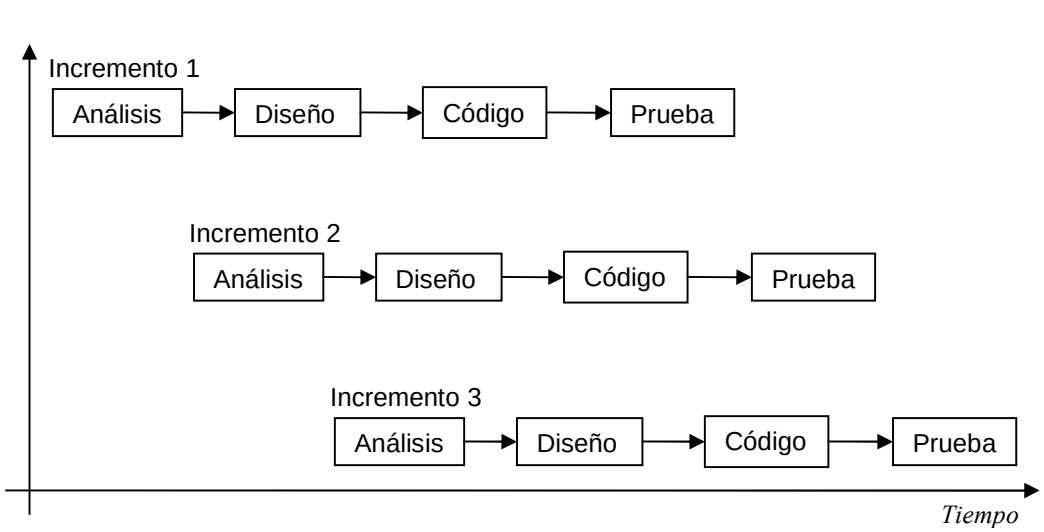

Fig. 1.2.1-5: El modelo Incremental

Cada secuencia produce un «*incremento*» en el software.[26] De forma que en la primera construcción se produce una funcionalidad elemental; y conforme avanzan los incrementos se perfecciona el sistema hasta que realiza funciones muy elaboradas. Además, en cualquier etapa se puede incluir la construcción de algún prototipo.

Este modelo presenta una gran ventaja cuando se tienen limitaciones de tiempo para la entrega de un determinado producto, pues permite que el cliente pueda empezarlo a utilizar aunque sea con un funcionamiento básico en el tiempo requerido.

Posteriormente el cliente recibe entregas sucesivas que perfeccionan su sistema hasta tener la versión completa del mismo. La diferencia entre el modelo incremental y el modelo de construcción de prototipos es ésta: que el modelo de prototipos sirve para investigar requisitos y por lo general el prototipo no sirve como sistema para el cliente; mientras que el modelo incremental realiza una serie de «*entregas*» del producto todas ellas utilizables por el cliente.

Sin embargo, este modelo ofrece diversos inconvenientes; en particular existe el riesgo de realizar iteraciones equivocadas si, por ejemplo, no se integran los resultados de las pruebas de la iteración anterior a la siguiente (véase la figura *1.2.5*); esto produce dificultades de integración en las entregas hasta colapsar el proyecto.

El mérito de éste método es ser la base de los *modelos evolutivos*, de aquellos que son flexibles al cambio en los requisitos. A continuación veremos algunos de ellos.

---

[25] Pressman R.S. op. cit. p. 23.
[26] McDermid, J., P. Rook; Software Development Process Models; CRC Press 1993. La referencia en Pressman R.S. op. cit. loc. cit.

1.2.1.7. El modelo en espiral

Propuesto originalmente por Boehm en 1988, este modelo incluye elementos nuevos al modelo incremental, tales como el *análisis de riesgos*; es un concepto distinto, pues no se inicia un nuevo incremento hasta terminado y evaluado el anterior.

EL MODELO EN ESPIRAL

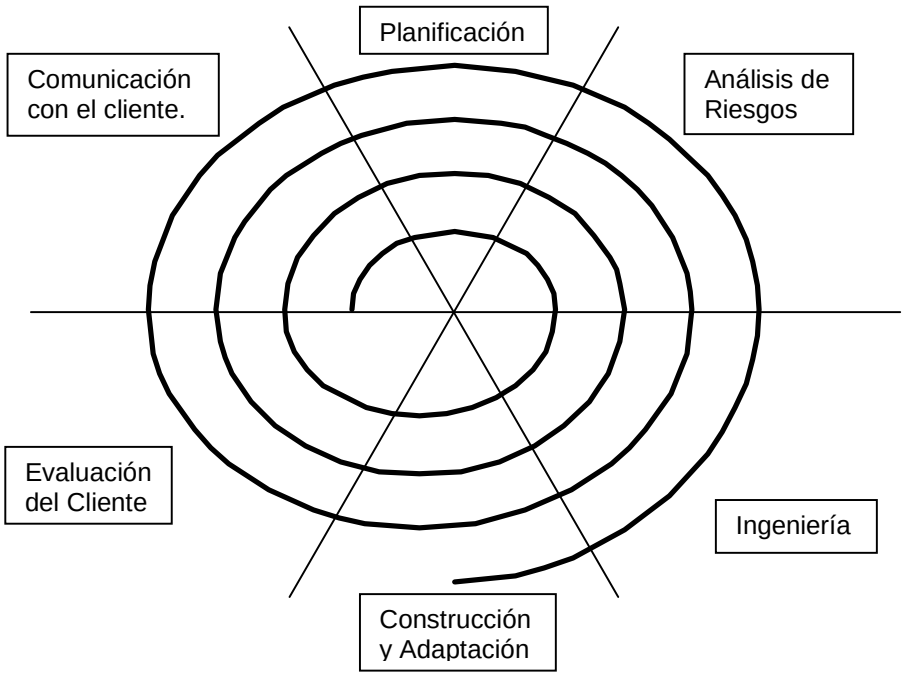

Fig. 1.2.1-6: Modelo en espiral

La figura *1.2.1-6* muestra un modelo en espiral típico, la línea del tiempo forma la espiral.

Este modelo involucra, para cada iteración, una serie de actividades a realizar para lograr un incremento en el sistema. Estas actividades forman *regiones comunes* para los distintos incrementos.

Cada incremento forma en sí mismo un proyecto, los primeros proyectos son más conceptuales y elementales; los últimos proyectos realizan funciones más elaboradas.

Las *regiones de actividad* podemos resumirlas en seis:

1. Comunicación con el cliente.
2. Planificación.
3. Análisis de riesgos.
4. Ingeniería (análisis y diseño).
5. Construcción y adaptación.
6. Evaluación del cliente

Este modelo tiene significativas ventajas sobre los anteriores, en particular, el acoplamiento entre los distintos incrementos es posible, pues no se inicia un nuevo incremento hasta no obtener la evaluación del cliente respecto del anterior.

Este acoplamiento, por una parte significa un retraso, comparado con el modelo incremental, pero por orta parte evita incrementos desperdiciados y gestiona de manera más adecuada los cambios en los requisitos propuestos por el cliente.

Otra de las ventajas está representada por las actividades de *análisis de riesgos* y la *gestión de la configuración* que son prioritarias en el modelo de espiral. Este modelo permite además, el uso de *prototipos de prueba* en cualquier parte de su ciclo de vida, incorporando las ventajas de este paradigma de desarrollo.

1.2.1.8. Modelo en espiral WINWIN

Este modelo es utilizado para comprender y negociar los requisitos de un sistema. No es más que una adaptación del modelo en espiral con una variante en las actividades orientadas a la negociación:

1. Identificar el siguiente nivel para los directivos.
2. Identificar las *condiciones de victoria* para los directivos (y desarrolladores).
3. Reunir las *condiciones de victoria*. Establecer los objetivos restricciones y alternativas del siguiente nivel (de manera que todos ganen).
4. Evaluar las alternativas del producto y del proceso y resolución de riesgos.
5. Definir el siguiente nivel del producto y del proceso, incluyendo particiones.
6. Validar las definiciones del producto y del proceso.
7. Revisión y comentarios.

Este modelo podemos aplicarlo en tres formas:

La primera, como una subdivisión de la tarea «*comunicación con el cliente*» en el modelo en espiral tradicional. **La segunda**, como *iteraciones al inicio* del proyecto y cuando la negociación sea adecuada se usa un modelo en espiral tradicional o bien la primera forma (inicio de este párrafo). **Finalmente** se puede hacer una *expansión en forma de «fractal»*; i.e. hacemos una pequeña espiral de varias iteraciones en cada etapa de «*comunicación con el cliente*».

1.2.1.9. Modelo Concurrente

Este modelo más que trazar un camino para construir eficientemente un sistema de software, intenta describir el *estado real* en que se encuentra el desarrollo del mismo.

Para esto, analiza cada una de las actividades del desarrollo del software, por ejemplo, las propuestas en el modelo secuencial clásico; y les asigna un estado, ya que el primer principio del modelo concurrente es que de una u otra forma, todas las actividades se desarrollan concurrentemente.

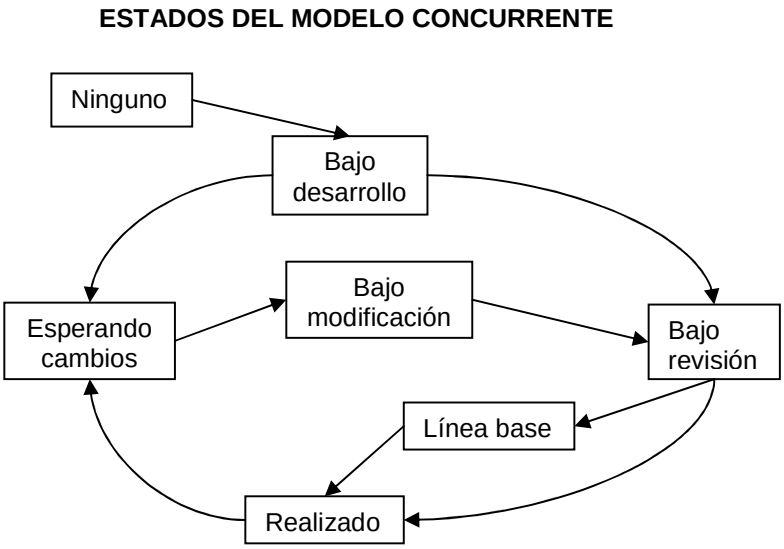

Fig. 1.2.1-7: Diagrama de Estados del Modelo Concurrente

Posteriormente se realiza una descripción de las transiciones entre estados debido a los eventos; por ejemplo, si se está realizando el análisis pero el cliente decide cambiar los requisitos, entonces la actividad «*análisis*» pasa al estado «*esperando cambios*» y la actividad «*entrevista con el cliente*» pasa al estado «*bajo modificación*».

Es posible que la construcción de determinada parte del sistema (actividad «*construir código*») no se vea afectada y que esté en la etapa de pruebas (estado «*bajo revisión*») y que la documentación de otro módulo no se vea afectada y esté lista, en el estado «*línea base*», etc.

1.2.1.10. Modelo Orientado a Objetos

El modelo basado en componentes que se originó en los 70, se transformó en el modelo *orientado a objetos*, que en los 90 tuvo un gran desarrollo.

Los componentes de software evolucionaron a un concepto llamado *clase*, que es una estructura que agrupa tanto datos como algoritmos de funcionamiento.

Una clase es un tipo de dato abstracto, del cual se sacan copias; algo así como el sistema de Platón en que los objetos reales son copias de un modelo ideal; esas copias son llamadas *objetos*.

Además estas *clases* tienen propiedades especiales, la más importante es la *herencia* que es la capacidad de una clase de transferirle todas sus características a otra clase.

Desde el punto de vista del proceso de desarrollo, este modelo inicialmente es igual que el modelo basado en componentes en el sentido de que el trabajo de desarrollo se puede dividir en partes más pequeñas que se pueden hacer de manera independiente e incluso aprovechar componentes de otros proyectos anteriores.

Pero ofrece otras ventajas mayores; la primera es la simple noción de *clase*, agrupando los datos y algoritmos en un *tipo abstracto* del que pueden obtenerse varias copias (instancias) y usarse según convenga.

Otra gran ventaja es la *herencia*, porque permite el desarrollo a muy alta velocidad de componentes similares e intercambiables.

Por el tiempo en que este enfoque fue desarrollándose, se conceptualiza como un modelo iterativo por naturaleza, en donde su ciclo de vida tiene la forma de una espiral, por lo que también incluye las ventajas del modelo en espiral.

1.2.2. El Proceso Unificado: descripción.[27]

En este apartado muestro una breve descripción de las principales características del Proceso Unificado del Desarrollo del Software desarrollado por Racional Inc.; comenzando por su historia, termino señalando de qué manera se incorporan los enfoques tradicionales y el motivo por el que elegimos este método para realizar el sistema.

1.2.2.1. Historia del Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (RUP por sus siglas en inglés) es el resultado de 30 años de experiencia en el desarrollo de sistemas y de la colaboración de múltiples corporaciones en el ramo; dando como resultado una de las metodologías más robustas que se encuentran en el mercado.

HISTORIA DEL PROCESO UNIFICADO DE DESARROLLO

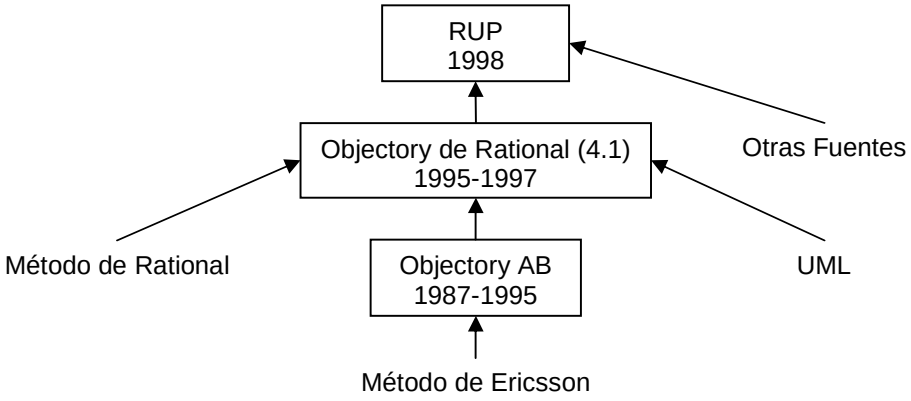

Fig. 1.2.2-1: Historia del Proceso Unificado de Desarrollo

1.2.2.1.1. El método de Ericsson

En 1967 la empresa Ericsson desarrollaba sus sistemas usando diagramas de bloques que representaban partes del sistema, aislados e independientes, interconectados, que se comunicaban mediante mensajes los cuales se especificaban en un diccionario de mensajes; estos bloques podían a su vez contener bloques internos; cada bloque realizaba una función específica llamada caso de negocio; los casos de negocio y el diccionario de mensajes dirigían el desarrollo de todo el sistema y servían para comunicarse con el cliente e informarle sobre los avances del proyecto.

Cada bloque se transformaba en una pieza ejecutable y podía ser instalada en el sistema incluso cuando éste estaba ejecutándose; éste requisito era indispensable, pues se trataba

[27] Esta sección está basada en Jacobson Ivar, Booch Grady, Rumbaugh James; *El Proceso Unificado de Desarrollo de Software*; 1ª Ed. en español, Adison-Wesley; Madrid 2000.

de sistemas de telecomunicaciones que no podían suspender sus servicios por ningún motivo.

Ivar Jacobson fue el creador de este sistema que luego daría lugar al paradigma de desarrollo basado en componentes.

1.2.2.1.2. SDL

El año de 1976, el CCITT, publicó el Lenguaje de Especificación y Descripción (SDL por sus siglas en inglés). Este lenguaje toma muchos aspectos del método de Ericsson, y los formaliza; por ejemplo, los bloques los llama procesos y los mensajes señales.

Desde entonces, SDL viene a ser un estándar ampliamente utilizado principalmente por los desarrolladores de sistemas orientados a red como UNIX en todas sus variedades.

1.2.2.1.3. Objectory AB

En 1987 Ivar Jacobson deja Ericsson y funda Objectory AB, en esta compañía realiza el método Objectory (el nombre es la abreviación de Object Factory) y durante ocho años trabaja con este método haciendo importantes aportes apoyado en la tecnología que antes usara en Ericsson y el enfoque orientado a objetos.

Formaliza los casos de negocio y acuña el término caso de uso, el proceso de Objectory estaba dirigido por casos de uso; que es la forma de captura de requisitos.

Para las diferentes actividades se establecieron distintos diagramas: casos de uso, análisis, diseño, implementación y pruebas; y fue requisito que se pudiera establecer una traza entre los diferentes diagramas, de tal forma que un cambio (importante) en uno de estos diagramas pudiera observarse en los demás.

Esta trazabilidad era la expresión de que el proceso estaba dirigido por los casos de uso.

Finalmente Objectory se concibió como un proceso de ingeniería que podía ser tratado como un producto de ingeniería; de esta forma Objectory podía desarrollarse y perfeccionarse a sí mismo; con esto se logró aumentar su robustez y consistencia con el tiempo.

1.2.2.1.4. Rational

Desde 1991 la empresa Rational Software Corporation estaba desarrollando una metodología para sustentar sus productos software de manera más eficiente.

Se basaron en dos principios fundamentales: la arquitectura y el desarrollo iterativo.

Rational SC consideró que sus sistemas debían estar basados en una arquitectura, entiéndase arquitectura como el esqueleto del sistema, aquella parte que da soporte al resto del sistema, que realiza las funciones más críticas y que es resistente a los cambios en los requisitos.

La arquitectura tuvo la innovación de presentarse con varias vistas: vista lógica, vista de procesos, vista física, vista de desarrollo mas una vista conjunta mediante escenarios.

Respecto al desarrollo iterativo también hubo avances significativos, pues las iteraciones eran agrupadas en fases para tener un mejor control sobre ellas.

Estas fases eran: inicio, elaboración, construcción y transición. Cada una de ellas tenía una o más iteraciones y cada fase buscaba un objetivo preciso; por ejemplo, la fase de elaboración buscaba construir la arquitectura del sistema.

Para 1995 esa metodología ya estaba madura y en 1996 Grady Booch que estuvo desde los comienzos de Rational SC se expresaba de esta forma:

- Un estilo de desarrollo dirigido por la arquitectura es normalmente la mejor aproximación para la creación de la mayoría de los proyectos complejos basados en el software.
- Para que un proyecto orientado a objetos tenga éxito debe aplicarse un proceso iterativo e incremental.

1.2.2.1.5. Rational Objectory

En 1995 Rational SC compra Objectory AB y se dio a la tarea de unificar las metodologías desarrolladas por ambas empresas.

En el momento de la fusión, Objectory 3.8 había demostrado que se puede crear y modelar un proceso de desarrollo software como si fuese un producto.

Había alcanzado un nivel de madurez capaz de soportar proyectos software relativamente complejos y documentarlos de manera robusta. Sin embargo, en áreas como la gestión de requisitos (excepto los casos de uso), implementación y pruebas, no estaba tan bien desarrollado.

Objectory decía poco sobre gestión del proyecto, gestión de la configuración, distribución y preparación del entorno de desarrollo.

Por esto se le añaden la experiencia y prácticas de Rational para formar el Proceso Objectory de Rational 4.1 (ROP 4.1 por sus siglas en inglés); en concreto se añadieron las fases para un desarrollo iterativo controlado y la arquitectura como la parte más importante del sistema.

El método iterativo pasó de ser un método general a ser un método dirigido por los riesgos donde se consideraba a la arquitectura en primer lugar. UML estaba desarrollándose también, así que se incorporó como el lenguaje de modelado de ROP.

Finalmente, gracias a las aportaciones de Royce, se añadieron algunas correcciones reforzando, por ejemplo, la gestión del proyecto.

1.2.2.1.6. UML

Ivar Jacobson, Grady Booch y James Rumbaugh son los creadores de UML, que fue el lenguage de Rational Objectory Process y actualmente lo es de Rational Unified Process.

UML es la abreviatura de *Unified Modeling Languaje*, el cual surge de la necesidad de tener un lenguaje estándar visual de modelado para el desarrollo software.

Este lenguaje hunde sus raíces en la experiencia de tres desarrolladores que trabajaron juntos en Rational Software Corporation: Grady Booch, James Rumbaugh e Ivar Jacobson.

Grady Booch era el autor del método Booch y James Rumbaugh el creador del método llamado Object Modelling Technique (OMT); Rumbaugh se incorpora a Rational en octubre de 1994 y junto con Booch unifican sus trabajos y para octubre de 1995 terminan la versión 0.8 de UML.

En octubre de 1995 Ivar Jacobson se incorpora a Rational y aporta sus trabajos a los de Booch y Rumbaugh; al poco tiempo publican la versión 0.9, compañías como HP, Microsoft e

IBM aportan ideas para la estandarización y en noviembre de 1997 se publica la versión 1.1 de UML.

1.2.2.1.7. Rational Unified Process

De 1995 a 1997 Rational SC compró y se fusionó con varias empresas, las cuales hicieron sus aportaciones al proceso, Ivar Jacobson, Grady Booch y James Rumbaugh en su libro sobre el Proceso Unificado de Desarrollo de Software presentan una lista a manera de ejemplo, podemos listar los siguientes:

1. Requisite Inc. aportó su experiencia en la gestión de requisitos.
2. SQA Inc. había diseñado un proceso de pruebas que acompañaba el desarrollo de sus productos, mismo que fue añadido a ROP.
3. Pure-Atria añadió su experiencia en gestión de configuración.
4. Performance Awareness añadió las pruebas de rendimiento y de carga.
5. Vigortech aportó su experiencia en ingeniería de datos.

Además, el método se vio beneficiado por aportaciones de usuarios externos que se habían beneficiado por el proceso (ROP) y el lenguaje (UML); con el tiempo, el método y el lenguaje se vieron robustecidos con estas aportaciones y diversas extensiones, por ejemplo, que las interfaces de usuario estén dirigidas por casos de uso, etc.

En junio de 1998 Rational Software Corporation publica una nueva versión que se llamó Rational Unified Process 5.0 (RUP); en la cual se hace énfasis de que es un método unificado de muchos colaboradores dentro y fuera de Rational SC.

1.2.2.2. Características de RUP

El proceso unificado de desarrollo (cuyo concepto fundamental fue desarrollado por Ivar Jacobson el cual es a la fecha el principal desarrollador del mismo) es un proceso con una triple característica: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

1.2.2.2.1. Dirigido por casos de uso

Los casos de uso son la forma de expresar los requisitos de forma tal que tanto los clientes, como los usuarios y los desarrolladores puedan comprender qué es lo que debe hacer el sistema para cada usuario en cada circunstancia.

Que el proceso esté dirigido por casos de uso quiere decir que el desarrollo del sistema de software, estará siempre orientado a buscar que el sistema haga lo que los usuarios quieren que realice.

Esto proporciona una ventaja significativa de eficiencia, pues la atención del equipo de desarrollo se va a centrar en resolver las necesidades expresadas por el cliente y no va a perder el tiempo en hacer otras cosas.

1.2.2.2.2. Centrado en la arquitectura

La arquitectura es el conjunto de casos de uso críticos para el funcionamiento del sistema, de forma tal que el resto de los casos de uso se acoplan a éstos. La arquitectura es el esqueleto de la construcción del sistema, no sólo para esta versión, sino para las venideras.

El desarrollo (análisis, diseño, codificación, documentación y pruebas) de los casos de uso de la arquitectura es conocido como *línea base de la arquitectura*. El Proceso Unificado de

Desarrollo está centrado en la arquitectura porque busca una solución (y su prueba) a los problemas críticos primero y establecer un patrón de desarrollo y evolución del sistema, de tal forma que se reduzcan los costos debido a cambios en los requisitos de última hora.

Establecer una arquitectura robusta significa reducir los riesgos a tiempo y crear una estructura resistente a los cambios en los requisitos posteriormente; esto es crucial para realizar sistemas eficientes y confiables.

1.2.2.2.3. Iterativo e Incremental

El Proceso Unificado de Desarrollo es iterativo e incremental porque busca crecer en base a iteraciones, una iteración es un incremento en el avance del sistema, a través de un proceso completo de desarrollo: casos de uso, análisis, diseño, implementación, pruebas y documentación.

El propósito de esto es múltiple. En primer lugar, la mitigación de los riesgos; además, que tanto los clientes y los usuarios, como los desarrolladores y jefes del proyecto tengan una medida sólida del avance del proyecto.

Otra de las ventajas es la capacidad de hacer evaluaciones tempranas para cambiar los requisitos o bien reprogramar la agenda de proyecto.

Siguiendo un patrón de pequeños incrementos se corrigen los defectos más críticos al principio y se reduce el tiempo de entrega.

Los creadores de RUP establecieron un diagrama por el cual se muestra los flujos de trabajo y las iteraciones, las fases del desarrollo del sistema y la carga de trabajo (aproximado) que se espera en cada parte:

FASES DEL PROCESO UNIFICADO DE DESARROLLO

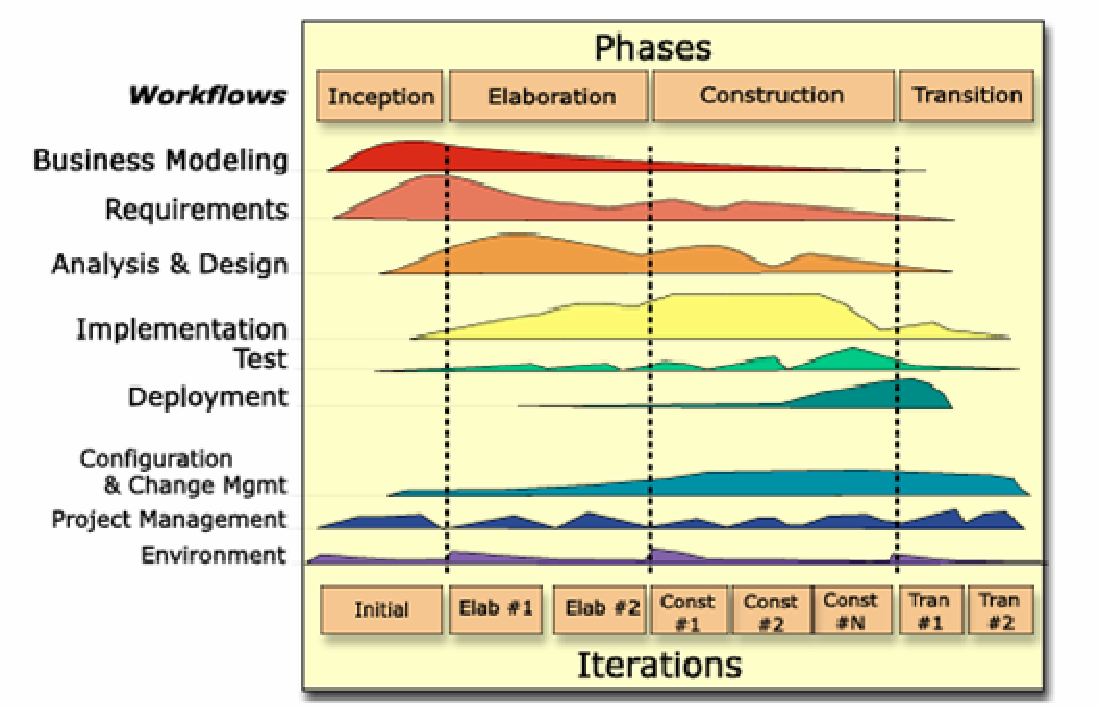

Fig. 1.2.2-2: Diagrama de fases del Proceso Unificado de Desarrollo

Como puede observarse, el proceso Unificado de Desarrollo de Software, incorpora la mayoría de los métodos tradicionales y de técnicas propias.

Podemos enunciar algunos: el método en cascada se encuentra en los flujos de trabajo fundamentales, el modelo en espiral se encuentra en que es iterativo e incremental, el modelo de prototipos no se muestra pero en el libro donde presentan RUP recomiendan la prueba con prototipos desde el comienzo; los modelos basados en componentes en su versión de modelos orientados a objetos son la base del funcionamiento del método; en particular de la arquitectura.

Por estos motivos y ventajas; y por ser el método más recomendado en la industria actualmente, es que hemos elegido RUP como el método para desarrollar el proyecto de la tesis.

1.2.3. Conceptos a Tener en Cuenta en el Diseño de Sistemas Web: Diferencias Frente a Otros Sistemas de Software

1.2.3.1. Nota Preliminar

A continuación se enunciarán los conceptos que se consideran necesarios para un correcto entendimiento de lo que son los sistemas Web y lo que engloban para entender las adaptaciones específicas que debe hacerse de la ingeniería de software aplicada a este ramo.

No se busca una formalidad estricta en el lenguaje de Internet sino establecer los conceptos básicos que se estarán utilizando en el resto del presente trabajo.

1.2.3.2. Internet

Internet es un conjunto de redes y servicios que en su conjunto forman una red que abarca a todo el mundo y da soporte a múltiples servicios, desde la navegación y el correo electrónico, hasta la transferencia de dinero en cuentas bancarias; también permite el intercambio de archivos y en general todo lo que signifique compartir información entre una red y otra a muy bajo costo.

1.2.3.3. Web

Podemos entender la Web desde un triple enfoque, en primer lugar, desde un enfoque *físico* o más bien, *fisiológico*; i.e. entender la Web desde el punto de vista de las tecnologías que le dan soporte y que reciben el nombre de Internet.

En segundo lugar, se puede ver desde un punto de vista *histórico* y como la aplicación de la *ingeniería del conocimiento* en un proyecto cuya meta es hacer un sistema de información global. En este enfoque puede verse un sistema Web como un documento electrónico, con características nuevas que la documentación impresa no posee.

En este segundo enfoque hay que tener en cuenta al auditorio, es decir, los lectores, para poder captar su atención y retenerlos hasta lograr el objetivo de nuestra publicación; es así que un sitio Web se basa en la Ingeniería del Conocimiento y en la Pedagogía para lograr el éxito.

Finalmente, los sistemas Web se pueden entender como aplicaciones software que funcionan desde Internet. Este enfoque lo explicaremos bajo el nombre *WebApp*.

Planteados los tres enfoques abordo el segundo, que es además el propio a este vocablo; la referencia es el sitio de la World Wide Web Consortium (W3C) que es la organización que marca las normas a seguir en este campo.

Web es la abreviación de World Wide Web, que es (según la W3C) un proyecto mundial cuya meta principal es hacer accesible para todo el mundo la información de todo el mundo.

Este proyecto (que también se abrevia WWW) hunde sus raíces a 1945, cuando Vannevar Bush escribió un artículo de carácter científico que tenía la característica de tener enlaces en su interior hacia otras partes del artículo u otros artículos.

En los 60 Doug Engelbart desarrolló un prototipo de navegador llamado “oNLine System” (NLS) con capacidad de seguir los vínculos, enviar correos electrónicos, etc. Inventó el *mouse* con el propósito de facilitar la navegación entre los enlaces.

A partir de entonces se fueron perfeccionando, por una parte, los navegadores de Internet, por otra, los protocolos de navegación (HTTP, FTP,etc.) y por otra, los lenguajes de los documentos (HTML, XML) hasta ser lo que son hoy en día.

Actualmente un navegador hace mucho más que mostrar un documento con enlaces, es prácticamente un *navegador de aplicaciones remotas* con características GUI, etc.

Por otra parte, los protocolos de red y los servicios de Internet también se han hecho muy variados y complejos; por ejemplo, existe la posibilidad de transferir información bancaria mediante un canal cifrado de forma transparente para las aplicaciones y los usuarios y de forma segura.

Los lenguajes de los documentos también evolucionan, y se diversifican, ahora se soportan imágenes, elementos GUI, subprogramas, “scripts”; la última tendencia es separar el formato (CSS) y los datos (XML) pero la forma en que esto se desarrollará queda aún pendiente.

1.2.3.4. WebSite: elementos a tener en cuenta

En este apartado ilustramos los elementos a tener en cuenta al construir un sitio Web cualquiera, tomamos la información de Lynch, Patrick J. & Horton, Sarah; Web Style Guide; 2nd Ed. Versión en Internet.[28]

En el diseño un sitio Web debemos considerar en primer lugar el auditorio, es decir, los visitantes del sitio. Este enfoque debe regir el diseño en su totalidad. ¿Cómo es este auditorio? Un estudio de mercado debe ser prioridad para cualquier sitio Web comercial, para identificar los gustos y expresiones culturales del grupo humano al que se quiere atraer al sitio; al margen de esto, existen una serie de características comunes a la mayoría de los cibernautas y que deben tenerse en cuenta.

Una característica común a la mayoría de los usuarios de Internet, es su falta de compromiso con los sitios que visitan; una persona que visita algún sitio no siente ninguna obligación de permanecer ahí; si le atrae, si le interesa, si encuentra lo que busca sin dificultad, regresa; en caso contrario no regresa.

Otra característica es la aleatoriedad de los documentos; es posible acceder a los documentos en cualquier orden y no necesariamente el que el diseñador haya predispuesto; de donde surge una pregunta cuya respuesta es vital para conservar al visitante ¿en qué parte del sitio me encuentro al leer esta página?

[28] Puede consultarse en http://www.webstyleguide.com/

La respuesta a esta pregunta son las opciones de navegación, que son los menús, barras de navegación, mapas del sitio, indicadores de avanzar o retroceder en el tema, índices, etc.

Por último añado otra característica, que es la de que todos los usuarios de Internet están sujetos a las restricciones tecnológicas que el mismo Internet impone; i.e. utilizan navegadores con determinada resolución de pantalla, los datos se transfieren a ciertas velocidades, etc.

Esto y las condiciones particulares de cada usuario dan origen a lo que se conoce como accesibilidad, que en un principio se entiende como las medidas que se toman para que personas con dificultades visuales, de movimiento o tecnológicas puedan acceder a la información; pero en un sentido más amplio se refiere a que la información esté al alcance de todos según sus particulares necesidades.

Un ejemplo de este concepto amplio de la accesibilidad es que ahora se navega en Internet desde el celular y los dispositivos de mano llamados Palm; los cuales no tienen las características de resolución y multimedia de los navegadores de una PC.

A partir de aquí conviene adoptar un enfoque que tienda a la especialización sin extenderse demasiado. Por lo que es necesario considerar algunos parámetros para identificar nuestro mercado objetivo así como la naturaleza de la información que pretendemos mostrar.

De acuerdo con la complejidad de la información que mostramos, podemos adoptar uno de los siguientes patrones en el diseño de la información del sitio:

Secuencias: Todas las páginas siguen un orden lineal, de tal forma que cada una (excepto las extremas) tienen una anterior y una siguiente), son muy predecibles y evitan la confusión.

SECUENCIA

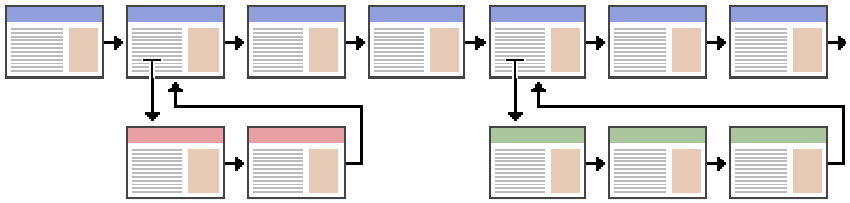

Fig. 1.2.3-1: Secuencia con algunas salientes

En sitios didácticos y de auto-aprendizaje resulta conveniente esta estructura.

Jerarquías: Cada página se encuentra ubicada con otras a un mismo nivel, tiene una de nivel superior y puede tener una o más de nivel inferior.

JERARQUÍA

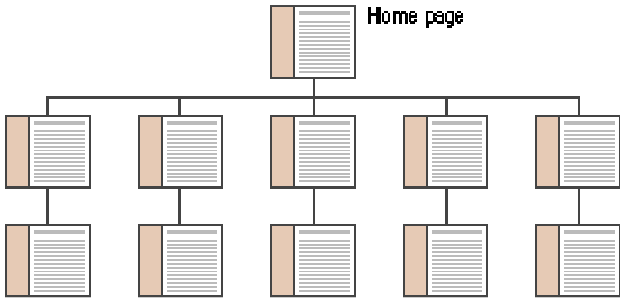

Fig. 1.2.3-2: Estructura jerárquica de un sitio

Esta estructura de la información es muy utilizada en sitios corporativos.

Redes: No se sigue ningún orden en especial, se usa esta estructura con público que conoce el tema y busca aspectos muy específicos; en general no es recomendable pues genera confusión.

RED (WEB)

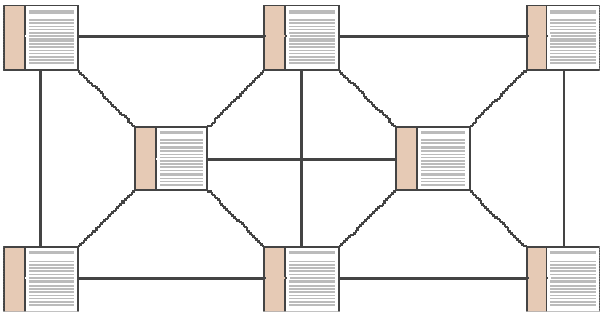

Fig. 1.2.3-3: Estructura en red (web)

El siguiente diagrama muestra una comparación de las estructuras (patrones) de la información descritos anteriormente:

ESTRUCTURAS DE INFORMACIÓN

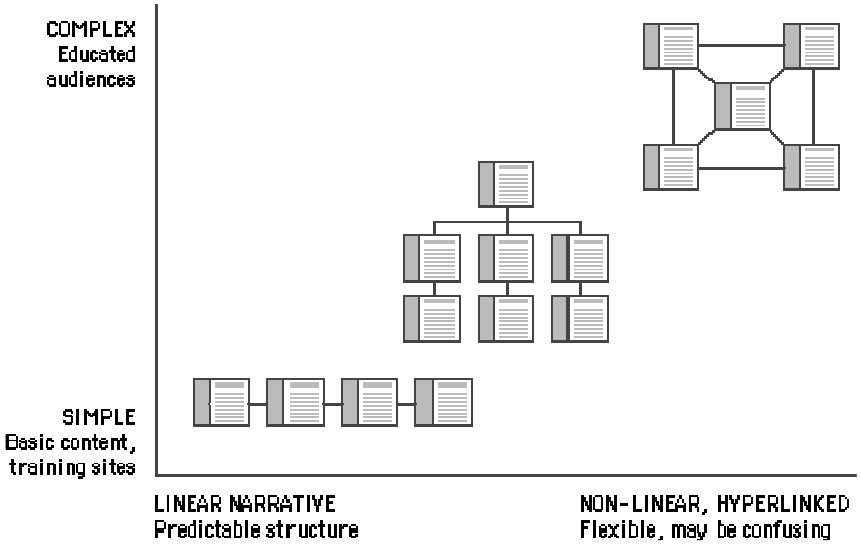

Fig. 1.2.3-4: Gráfica comparativa de las estructuras de información

Según los autores del libro en que nos estamos apoyando, los diferentes contenidos de un sitio Web se pueden ajustar a la misma gráfica y lo muestran de la siguiente forma:

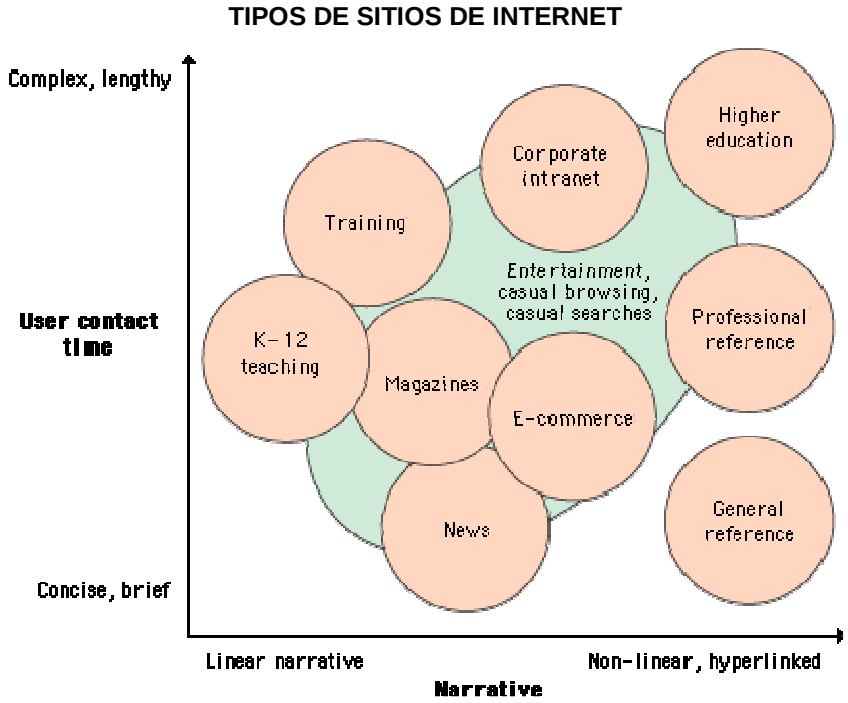

Fig. 1.2.3-5: Gráfica comparativa de los diferentes tipos de sitios de Internet

Por tanto, si elegimos por ejemplo, realizar un sitio para una organización, para sus miembros; una estructura jerárquica con algunos enlaces no jerárquicos es lo más adecuado.

1.2.3.5. *WebApp*: Diferencias y restricciones respecto de otros sistemas software

En este apartado mostramos los rasgos distintivos de los sistemas software que funcionan desde Internet, usando el enfoque Web de un sitio que ofrece el servicio y múltiples usuarios que hacen uso de él.

Apoyándonos en R.S. Pressman[29] y en nuestra experiencia personal, afirmamos que los rasgos distintivos para las aplicaciones Web (*WebApp*) son, según el autor, los siguientes:

- Intensivas en red.
- Dirigidas por el contenido.
- En continua evolución.
- Inmediatez (el tiempo entre el concepto y su colocación en el mercado es muy reducido).
- Seguridad: es imperativa, pues virtualmente todo el mundo puede acceder al sistema.
- Estética: para atraer al usuario y que regrese.

De nuestra experiencia, encontramos las siguientes restricciones en una aplicación Web:

- Ubicada en el esquema cliente-servidor, donde el cliente son los navegadores y el servidor puede ser un servidor o una red de servidores.
- Limitación de protocolo: HTTP, HTTPS, FTP.
- El cliente es la mayoría de las veces un navegador Web, por tanto, el servidor envía sus resultados en un conjunto de lenguajes específicos: HTML, XML, Javascript, etc.

[29] Pressman R.S. op. cit. pp. 522-523.

DESARROLLO

Este capítulo describe el proceso que se siguió para obtener el sistema solicitado; igualmente describe el sistema y sus características. El capítulo tiene seis partes: las dos primeras se refieren a la primera entrevista con el cliente y a la solución del problema de hospedaje del sitio (*Web Hosting*) respectivamente; las cuatro últimas corresponden con las cuatro fases del Proceso Unificado:[1] *Inicio*, *Elaboración*, *Construcción* y *Transición*.

[1] Véase la sección *1.2.2*. del presente trabajo.

2. DESARROLLO

2.1. PRIMERA ENTREVISTA CON EL CLIENTE: OB-TENCIÓN DE LOS REQUISITOS BÁSICOS

Un requisito es una *condición o capacidad que debe cumplir un sistema*.[2]

Durante la entrevista se obtuvieron los requisitos que podría tener el sistema, quedando sujetos a cambios y refinamiento posteriores, los cuales se dividen en dos grupos:

1. Requisitos funcionales.
2. Requisitos no funcionales.

2.1.1. Requisitos Funcionales

Por requisito funcional entendemos aquel que *especifica una acción que debe ser capaz de realizar el sistema sin considerar restricciones físicas*,[3] de entorno o de implementación.

El sistema está delimitado, de acuerdo con las necesidades del cliente, de la siguiente forma:

1. El sistema deberá realizar en su totalidad y automáticamente el control del flujo de documentos por vía electrónica según el modelo visto en la sección *1.1.2 Ámbito del Negocio: Necesidades Expresadas por el Cliente*.
2. El sistema deberá dar soporte total y automático al sitio Web, para el manejo de su contenido y para su mantenimiento.[4]

De acuerdo con esta delimitación, se establecieron los siguientes requisitos funcionales:

1. Autenticación de los consejeros.
2. Subsistema para el emisor (interfaz para subir documentos, capturar eventos, etc.).
3. Interfaz para control de datos personales (captura y edición de datos personales).
4. Subsistema de autorización de documentos.
5. Subsistema para los receptores con sus respectivos privilegios.
6. Sistema de rotación de los contenidos publicados en el sitio Web.

2.1.2. Requisitos no Funcionales

Un requisito no funcional es aquel que *especifica propiedades del sistema, como restricciones del entorno o implementación, rendimiento, dependencias de la plataforma, mantenibilidad, extensibilidad o fiabilidad*.[5]

De acuerdo con la misma delimitación del sistema, se establecieron los siguientes requisitos no funcionales:

1. El sistema de comunicación interna se realizará vía Web.
2. El sistema de comunicación interna y el sitio Web, son en realidad el mismo sistema, con una parte pública y una privada.

---

[2] Jacobson, Ivar; Booch, Grady; Rumbaugh, James; *El Proceso Unificado de Desarrollo de Software*; Addison-Wesley; Edición en Español; Madrid 2000; Glosario General, p. 432.
[3] *Ibid*.
[4] Ver la sección 1.1.2: *Ámbito del Negocio: Necesidades Expresadas por el Cliente*.
[5] *Ibid*.

3. Se debe buscar un servidor que de hospedaje (WebHosting) y registro de dominio para el sitio.
4. Se deberá realizar al menor costo posible.
5. El sistema debe tener la capacidad de ser mantenido por personal sin conocimientos en computación (mantenimiento del contenido vía Web).

2.2. SOLUCIÓN AL PROBLEMA DE WEB HOSTING: IMPLICACIONES EN LOS REQUISITOS DEL SISTEMA.

2.2.1. Criterios de selección

Los criterios de selección se dividen en: requisitos mínimos y características adicionales; para que pueda operar el sistema se requiere de parte del servicio Web Hosting lo siguiente:

1. Que tenga soporte de base de datos (MySQL, MSSQL, PostgreSQL, etc.)
2. Que tenga soporte para lenguaje de programación de páginas dinámicas *server-side* (cgi, Perl, PHP, ASP, JSP, etc.)
3. Que tenga suficiente espacio para almacenar la información (mínimo 100MB)

Además de estos criterios mínimos de funcionamiento, están otros criterios adicionales, los cuales son:

1. Menor costo posible, porque el cliente dispone de poco capital.
2. Mantenimiento 24x7.
3. Servicio de respaldo de información y respaldo de energía.
4. Soporte técnico.
5. Acceso vía FTP o SSH.

2.2.2. Características del servicio contratado

El servicio contratado fue www.suempresa.com, los cuales ofrecen el siguiente plan:

TABLA 2.2.2-1: Plan de donativo (no se cobra el hospedaje, sólo el dominio)

CARACTERÍSTICA	CANTIDAD
Buzones de correo bajo su dominio	Ilimitados
Espacio en disco duro (MB)	50
Transferencia mensual incluida (GB)	20
Base de Datos (MySQL)	2
Dominios soportados	1

Además, todos los planes incluyen:

TABLA 2.2.2-2: Servicios incluidos en todos los planes

SERVICIO
Buzones de correo accesibles vía Pop3/Smtp/Web
Webmail (Lector de correo vía web)
Redireccionamientos de correo ilimitados (e-mail forwardings)
Cuenta de correo Catch All
Autorespondedores de E-mail
Grupos de correo bajo su dominio
Panel de control seguro y en tiempo real
Estadísticas detalladas de visitas
Acceso a su sitio por FTP las 24 hrs. del día
Administrador de archivos via web
Directorios web protegidos con contraseña (.htaccess)
SSL (https:// - No incluye certificado)
Disponibilidad para creación de Subdirectorios
Disponibilidad de Web Forwarding
Páginas de error personalizables

SERVICIO
PHP, Perl 5.x, Mod_Perl, Python, CGI, Apache-ASP (No compatible con el ASP de MS)
SSI (Server side includes)
Directorio CGI particular
Extensiones de Front Page versión 2002
Herramienta de administracion de bases de datos (PhpMyAdmin)
Compatibilidad con Dreamweaver, Flash y Shockwave
Soporte de archivos Flash, Shockwave, VRML, Javascript, XML, Midi, MP3 y más
Dominio servido por nuestro DNS
Respaldo y restauración de la información de su sitio desde su Panel de Control
Reporte de tráfico Reporte de uso de espacio
Accesibilidad a su sitio con o sin "www"
Acceso a archivos de bitácora (logs) de errores
Acceso a archivos de bitácora (logs) de visitas
Ancho de banda libre sobre demanda (hits ilimitados)
Apache Web Server
Infraestructura y equipos de alta seguridad y disponibilidad
Linux Red Hat
Monitoreo 24x7
Respaldos dobles diarios
Respaldos semanales y mensuales completos en cinta magnética
Soporte Técnico bilingüe vía e-mail o telefónica.
Sistema de respaldo de energía y estaciones eléctricas de emergencia a Diesel.

El costo para dos años es de $399 más IVA por el registro de dominio: www.cmdh.org. Se pidió además, un deducible de impuestos por $3499.

Este servicio fue elegido por lo siguiente:

1. Cumplía sobradamente con los requisitos.
2. Nadie más ofrecía este servicio a un costo tan bajo.

2.2.3. Implicaciones en el desarrollo del sistema

Las implicaciones en el desarrollo del sistema, son, principalmente, las siguientes:

En base de datos, MySQL, en lenguaje de programación PHP.

Otra restricción, es limitar el sistema a dos bases de datos como máximo y a ajustarlo al espacio concedido en disco: 500 MB como máximo. En capítulos posteriores se discutirá la estrategia de administrar el espacio. Adelantamos aquí que el espacio de 500MB es suficiente para almacenar la información del cliente durante un año y medio o dos aproximadamente sin necesidad de eliminar nada del servidor.

No se puede utilizar SSH o telnet para mantener el sitio, pero sí FTP o el *panel de control*. Para la base de datos debe utilizarse el phpMyAdmin que tienen ellos instalada, de donde, la construcción de la misma se hace mediante *scripts* que se colocan en el servidor y se ejecutan.

2.3. FASE DE INICIO: ESTUDIO DE LA FACTIBILIDAD, PLANEACIÓN BÁSICA Y ESBOZO DE LA ARQUITECTURA

2.3.1. Iteración de la fase de inicio

La fase de inicio tiene como propósito principal obtener la aprobación del producto por parte del cliente, adicionalmente se encuentran los siguientes objetivos:[1]

1. Entender lo que va a hacer el sistema y sus principales funciones
2. Ver si el sistema es útil al cliente
3. Identificar al menos una solución posible
4. Planear el proyecto, los costos y el calendario
5. Decidir los procesos y las herramientas a utilizar

Durante esta fase se realizaron las siguientes actividades:

1. Se formuló la hipótesis para incluir el Diagrama Entidad-Relación como una extensión a los modelos UML
2. Se hizo un esbozo de la arquitectura
3. Se gestionaron los riesgos
   Nota: el esbozo de arquitectura y la gestión de los riesgos se realizaron en paralelo y ambas actividades fueron mutuamente influyentes.
4. Se elaboró el documento de visión

2.3.1.1. Hipótesis para incluir el Diagrama Entidad-Relación

Creemos que es conveniente utilizarlo para facilitar el proceso de desarrollo.

2.3.1.2. Esbozo de la arquitectura

Se hizo con la doble finalidad de determinar lo que debe de realizar el sistema y para tratar de identificar al menos una solución posible.

Los flujos de trabajo fueron: requisitos, análisis y diseño; centrando la atención en los requisitos; en cuanto al análisis y al diseño sólo es un esbozo para identificar si existe al menos una solución.

2.3.1.3. Gestión de riesgos

Se identificaron los principales riesgos en torno a la justificación del sistema, su alcance y conveniencia; también se realizaron acciones para prevenir los riesgos y lograr la puesta en marcha del proyecto.

2.3.1.4. Documento de Visión

El documento de Visión es una síntesis de los requisitos, entorno, alcance, perfil del usuario, etc. que permite determinar si la realización de un proyecto está justificada o no.

Las actividades anteriores (formulación de hipótesis, esbozo de la arquitectura, gestión de riesgos) tienen como meta la redacción de este documento.

---

[1] Kroll, P. y Kruchten, P.: 2003. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison-Wesley. Capítulo 6

2.3.2. Extensión de los Modelos: Añadiendo el Diagrama Entidad-Relación

El Diagrama Entidad – Relación (DER) es una parte fundamental del análisis de sistemas de bases de datos tradicionales.[2]

Proponemos la siguiente hipótesis a comprobar mediante el desarrollo del sistema:

Hipótesis principal de DER: Aunque el DER no forma parte de RUP ni de UML, resulta conveniente incluirlo.

De comprobarse afirmativa la hipótesis se podrá responder a la siguiente pregunta ¿en qué parte del desarrollo del sistema ha de incluirse y para qué?

Para apoyar esta hipótesis proponemos a su vez otras que también serán verificadas en el desarrollo del sistema:

Hipótesis 1: Para este proyecto conviene colocar el Diagrama Entidad-Relación (DER) en una posición intermedia entre el análisis y el diseño, ya que contiene elementos de análisis (comprensión de requisitos y comprensión de la realidad) y una parte de diseño (solución al problema, prototipos para construir clases de diseño, etc.).

UBICACIÓN DEL *DER* EN EL DESARROLLO DEL SOFTWARE

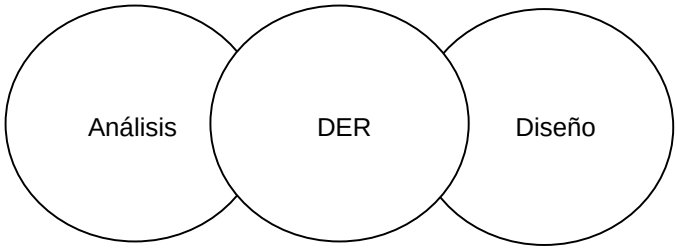

Fig. 2.3.1-1: Ubicación del DER *en el desarrollo del software*

Las tres principales ventajas para utilizar el DER son:

1. Permite comprender mejor la realidad a tratar, en particular, la estructura de la información y los datos persistentes con que el sistema trabaja (análisis). **Hipótesis 2:** Esta comprensión de la realidad se facilitará más si se trata primero mediante un DER que mediante UML.
2. **Hipótesis 3:** Permite pasar del análisis al diseño con mayor facilidad que directamente de UML, debido a su posición intermedia; y pueden construirse clases de diseño con sus métodos y propiedades basados en los atributos de las entidades (DER) y de los requisitos (realizaciones de los casos de uso).
3. **Hecho:** Es un estándar que dadas las *técnicas de cuarta generación* (T4G) permite la construcción inmediata del código para la base de datos a utilizar; en este caso MySQL.

Criterio: La notación a utilizar (e.g. Oracle, Crows Foot)[3] dependerá del software utilizado para realizar los diagramas (e.g. Erwin, DBDesigner) y el programa para realizar los diagramas dependerá de dos cosas: a) La facilidad para manipular los diagramas y b) La posibilidad de generar código compatible con MySQL.

---

[2] Chen, Peter Pin-Shan; *The Entity-Relationship Model - Towards a Unified View of Data*; ACM Transactions on Database Systems; March 1976. Citado en http://citeseer.csail.mit.edu/context/76/0

[3] Para ver una comparación entre las distintas notaciones de bases de datos consulte http://cisnet.baruch.cuny.edu/holowczak/classes/9440/entityrelationship/

La fase de inicio es una etapa muy temprana para definir el programa gestor de diagramas DER definitivo, sin embargo, se tienen algunos candidatos.

En el capítulo de las conclusiones evaluaremos la validez de las hipótesis.

2.3.3. Análisis de Riesgos y Factibilidad

2.3.3.1. Introducción

Este documento contiene, los riesgos identificados en esta fase de desarrollo y los criterios que se siguieron tanto en la elaboración de la lista de los riesgos como en la toma de las decisiones.

2.3.3.2. Criterios

2.3.3.2.1. Criterio de formulación de la tabla

La tabla consta de 7 campos: *probabilidad*, *impacto*, *P X I*, *riesgo* (*nombre*), *consecuencia*, *estado (notas)*, *contingencia*:[4]

> La *probabilidad* y el *impacto* son apreciaciones subjetivas, su escala es de 0 a 1. *P X I* es el producto de la *probabilidad* y el *impacto*.[5]
>
> La *probabilidad* indica que tan factible es que suceda el riesgo, y el *impacto* es cuánto abarca del proyecto o la gravedad del riesgo.
>
> *Riesgo* es la amenaza que se quiere prevenir. *Consecuencia* son los efectos del riesgo en caso de verificarse.
>
> *Estado (notas*) es el estado del riesgo y notas adicionales al respecto. La *contingencia* es la acción a realizarse en caso de que el riesgo se materialice.[6]

En esta etapa de desarrollo por ser breve se ha hecho una simplificación en el sentido de simplemente indicar de manera informal el estado del riesgo.[7]

El motivo por el que se eligió esta forma es la simplicidad, ya que estamos en la fase de inicio y se requieren elementos simples pero efectivos y el tiempo es un elemento valioso que no puede perderse.

2.3.3.2.2. Criterio de los renglones

Los renglones están acomodados de riesgo más delicado al menos delicado (o ya resuelto), con el objetivo de atender primero el riesgo más urgente o bien el más importante.

2.3.3.2.3. Criterio del uso

1. Atender los riesgos no resueltos considerando su prioridad.
2. Mantener la lista de los riesgos al día.
3. Aprender de los riesgos.

---

[4] Para una referencia completa sobre lo que el análisis de riesgos véase *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*; 2000 Edition; Project Management Institute, Newton Square, Pennsylvania USA; Capítulo 11.
[5] *Ibid.*
[6] *Ibid.*
[7] La definición formal de *estado* de un riesgo y sus valores en *PMBOK Guide;* capítulo 11.

2.3.3.3. Lista de Riesgos

TABLA 2.3.3-1: Lista de Riesgos

Claves: P = probabilidad. **I** = impacto. **P x I** = probabilidad x impacto.

P	I	P X I	RIESGO	CONSECUENCIA	ESTADO Y NOTAS	CONTINGENCIA.
0.9	0.8	0.72	No obtener el estudio de mercado Web.	No comprender los requisitos del sistema como sitio Web, ofrecer un sitio Web ineficaz para transmitir la imagen del cliente y captar y retener la audiencia querida por el cliente.	**No controlado**: depende de terceros, en caso de no obtenerse en el corto plazo.	Negociar con el cliente las posibles alternativas
0.9	0.7	0.63	Incremento sin control de documentos y artefactos.	Se pierde la referencia y la congruencia interna del proyecto, así como la medida del avance. Provoca retrasos en el proyecto.	**No controlado:** Se sugiere una revisión de los documentos nuevos y su inclusión jerárquica, su paginación y su acomodo en la agenda para su seguimiento.	Buscar asesoría
0.5	0.7	0.35	Carecer de arquitectura	No comprender los requisitos del cliente. No resolver las necesidades del cliente No estar en capacidad tecnológica de hacer el sistema.	**En proceso:** para eso la fase de inicio esboza la arquitectura y la fase de elaboración construye la arquitectura.	Buscar asesoría
0.5	0.7	0.35	Poco tiempo e información para tener una buena arquitectura.	No poder realizar la arquitectura del sistema. Fase de inicio: esbozar.	**Controlado:** Para la fase de inicio sólo es necesario esbozarla, por tanto se utilizarán patrones y modelos basados en experiencias pasadas y se adaptarán al caso presente con sus respectivas notas.	Buscar asesoría
0.4	0.8	0.32	Programas con márgenes muy cortos de tiempo.	Retrasos en programa debido a eventualidades.	**Controlado:** Modificar y ajustar los planes.	Buscar asesoría. Negociar las fechas de entrega

P	I	P X I	RIESGO	CONSECUENCIA	ESTADO Y NOTAS	CONTINGENCIA
0.4	0.8	0.32	Artefactos y documentos incompletos.	Proyecto incompleto. Colapso por ambigüedades.	**Controlado:** Agenda de pendientes. Planes de fases y artefactos. Control de versiones y revisiones.	Establecer un límite en la profundidad de los documentos y artefactos y sacrificar contenido
0.4	0.8	0.32	Desconocer los artefactos que el proyecto requiere producir.	No poder producir los artefactos necesarios. Retrasos. Colapso del proyecto.	**Controlado:** en el plan está contemplada la investigación de cada artefacto.	Buscar asesoría
0.2	0.8	0.16	Desconocer los riesgos del sistema.	Tener un colapso a la mitad o en etapas posteriores que lleven al retraso indefinido o la cancelación del proyecto después de un gran desgaste.	**Controlado:** la primera versión es esta lista, el análisis de los objetivos de la fase de inicio delimita el alcance de este análisis a los riesgos más críticos.	Buscar asesoría
0.1	1	0.1	Desconocer los límites de lo que hace el sistema y lo que no hace.	No poder realizar un plan, no poder realizar una arquitectura, desgaste, cancelación del proyecto.	Controlado al 90% falta el estudio de mercado Web.	Entrevistarse con el cliente
0.1	1	0.1	Que el proyecto no resuelva las necesidades del cliente.	Que todo el esfuerzo realizado no sirva para nada.	Controlado al 90% falta el estudio de mercado Web, lo demás está documentado. Hay un borrador de casos de uso.	Buscar otra solución mediante sistemas de terceros.
0.1	0.4	0.04	Criterios en la tabla de riesgos inadecuados.	Confusión en el manejo de los riesgos y ambigüedad en el documento.	Controlado: al final del día analizar la lista y actualizar criterios.	Investigar en fuentes especializadas.
0	0.6	0	Cambios en la lista de riesgos sin control.	Perder la noción del proceso de desarrollo. Entrega de documento de riesgos de mala calidad. Mal entendimiento de los riesgos en etapas posteriores.	Resuelto: se creó un archivo de control de cambios tipo changelog.	Iniciar una nueva lista de riesgos.
0	0.8	0	Carecer de herramientas de trabajo.	No poder producir los artefactos necesarios. Retrasos. Colapso del proyecto.	Resuelto: Rational Rose 2002.	Buscar herramientas parciales.
0	0.9	0	No comprender los objetivos de cada fase.	Hacer mal los planes, o muy simples o muy complicados.	Resuelto: Para la fase de inicio se simplificó el plan y se ajustó en la versión 2 a justificar la puesta en marcha del proyecto restringiendo los artefactos a su versión de borrador apoyados en documentos.	Buscar asesoría

2.3.3.2.4. Criterio de evolución

Cada fase tiene su lista de riesgos, la cual podrá incluir los riesgos de una fase anterior según las necesidades de desarrollo, los criterios y el modelo podrán variar según se requiera.

2.3.4. Esbozo de la Arquitectura: Selección de Patrones

En este apartado se esboza la arquitectura, con el propósito de ilustrar que existe al menos una solución posible; no se pretende que ésta sea la arquitectura mejor ni que sea la definitiva, pues eso es el propósito de la fase siguiente.

Otro objetivo es presentar de una forma muy breve la noción de patrón y algunos patrones propuestos en la construcción del sistema.

Los patrones son formas de construcción de sistemas que han resultado útiles a lo largo del tiempo en diversos sistemas. Por ejemplo: la arquitectura cliente-servidor.

La arquitectura está conformada por varias vistas: de los casos de uso, de análisis, de diseño, etc.; para la fase de inicio en que lo más importante es delimitar las funciones del sistema, la vista de casos de uso incluye a todos los casos de uso encontrados.

Para este proyecto no es necesaria la vista de código ni de pruebas, entre otras cosas porque se carece de código que probar.

Por tanto la arquitectura contiene tres vistas: casos de uso, análisis y diseño.

2.3.4.1. Vista del Modelo de Casos de Uso

El modelo de casos de uso en la fase de inicio se utiliza para tres cosas:

1. Delimitar el proyecto.
2. Verificar que el sistema realice lo que el cliente quiere.
3. Es parte de la arquitectura, por tanto, verifica que se puede construir el sistema.

Por simplicidad se ha descompuesto en dos modelos para ilustrar mejor las cosas y son los siguientes:

1. Genealogía de actores: explica las características y relaciones de los distintos actores del sistema.
2. Modelo de todos los casos de uso: contiene todos los casos de uso para lograr comprender bien los requisitos.

2.3.4.1.1. Genealogía de los Actores

GENEALOGÍA DE ACTORES

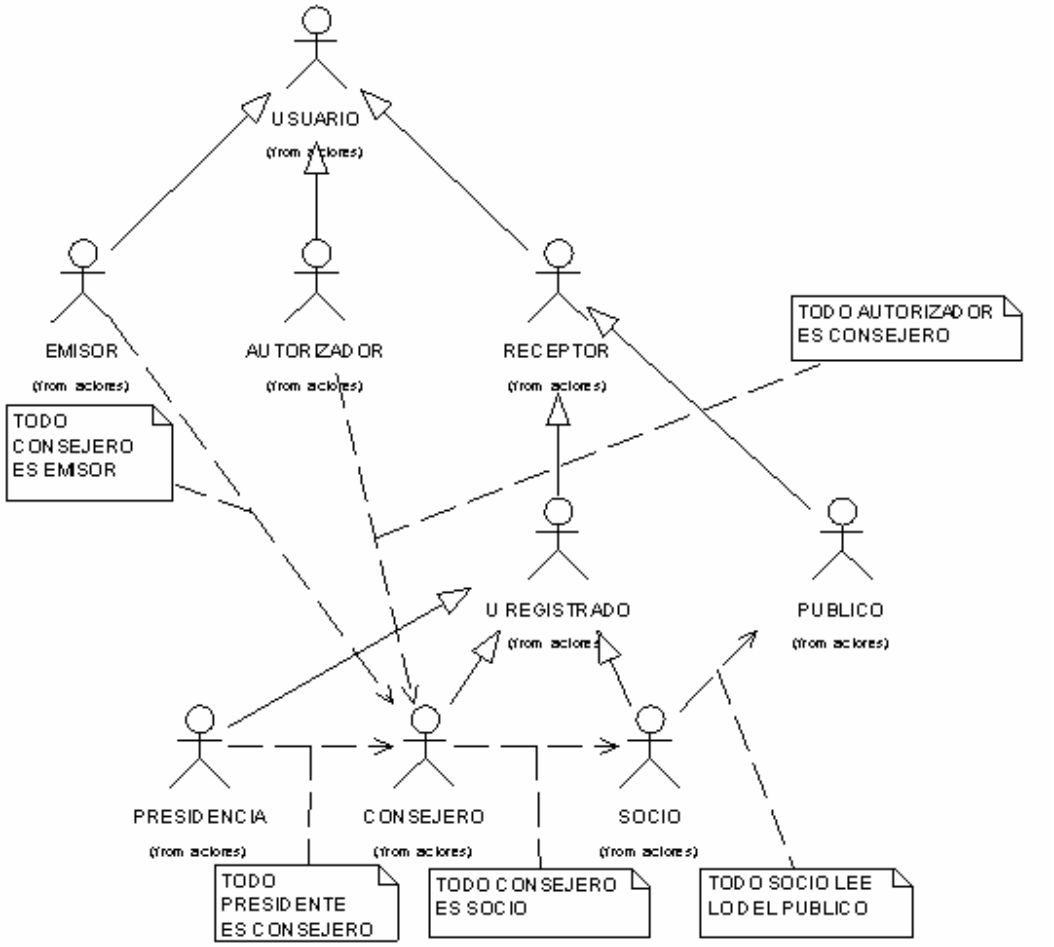

Fig. 2.3.4-1: Genealogía de Actores

Explicación de los actores:

- **USUARIO:** Usuario del sistema. Esta clase de actor se usa para asociar un caso de uso que es aplicable a todos los actores.
- **EMISOR:** Emite un documento para ser visto por los miembros de la CMDH.
- **AUTORIZADOR:** Autoriza o veta el documento enviado por el *emisor*. Puede cambiar el tipo de *receptor* restringiendo o ampliando el círculo de *receptores* de ese documento.
- **RECEPTOR:** Lee el documento, tiene distintos niveles de privilegios.
- **U REGISTRADO (USUARIO REGISTRADO):** Este actor es una simplificación para los diagramas de casos de uso en que se hace distinción entre usuarios registrados y no registrados. Todo usuario registrado posee privilegios y puede iniciar o finalizar una sesión.
- **PRESIDENTE:** Puede ver todos los documentos.
- **CONSEJERO:** Puede ver todos los documentos excepto los estrictamente reservados al presidente.
- **SOCIO:** Puede ver todos los documentos excepto los estrictamente reservados al presidente y a los consejeros.
- **PÚBLICO (USUARIO NO REGISTRADO):** Puede ver únicamente los documentos permitidos al público en general. Nota: el público puede dividirse en grupos de interés, como estudiantes, mujeres, periodistas, etc.

2.3.4.1.2. Modelo con todos los casos de uso

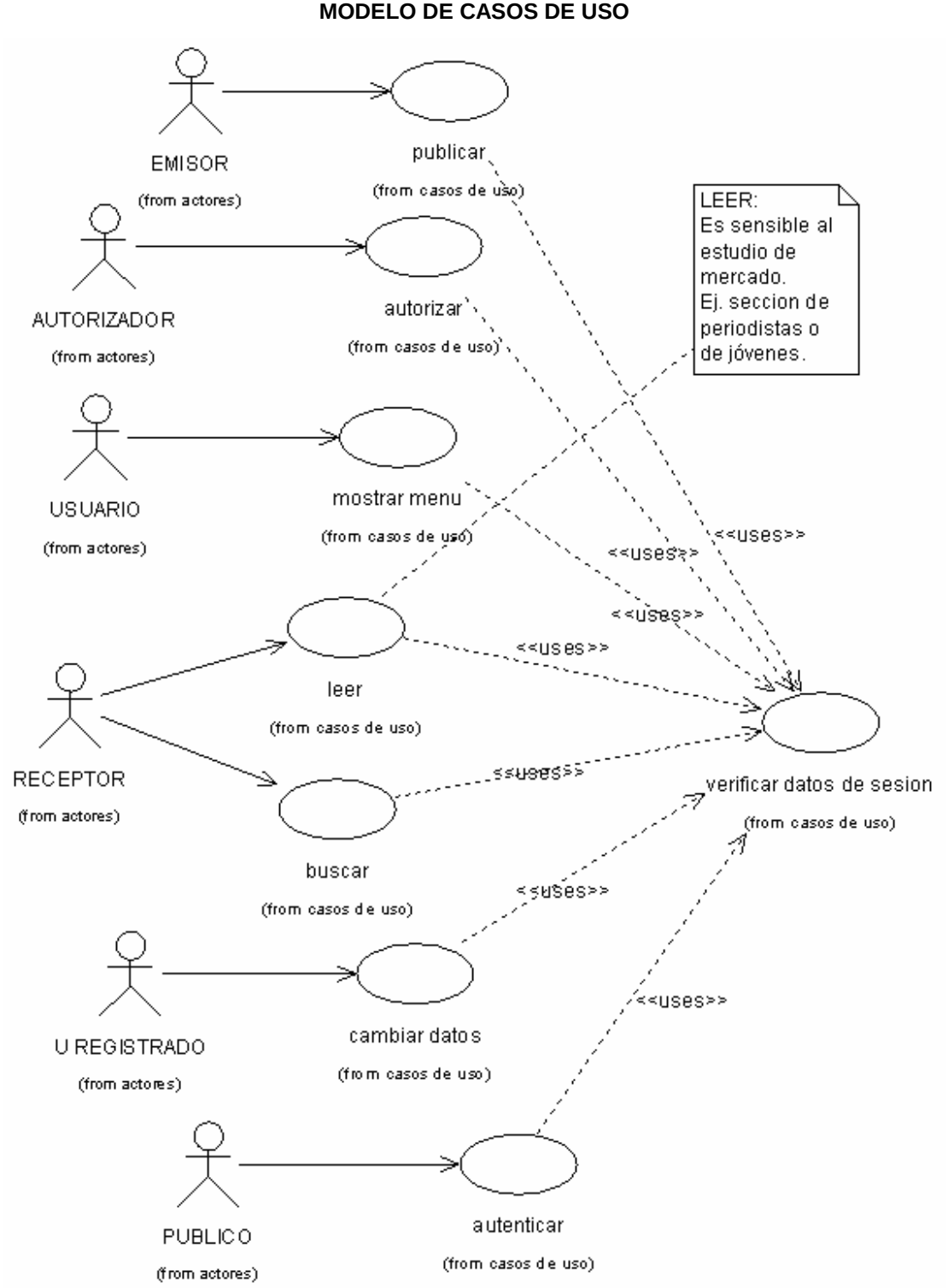

Fig. 2.3.4-2: Modelo de Casos de Uso

Explicación de los casos de uso:

- **publicar:** publica un documento en el sistema para que otros lo vean.
- **autorizar:** autoriza un documento para que otros lo vean y especifica/corrige quiénes lo pueden ver (esto debe discutirse en la fase de elaboración).
- **mostrar menú:** muestra el menú con las opciones que puede hacer el usuario, varía según el actor del sistema.
- **leer:** muestra los documentos publicados con los permisos correspondientes a cada tipo de usuario. Nota: este caso de uso es sensible a los estudios de mercado Web, por ejemplo, si hay una sección de periodistas o de jóvenes. También es afectado por las opciones de navegación en el sitio como la navegación por temas.
- **buscar:** busca un documento para leer, primero identifica al usuario para ver sus privilegios luego procesa las palabras de búsqueda y finalmente muestra la lista de documentos que pueden ser leídos.
- **cambiar datos:** cambia datos personales en la base de datos del sistema como el nombre, apellidos, etc.
- **autenticar:** mediante este caso de uso, un usuario registrado inicia su sesión y así es identificado por el sistema y se le reconocen sus privilegios.
- **verificar datos de sesión:** este caso de uso es usado para obtener los privilegios de un usuario y determinar las cosas que le esta permitido hacer en el sistema y las partes a las que tiene acceso.

2.3.4.2. Vista del Modelo de Análisis

Considerando los objetivos de la fase de inicio, la arquitectura, en su vista del modelo del diseño estará dada por las realizaciones de los casos de uso mediante clases de análisis.

Dado el riesgo del tiempo, se usará un patrón descubierto en experiencias pasadas,[8] trabajando con sistemas Web; lo hemos llamado *patrón genérico de la realización de casos de uso-análisis en sistemas Web*.

El diagrama del patrón genérico de análisis contiene las siguientes clases:

1. **capturador:** clase de análisis tipo interfaz, que captura los datos y peticiones del usuario y las envía para ser procesadas. E. g. mediante un formulario Web.
2. **validador:** clase de análisis tipo control que valida las peticiones de usuario para verificar que sean validas desde el punto de vista lógico.
3. **operador:** clase de análisis tipo control que procesa la petición del usuario realizando la lógica del negocio, comunicándose con los bancos de datos necesarios y/o con otros operadores.
4. **controlador (de flujo/de resultado):** clase de análisis tipo control que es responsable de controlar el flujo del programa de acuerdo a los resultados obtenidos en cada etapa y en cada clase, en particular procesa los resultados de una petición dada y los mensajes de error obtenidos.
5. **mostrador:** clase de análisis tipo interfaz que muestra los resultados obtenidos y/o los mensajes de error.

[8] En el servicio social en la DGSCA

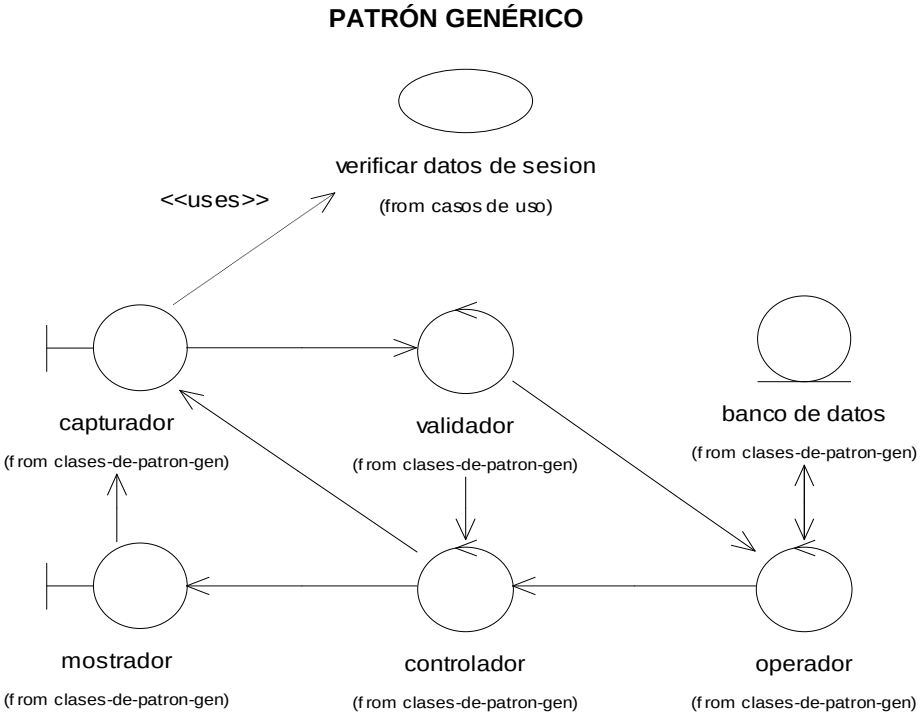

Fig. 2.3.4-3: Patrón genérico de realización de casos de uso-análisis en sistemas Web

La secuencia típica de este patrón es como sigue: *La clase capturador verifica los parámetros de la sesión del usuario y sus permisos de seguridad; el usuario introduce su petición mediante la interfaz capturador (e.g. un formulario), ésta, es validada por el validador, si la información es incongruente (e.g. formulario vacío) se envía un mensaje a la clase controlador para que tome una decisión (e.g. volver a pedir datos), en caso de que la información sea correcta, se pasa la petición (y los datos) a la clase operador para que ejecute la lógica del caso de uso (e.g. realizar una búsqueda); los resultados de la acción realizada y/o los mensajes de error son remitidos a la clase controlador para que tome la decisión de volver a pedir la información (pudiendo mostrar un mensaje con el resultado de la petición anterior) remitiendo a la clase capturador o bien mostrar los resultados sin volver a pedir la información (e.g. los resultados de la búsqueda o "error de sistema") remitiendo a la clase mostrador, la clase mostrador tiene la opción de volver a capturar datos y transfiere el control de la sesión a la clase capturador*.

Los casos de uso de este proyecto siguen este esquema con algunas variantes como a continuación se muestra, añadiendo ilustraciones donde sea necesario.

Los casos de uso *publicar* y *buscar* repiten el patrón de modo casi idéntico, únicamente que la clase *mostrador* después de publicar los resultados le transfiere el control a la clase *capturador* de forma automática para una nueva captura.

El caso de uso *leer* es una simplificación: las clases *capturador* y *mostrador* se fusionan y la petición del usuario es implícita (invisible pero real); se suprime la clase *validador* y las clases *operador* y *controlador* se fusionan en una sola llamada *filtro*. La secuencia es: aparece el caso de uso, automáticamente hace la petición que pasa por un *filtro* de seguridad, se obtienen los datos del banco de datos y se devuelven para ser mostrados.

El caso de uso *verificar datos de sesión* es otra simplificación: evidentemente la clase *mostrador* no invoca al caso de uso *verificar los datos de sesión* porque haría un proceso

infinito. Además esta clase es transparente para el usuario, ya que el caso de uso es invocado por otros casos de uso y no actores; además, la petición es muy concreta: *este usuario qué privilegios tiene* por tanto se suprime la clase *validador* por innecesaria; la clase *operador* y la clase *controlador* se fusionan en la clase *controlador de sesiones* la cual es capaz de iniciar, otorgar información y terminar una sesión, comunicándose con el *banco de datos de la sesión* que es una instancia o una clase heredada de la clase *banco de datos*.

El caso de uso *autorizar* es una ampliación: la clase *capturador* primero realiza una consulta directa al *banco de datos de documentos en espera* para mostrarlos al usuario y solicitarle los autorice o los rechace, posteriormente sigue el mismo patrón genérico.

El caso de uso *cambiar datos* es muy similar, primero obtiene todos los datos que van a ser modificados para mostrarlos, luego sigue el patrón genérico.

El caso de uso *mostrar menú* opera con la interfaz de *mostrador* (no existe *capturador* o se fusiona con *mostrador*) verificando los datos de sesión y verificando la ubicación en el sitio mediante la clase *operador*, los elementos a mostrar en un *banco de datos de menú* y presentándolos otra vez al *mostrador*.

El caso de uso *autenticar* es más complejo, debido a que la clase *operador* y la clase *banco de datos* se duplican, por tanto, se muestra a continuación este proceso.

REALIZACIÓN EN ANÁLISIS DEL CASO DE USO *INICIAR SESIÓN*

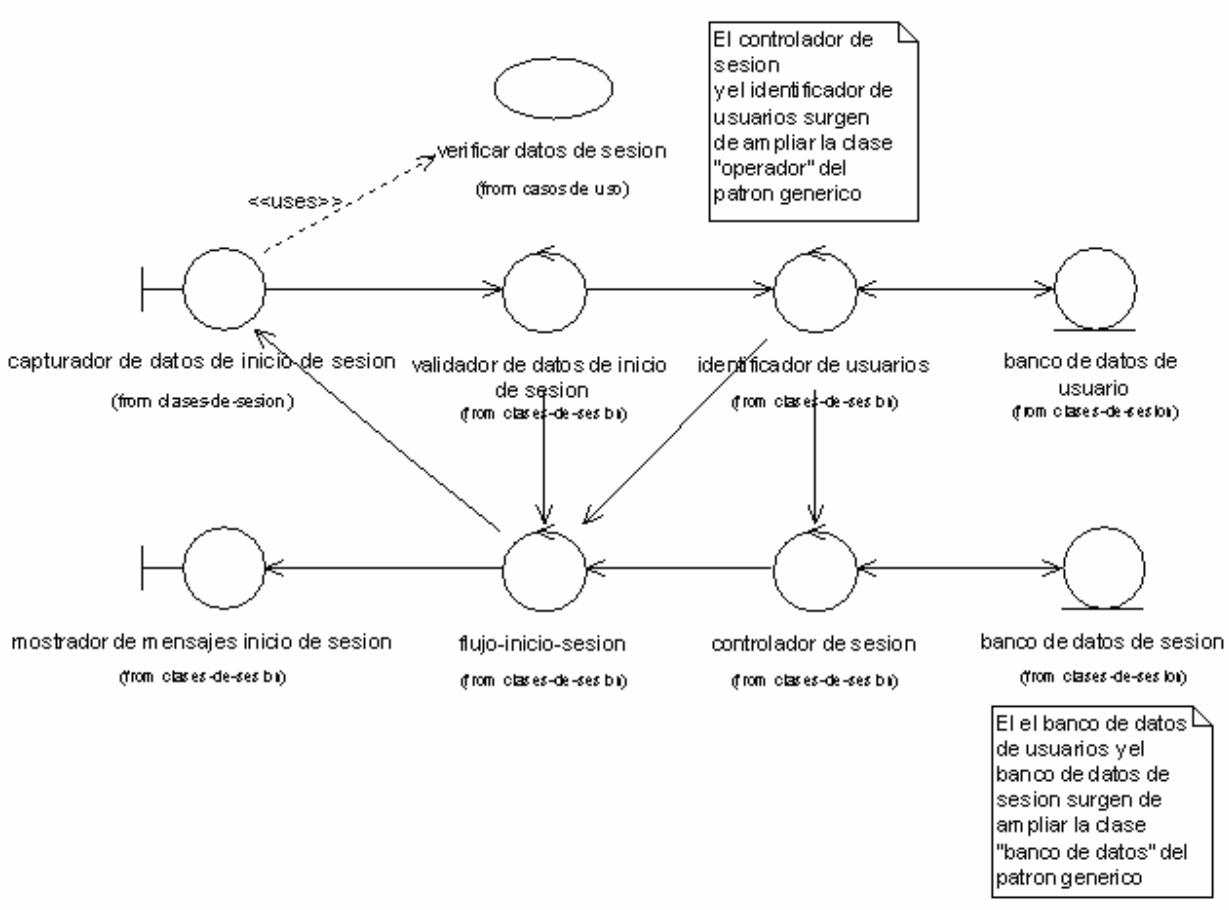

Fig. 2.3.4-4: Realización en análisis del caso de uso iniciar sesión

Con este patrón se han cubierto todos los casos de uso explicando las variantes que son necesarias de hacer, para la fase de inicio con esto es suficiente para mostrar desde el punto de vista del análisis que es posible construir las realizaciones de todos los casos de uso.

En la fase de elaboración se buscará la forma óptima de estas realizaciones y se detallarán los datos para construir el sistema y que funcione.

2.3.4.3. Vista del Modelo de Diseño

El diseño en la fase de inicio tiene como objetivo esbozar la arquitectura. La arquitectura se ilustra mediante los patrones y los subsistemas sin entrar en detalle en las clases y colaboraciones a menos que sea necesario.

Esto ocurre porque la fase de inicio centra su atención en el análisis del negocio, i.e. en si el sistema a construir va a beneficiar al cliente o no; por tanto, el aspecto de la arquitectura tiene como propósito mostrar la existencia de un camino hacia la solución de manera *ilustrativa*.

2.3.4.3.1. Patrones propuestos de diseño

Los patrones principales escogidos para el diseño son tres:

1. Patrón de capas (plano vertical).
2. Patrón de organización de datos (diagrama entidad-relación).
3. Patrón de navegación (plano horizontal).

2.3.4.3.2. Patrón de capas

El patrón de capas indica la estructura que va a tener el sistema para resolver las demandas del cliente desde una óptica de componentes. A partir de aquí se desprenden varias cosas como son los paquetes, subsistemas, etc.; así como otros patrones para problemas más específicos.

MODELO BASADO EN EL PATRÓN DE CAPAS

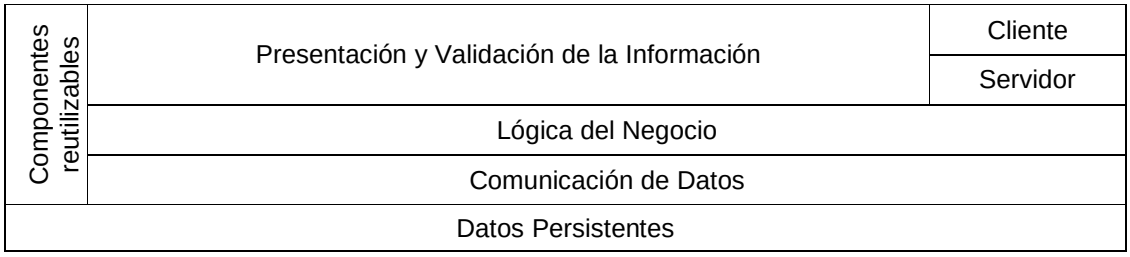

Fig. 2.3.4-5: Modelo basado en el patrón de capas

Explicación de las capas

La *capa de Datos Persistentes* contiene los componentes que almacenan los datos de manera permanente; usualmente es una base de datos. Y en el caso de este proyecto la mayor parte es una base de datos en MySQL.

La *capa de Comunicación de Datos* contiene una serie de componentes especiales para realizar consultas con la base de datos y para acceder a los datos persistentes de manera eficiente.

La *capa de la Lógica del Negocio* contiene los componentes y subsistemas encargados de realizar los casos de uso.

La *capa de Presentación y Validación de la Información* contiene los componentes necesarios para dar un formato adecuado a la información a ser presentada al usuario de acuerdo a las necesidades (XML, HTML, texto) y para validar la información introducida por el usuario en los formularios.

Esta capa se divide en dos: el lado del cliente (que contiene Javascript, DHTML, CSS, etc.) y el lado del servidor que contiene programas para dar el formato adecuado.

Un ejemplo de esta capa del lado del servidor, es cuando el cliente elige la opción *frameset*, entonces el sitio, los menús, etc., se muestran mediante *frames*, otro cliente elige la opción *versión para imprimir* entonces las cosas se muestran de forma tal que puedan ser enviadas a una impresora sin problemas; y así sucesivamente.

Finalmente hay una *capa lateral* que es el *repositorio de Componentes Reutilizables*, el cual va creciendo primero con componentes de proyectos anteriores y luego con componentes que realizan desarrolladores de PHP en todo el mundo.

Existe un sitio *open source* donde se generan 2 componentes al día y los para diversas cosas, desde bases de datos, hasta manejo de formularios, control de archivos, XML, etc.

2.3.4.3.3. Patrón de organización de datos

El patrón de organización de datos, que está expresado por el diagrama entidad-relación junto con algunas adiciones menores corresponde a la capa de datos persistentes y describe la forma en que está organizada la información.

El diagrama entidad-relación candidato realizado hasta la fase de inicio es el siguiente:

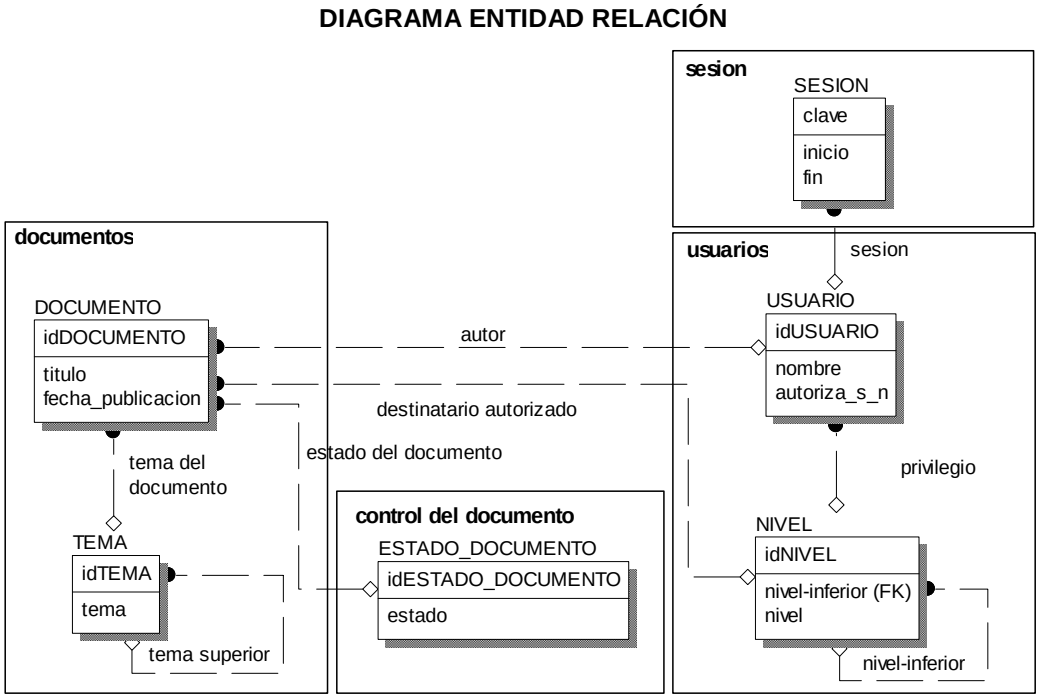

Fig. 2.3.4-6: Diagrama entidad-relación propuesto en la fase de inicio

Considérese provisional este diagrama y con fines ilustrativos.

Esta capa presenta algunas adiciones, entre otras el contenido estático del portal y los datos persistentes del lado del cliente (cookies).

2.3.4.3.4. Patrón de navegación

El patrón de navegación se apoya en la organización de la información, la cual va a tener una estructura de árbol; el árbol de navegación está soportado por los temas (del diagrama entidad-relación) y por los casos de uso cuando éstos se realizan.

Este patrón consiste en una máquina de estados a manera de un autómata, donde cada estado se define por diversas *variables de estado*.

Las variables de estado esbozadas son:

1. La *posición* indicada por el tema (o la clave del tema).
2. La *sesión* que indica los privilegios del usuario, etc.
3. La *acción* que es el caso de uso que realiza el usuario.
4. El *documento*: opcionalmente un documento a mostrar.

Con éstas variables se puede determinar el lugar en el *árbol* de la información y las opciones que cada usuario tiene en cada momento. Adicionalmente pueden existir otras variables, pero éstas deben ser consideradas parámetros específicos de un caso de uso en particular.

2.3.4.3.5. Subsistemas de la capa de lógica del negocio

A continuación se ilustran los subsistemas esbozados para la capa de la lógica del negocio.

SUBSISTEMAS DE LA CAPA *LÓGICA DEL NEGOCIO*

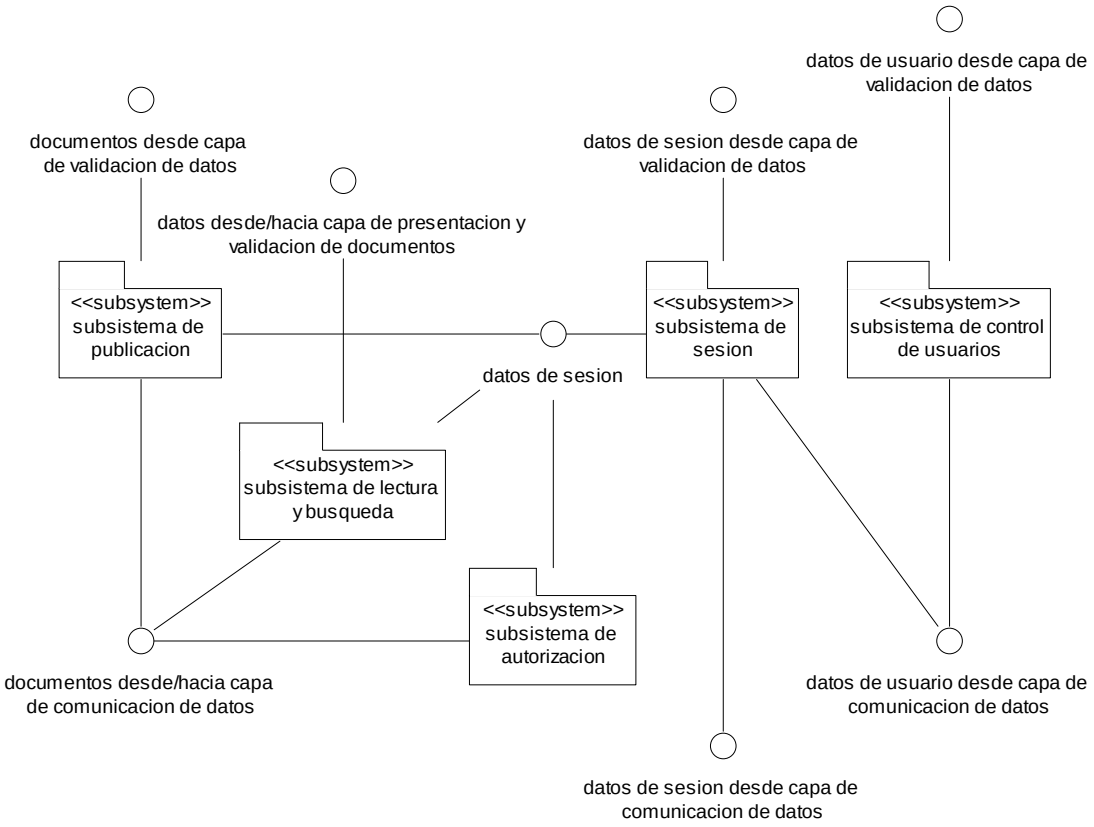

Fig. 2.3.4-7: Subsistemas de la capa de lógica del negocio

2.3.5. Documento de Visión

2.3.5.1. Introducción

En el entorno profesional se utilizan dos documentos para justificar un proyecto:[9]

1. El documento de visión.
2. El documento de especificación de requerimientos de software (SRS).

Para la fase de inicio se realizan estos documentos únicamente hasta obtener la aprobación del cliente. Estos documentos pueden irse completando y corrigiendo a lo largo de las fases hasta quedar listos al final o bien pueden no utilizarse dependiendo de la naturaleza del proyecto.

Por tanto la atención en esta fase se centra en el aspecto comercial y de los beneficios frente a los competidores.

Este proyecto tiene características muy particulares, entre otras, que la aprobación fue inmediata, por lo cual esta fase tiene como propósito esbozar los objetivos que fueron expuestos en el capítulo anterior, con algunas limitaciones y diferencias:

1. El costo del producto es meramente simbólico, ya que se trata de una organización no lucrativa a la cual se le está donando el sistema. Por tanto se omite el cálculo de los costos.

2. El documento de especificación de requerimientos (SRS) es fácilmente sustituible por dos documentos más sencillos: una lista de requisitos y un diagrama de casos de uso. La razón de esto, es que el sistema es sencillo y puede expresar los requerimientos de esta forma, además de que estos documentos ya se tienen; mientras que el SRS no se tiene y es desconocido.

Varias partes del documento de visión ya están desarrolladas en el presente trabajo de tesis; por tanto serán omitidas y en su lugar se anotará la referencia del lugar donde se encuentran.

2.3.5.1.1. Propósito del Documento de Visión

Este documento proporciona la visión del Portal de Control de Miembros y Rotación Automática de Contenidos que se desarrollará para el servicio de la Comisión Mexicana de Derechos Humanos, A. C. (CMDH)

Fundamental para el éxito de un proyecto de software es el documento de Visión, que identifica en un nivel alto y organiza las necesidades de los usuarios y las capacidades de una aplicación. Este documento se actualizará como sea necesario y se distribuirá entre los miembros del equipo de trabajo y demás personal involucrado.[10]

[9] Leffingwell, D. y Widrig, D.: 2000. *Managing Software Requirements: A Unified Approach*, Addison-Wesley. Apéndices B y C.
[10] *Ibid.*

2.3.5.1.2. Panorámica del producto

La Comisión Mexicana de Derechos Humanos A. C. es una organización no gubernamental que busca generar y promover una cultura de los derechos humanos en México y el mundo; procurando, a través de su estudio, difusión y defensa; el conocimiento, reconocimiento, vigencia y respeto de los mismos, basados en la dignidad de la persona humana.

Cabe señalar que esta asociación tiene presencia en varios estados al interior de la República por medio de consejeros, los cuales tienen un papel decisivo en la misma. Igualmente hay simpatizantes que tienen el derecho de participar en determinados eventos y de estar informados de ciertas cosas.

Dicha institución solicitó la elaboración de un portal de Internet en el cual publicar sus diferentes actividades para sus miembros, la prensa y el público en general. Esta agrupación necesita un medio mediante el cual llevar el control de los consejeros y simpatizantes y mantenerlos informados de las diferentes actividades. Asimismo los consejeros necesitan una forma sencilla de publicar sus opiniones al resto de la comunidad. Todo esto debe de pasar por medio de un control central que autorice los contenidos y administre los diferentes temas a tratar.

Se espera que el sistema pueda ser controlado al 100% a través de Internet y que el (los) administrador(es) del mismo puedan configurarlo sin necesidad de conocimientos técnicos especializados.

Uno de los requisitos pedidos por el cliente, fue la reducción al máximo de los costos, pues se encuentran escasos de fondo.

2.3.5.2. Descripción del usuario

2.3.5.2.1. Demografía de usuarios

Los usuarios se dividen en dos grupos: los usuarios internos y los usuarios externos. Los usuarios internos son los consejeros tanto del Consejo Consultivo como del Consejo Directivo.

Los usuarios externos son los visitantes de la página de Internet, que son tanto los socios y simpatizantes como todos aquellos que están interesados en los derechos humanos como se describe adelante.

En México hay aproximadamente 14 millones de usuarios de Internet,[11] no se espera una concurrencia de los 14 millones de usuarios, pero si que mensualmente haya algunas cientos de visitas a la semana.

2.3.5.2.2. Perfiles de usuario

Los usuarios del sistema son los consejeros, los socios, y el público.

El público se agrupa según el tipo de audiencia, que se puede clasificar de acuerdo a los grupos de interés, que podemos resumir en los siguientes:

- Jóvenes
- Mujeres
- Prensa

[11] Fuente: COFETEL, http://www.cofetel.gob.mx.

- Políticos
- Iniciativa privada
- Líderes sociales
- Organizaciones de la sociedad civil
- Organismos internacionales
- Instituciones educativas

Los cuales buscan determinado tipo de información en el sitio y de determinada forma.

2.3.5.2.3. Entorno del usuario

Usuarios internos

Entorno físico: están distribuidos por toda la república mexicana y se reúnen una vez cada tres meses. Su forma ordinaria de comunicación, además de las reuniones trimestrales, es el correo electrónico, el teléfono y el fax.

Entorno informático: utilizan máquinas Windows®, desde la versión 98 hasta la versión XP, con un navegador Internet Explorer® versiones 5 y 6. Las conexiones son por módem en la mayoría de los casos y no se tienen conocimientos de computación. Para los documentos suelen usar Microsoft™ Office en sus versiones que van de 97 a XP.

Usuarios del Web

Cada grupo tiene sus propios intereses y preocupaciones, pero todos tienen en común que en mayor o menor medida se interesan por los derechos humanos.

2.3.5.2.4. Principales necesidades del usuario

Ya se explicó en la sección *1.1.1.Necesidades expresadas por el cliente*, del presente trabajo de tesis.

2.3.5.2.5. Alternativas y competidores

Por el momento no hay otros competidores ya que no se está ofreciendo ningún pago de importancia por el desarrollo del sistema.

2.3.5.3. Descripción general del producto

2.3.5.3.1. Perspectiva del producto

El sistema desde el punto de vista interno tiene la perspectiva de ser una herramienta que facilite la comunicación de los miembros del consejo, haga más ágiles las reuniones y sea un factor de unidad entre los miembros de la CMDH.

Desde el punto de vista del sitio Web, se espera que sea una referencia para todos aquellos que se interesen en los derechos humanos y permita dar una buena imagen de la institución ante la opinión pública.

2.3.5.3.2. Posición del producto

La posición del producto es, a nivel interno, la de ser el medio por el cual se realice la comunicación oficial del consejo directivo en el tiempo que no se celebren las reuniones trimestrales.

*2.3.5.3.3. Resumen de rasgos (***features***)*

Sistema interno:

- Capaz de almacenar documentos.
- Capaz de autorizar/rechazar documentos.
- Capaz de establecer el tipo de receptor del documento.
- Capaz de mostrar los documentos de acuerdo al nivel de privilegio de lectura.
- Capaz de rotar los documentos de forma que el más nuevo aparezca hasta arriba.
- Capaz de borrar los documentos que se vayan indicando.

Sistema Web:

- Capaz de mostrar los documentos y páginas en un orden temático y mostrando el más nuevo hasta arriba.
- Capaz de hacer búsquedas en los encabezados y en los resúmenes de los documentos.

2.3.5.3.4. Supuestos y dependencias

Se decidió realizar el producto en Web, el proveedor del hospedaje ofrece PHP, MySQL y FTP. Por tanto, la construcción del sistema se atendrá a estas restricciones.

2.3.5.3.5. Costo y asignación de precio

Por tratarse de una organización no gubernamental y por considerarse este sistema un servicio a la sociedad el costo será meramente simbólico.

2.3.5.4. Rasgos del producto

Los rasgos del producto son descritos con mayor claridad en el subcapítulo *2.1.Primera entrevista con el cliente: obtención de los requisitos básicos* del presente trabajo de tesis.

2.3.5.5. Casos de uso

Los casos de uso están descritos en la sección *2.3.4.Fase de Inicio: Esbozo de la arquitectura* del presente trabajo de tesis.

2.3.5.6. Otros requerimientos del producto

2.3.5.6.1. Estándares aplicables

Unified Modeling Language (UML)[12]

2.3.5.6.2. Requerimientos del sistema

Los requerimientos del sistema están descritos en el subcapítulo *2.1.Primera entrevista con el cliente: obtención de los requisitos básicos* del presente trabajo de tesis.

2.3.5.6.3. Licencias e instalación

GNU General Public License[13]

[12] http://www.uml.org/
[13] http://www.gnu.org/copyleft/gpl.html

2.3.5.6.4. Requerimientos de desempeño

No se han especificado aún.

2.3.5.7. Requerimientos de documentación

2.3.5.7.1. Manual de usuario

Debe ser fácil de comprender por personas que no tienen conocimientos informáticos.

2.3.5.7.2. Ayuda en línea

El sistema debe tener ayuda en la Web y la posibilidad de enviar correo electrónico al (los) administrador(es) del sitio para preguntar dudas acerca del uso del sistema.

2.3.5.7.3. Guías de instalación, configuración y archivo léeme

El sistema va a poseer manuales impresos de instalación, configuración, *léeme*, para su uso correcto.

2.3.5.7.4. Etiquetas y empaque

El sistema carece de etiquetas, ya que se instala directamente en el servidor.

2.3.5.8. Requerimientos del proyecto

El método a elegir debe ser uno que ofrezca la solución de manera sencilla y económica, entendiendo económica tanto en dinero como en tiempo.

Se eligió el proceso unificado de desarrollo, ya que ofrece un lenguaje muy flexible (UML) y está basado en más de 30 años de experiencia. Además, permite que el cliente introduzca cambios en los requisitos en cualquier etapa de desarrollo y poder entregar los resultados a tiempo.

2.3.5.8.1. Características del proceso del proyecto

Vea la sección *1.2.2.El Proceso Unificado: Descripción* del presente trabajo para más información.

2.3.5.8.2. Análisis de riesgos en el proyecto

Esta parte se encuentra en la sección: *2.3.3.Fase de Inicio: Análisis de riesgos y factibilidad* del presente trabajo de tesis.

2.4. FASE DE ELABORACIÓN

2.4.1. Descripción de la Arquitectura

2.4.1.1. Introducción

Por arquitectura entendemos «*la estructura o las estructuras del sistema que comprenden los elementos del software, las características externamente visibles de estos elementos y las relaciones entre ellos*».[1]

Esta definición implica cosas muy importantes, entre otras el hecho de que los elementos se van a especificar desde un punto de vista externo y en la medida en que afecten a todo el sistema.

La descripción de la arquitectura está formada por los siguientes artefactos:

- Modelo de casos de uso
- Modelo de análisis
- Diagrama entidad-relación (no forma parte del estándar de RUP ni UML)
- Modelo de diseño
- Modelo de distribución
- Modelo de implementación

A continuación se presentan estos artefactos con sus notas respectivas de acuerdo a la forma que adquirieron al final de la fase de elaboración.

Posteriormente se describirán brevemente las dos fases que formaron el proceso de desarrollo mediante el cual se obtuvo esta versión de la arquitectura, así como los riesgos, pruebas, criterios y decisiones tomadas durante el proceso.

Finalmente se añaden los documentos y estándares generados durante el proceso y que sirven para describir los protocolos y formatos de los datos, así como su semántica.

[1] Bass, Clements, Kazman; *Software Architecture in Practice (2nd edition)*; Addison-Wesley 2003. Se recomienda visitor la página http://www.sei.cmu.edu/architecture/definitions.html donde se discuten las diferentes definiciones de arquitectura de software.

2.4.1.2. Modelo de Casos de Uso

El modelo de caso de uso tiene dos diagramas, el primero explica a los *actores* desde un punto de vista conceptual, el segundo es el *diagrama de casos de uso* significativos para la arquitectura.

DIAGRAMA PARA EXPLICAR LOS ACTORES

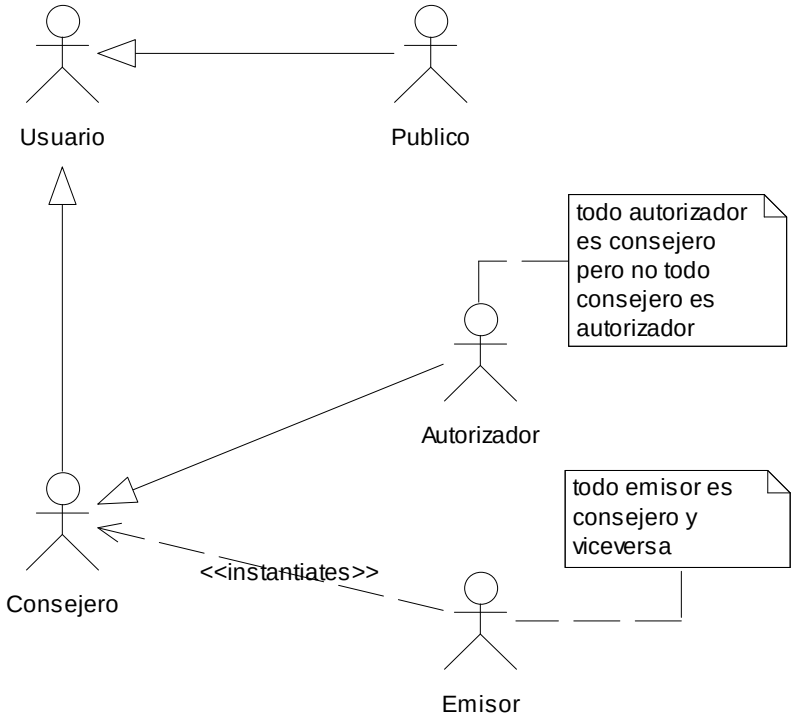

Fig. 2.4.1-1: Diagrama de actores del sistema

Explicación:

- **Usuario:** el usuario es el actor más general que existe y engloba, por decirlo de una forma a todos los demás.
- **Público:** se refiere a todos los actores de los que no se tiene una cuenta en el sistema o que no han iniciado sesión.
- **Consejero:** son los consejeros directivos o consultivos, los cuales poseen una cuenta en el sistema.
- **Autorizador:** es un consejero con privilegios especiales para administrar la información del portal.
- **Emisor:** es el rol que cumple cualquier consejero cuando emite (publica) un documento.

DIAGRAMA DE CASOS DE USO

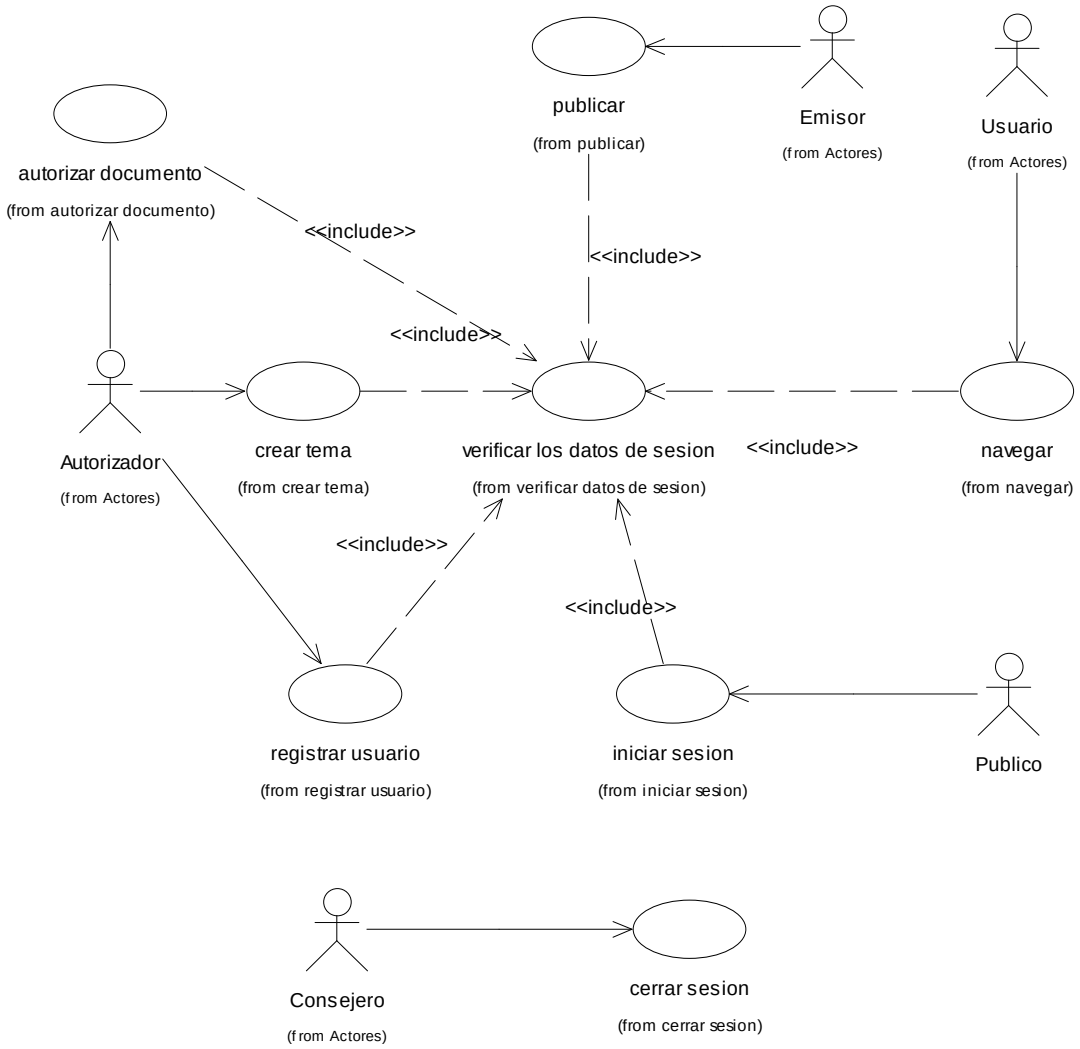

Fig. 2.4.1-2. Diagrama de casos de uso significativos para la arquitectura

Explicación:

- **Publicar:** el emisor envía un documento al sistema, el sistema revisa la información y coloca el documento en la base de datos en estado de pendiente esperando ser autorizado.
- **Autorizar documento:** el autorizador revisa los documentos publicados y autoriza o rechaza los mismos, asignándoles la categoría de públicos o privados.
- **Crear tema:** los documentos están agrupados en temas, estos temas los crea el autorizador.
- **Registrar (nuevo) usuario:** los usuarios deben ser estrictamente consejeros, el método más sencillo es que el autorizador los registre; una vez que se capturan los datos, se envía por correo el *login* y *password* al usuario (consejero) que fue dado de alta en el sistema.

Iniciar sesión: cuando cualquier usuario ingresa al sistema, de entrada es tratado como público hasta que inicia la sesión, una forma es mediante el inicio de sesión por *login* y *password*.

- **Navegar:** consiste en obtener la información del sitio, como los documentos (contenido) y los temas en que se agrupan (menú de navegación). Este caso de uso debe considerar la seguridad del sistema.
- **Iniciar sesión:** un usuario inicia su sesión mediante un nombre (*login*) y una contraseña (*password*).
- **Cerrar sesión:** todo usuario con sesión activa podrá cerrar sesión, entonces el sistema lo tratará como público.
- **Verificar los datos de sesión:** verifica que la sesión sea válida.

2.4.1.3. Modelo de Análisis

El modelo de análisis está integrado por los siguientes paquetes y dependencias:

PAQUETES DEL MODELO DE ANÁLISIS

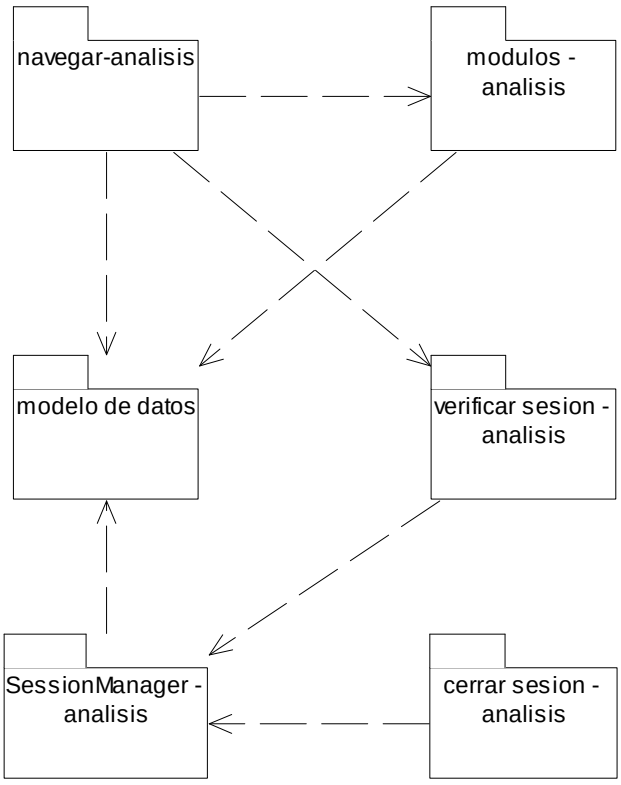

Fig. 2.4.1-3. Paquetes del modelo de análisis

Explicación breve de los paquetes:

- **Modelo de datos:** contiene los objetos persistentes, este modelo está directamente vinculado con el Diagrama Entidad-Relación que se muestra adelante.
- **SessionManager – análisis:** contiene la vista de análisis de las clases que controlan las sesiones de usuario.
- **Cerrar-sesión – análisis:** explica la realización del caso de uso: cerrar sesión desde el punto de vista del análisis.

- **Verificar sesión – análisis:** explica la realización del caso de uso: verificar los datos de sesión desde el punto de vista del análisis.
- **Módulos – análisis:** contiene el análisis de las realizaciones de casos de uso.
- **Navegar – análisis:** contiene el análisis de la navegación (el concepto se define más adelante).

2.4.1.3.1. Especificación de cada paquete

Modelo de datos

Contiene un diagrama de clases de tipo entidad, cuya característica es que son persistentes. Este modelo se ha construido con el propósito de investigar y elaborar la arquitectura, las normalizaciones menores se omitieron y la versión no es definitiva.

MODELO DE DATOS PERSISTENTES EN ANÁLISIS

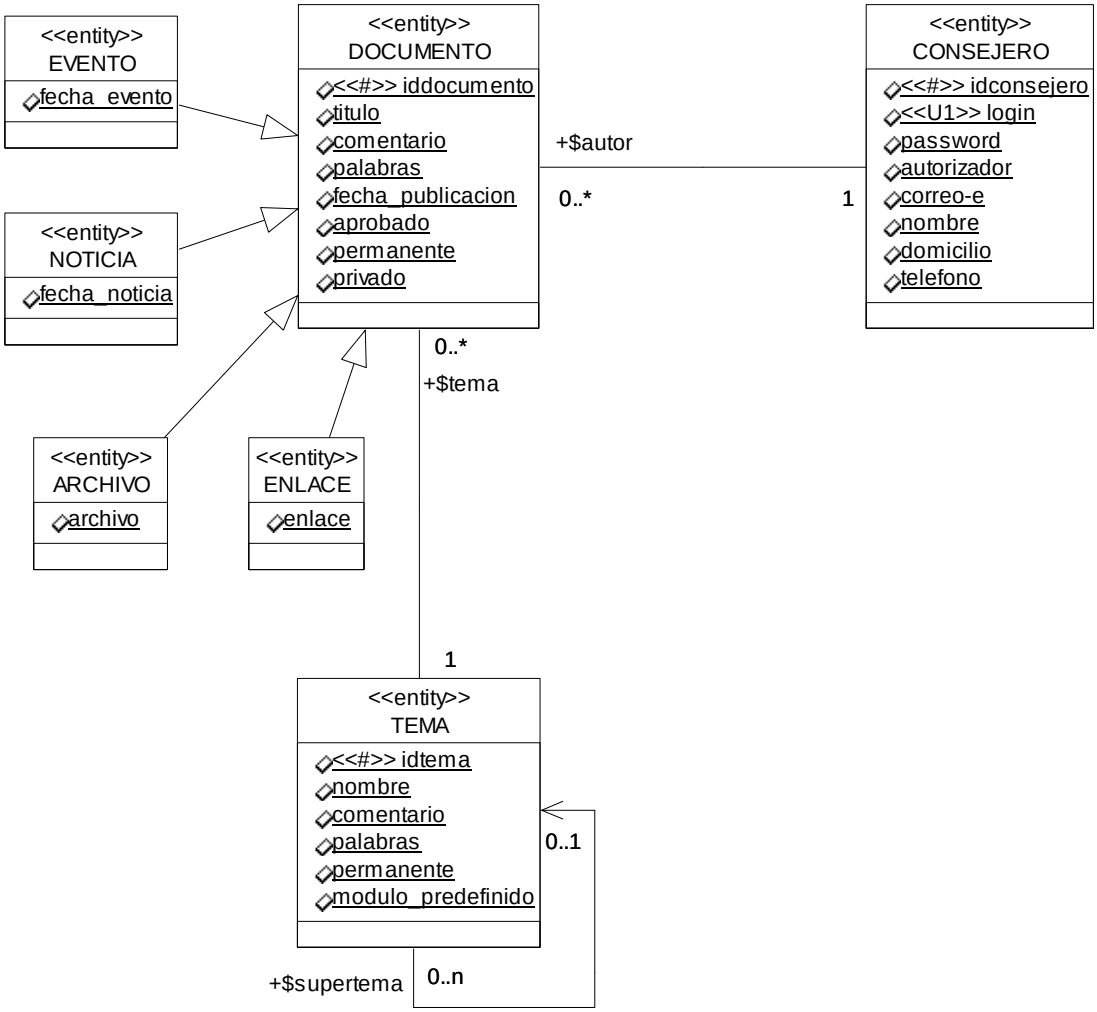

Fig. 2.4.1-4: Modelo de datos persistentes en análisis

Explicación:

Estereotipos que no son parte del estándar UML:

- **#:** Se refiere a la clave primaria.
- **U[*número*]:** se refiere a un atributo único, E. g. U1 indica que todas las propiedades que tengan este estereotipo deben ser únicas de forma combinada.

Clases de entidad:

- **Tema:** se refiere a los temas en que se agrupan los documentos a publicar, están ordenados jerárquicamente formando un árbol.
- **Consejero:** contiene la información de los consejeros, tanto la personal como la necesaria para iniciar la sesión.
- **Documento:** es la información que se va a publicar.
- **Evento:** cualquier evento programado por la CMDH para realizar en el futuro
- **Noticia:** noticias de la CMDH y de otros lados
- **Archivo:** cualquier archivo (Word, pdf. etc.) que se quiera publicar.
- **Enlace:** enlace a algún sitio en Internet.

Atributos:

- **Palabras en tema y documento:** es una lista de palabras clave para realizar búsquedas.
- **Módulo predefinido en tema:** cada tema tiene un módulo (realización de caso de uso) predefinido, que sirve para listar documentos, mostrar cierto contenido o realizar búsquedas filtradas.
- **Permanente en tema:** indica que el tema no se puede eliminar (e.g. cuando el tema está asociado a un módulo importante)
- **Autorizador en consejero:** indica si el consejero es también autorizador o no.
- **Permanente en documento:** indica que el documento no será borrado de manera automática por cuestión de espacio.
- **Privado en documento:** indica que el documento sólo puede ser visto por consejeros.

Relaciones:

- Todo tema puede estar teniendo uno y sólo un supertema (tema superior).
- Todo tema puede ser poseedor de uno o más subtemas
- Un consejero puede ser autor de uno o más documentos
- Un documento debe ser hecho por uno y sólo un consejero.
- Un tema puede estar agrupando a uno o más documentos
- Un documento debe ser catalogado en uno y sólo un tema.

Session manager – analisis

Este paquete contiene el siguiente diagrama de clases:

DIAGRAMA DE CLASES DEL PAQUETE *SESSIONMANAGER - ANÁLISIS*

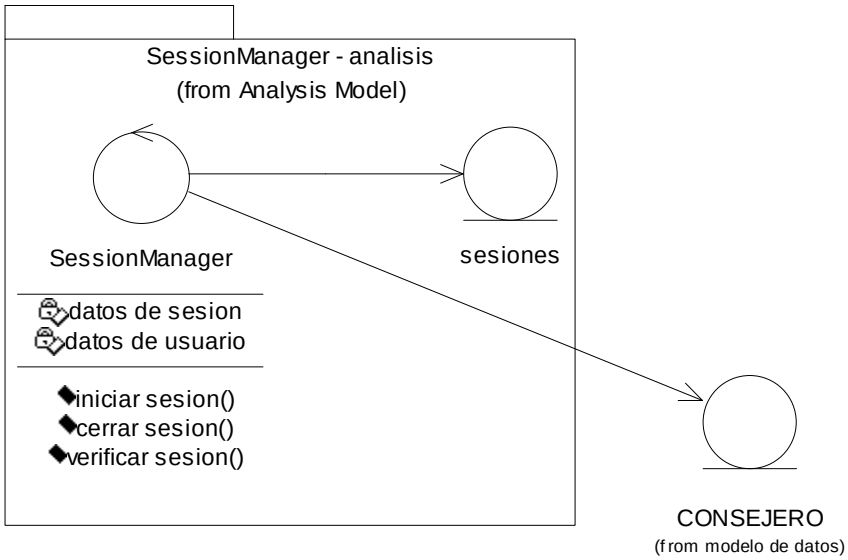

Fig. 2.4.1-5: Diagrama de clases del paquete SessionManager - análisis

Explicación:

- **SessionManager:** clase que controla las sesiones.
- **sesiones:** clase que almacena la información de las sesiones (es clase de análisis, en el diseño puede plantearse como parte de una base de datos o por otro método como las variables de sesión del PHP o las *cookies*)

La clase *SessionManager* se comunica con la clase *CONSEJERO* (paquete: modelo de datos) para verificar que los datos de inicio de sesión sean correctos (*login,password*) o para verificar que las variables de sesión (en sesiones) sean correctas.

Cerrar sesión – análisis

Este paquete describe la realización del caso de uso cerrar sesión desde el punto de vista del análisis.

Esta operación está descrita por el siguiente diagrama de secuencia:

REALIZACIÓN DEL CASO DE USO *CERRAR SESIÓN*

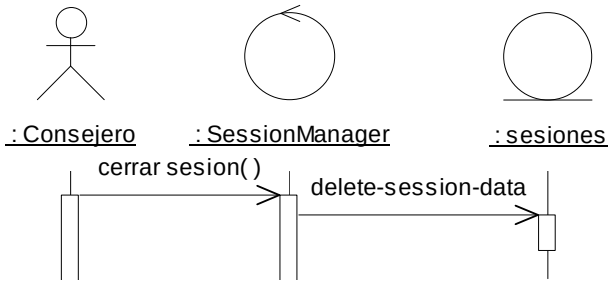

Fig. 2.4.1-6: Diagrama de secuencia que describe la realización del caso de uso cerrar sesión

Explicación:

Secuencia para cerrar sesión:

1. El consejero invoca la clase *SessionManager* mediante el método *cerrar sesion()*
2. *SessionManager* elimina los datos de sesión de la clase de entidad *sesiones*.
3. El resultado es devuelto al usuario que invoco el objeto y su operación.

Verificar sesión análisis

Paquete con la realización del caso de uso verificar datos de sesión en análisis.

Esta operación queda descrita en el siguiente diagrama:

REALIZACIÓN DEL CASO DE USO *VERIFICAR DATOS DE SESIÓN*

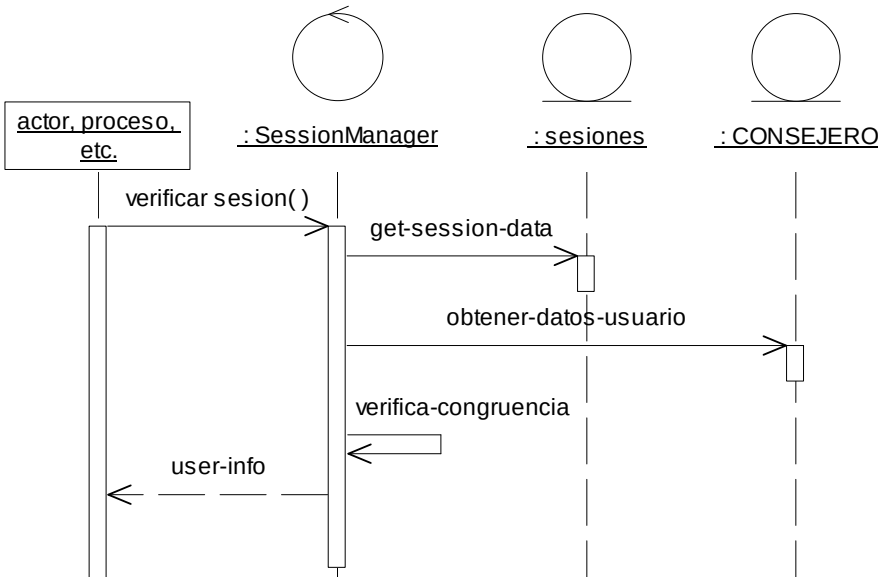

Fig. 2.4.1-7: Diagrama de secuencia de la realización del caso de uso verificar datos de sesión

Explicación:

Secuencia para verificar sesión:

1. Alguna clase o paquete o proceso, etc. invoca la clase *SessionManager* mediante el método *verificar sesion()*.
2. *SessionManager* obtiene de la clase de entidad *sesiones* la información de la sesión.
3. *SessionManager* valida los datos de la sesión comparándolos con la información de la clase de entidad *CONSEJERO.*
4. *SessionManager* aporta la información de la sesión confirmada y la información adicional al objeto que lo solicitó.
5. En caso de sesión inválida se devuelve un mensaje de invalidez de sesión y se elimina la sesión inválida de la clase de entidad *sesiones*.

Módulos – análisis

En este paquete se incluirán las realizaciones de los casos de uso desde el punto de vista del análisis. Ya que el enfoque es modular y los paquetes son independientes unos de otros y no afectan al resto del sistema no se consideran en sí mismos significativos para la arquitectura.

Por tanto sólo se mencionan, para esta etapa se han registrado los siguientes pudiendo aumentar su número:

PAQUETES DENTRO DEL PAQUETE *MÓDULOS-ANÁLISIS*

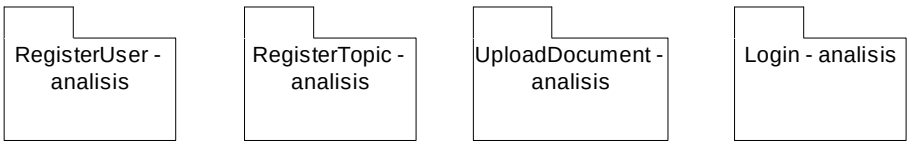

Fig. 2.4.1-8: Paquetes dentro del paquete módulos-análisis

Explicación:

- **RegisterUser-análisis:** contiene la realización del caso de uso registrar usuario en su versión de análisis.
- **RegisterTopic-análisis:** contiene la realización del caso de uso crear tema en su versión de análisis.
- **Login-análisis:** contiene la realización del caso de uso iniciar sesión en su versión de análisis.
- **UploadDocument-análisis:** contiene la realización del caso de uso publicar en su versión de análisis.
  Nota: otros módulos se pueden agregar según se necesite.

Navegar análisis

Este paquete describe la navegación en general; por navegación entendemos el desplazamiento que realiza el usuario en un sitio a través de la información.

Si la información está organizada por temas, entonces el usuario se “desplaza” a través de los temas hasta encontrar lo que busca.

Por tanto se requiere de un sistema que le presente al usuario las opciones por las cuales pueda “desplazarse” junto con las acciones (casos de uso) que pueda realizar.

Para que quede más claro se ilustrará mediante un ejemplo:

Supongamos que un usuario ingresa a una página X, por ejemplo www.amazon.com para comprar un libro, la página se muestra de la siguiente forma:

EJEMPLO PARA ILUSTRAR LA NAVEGACIÓN

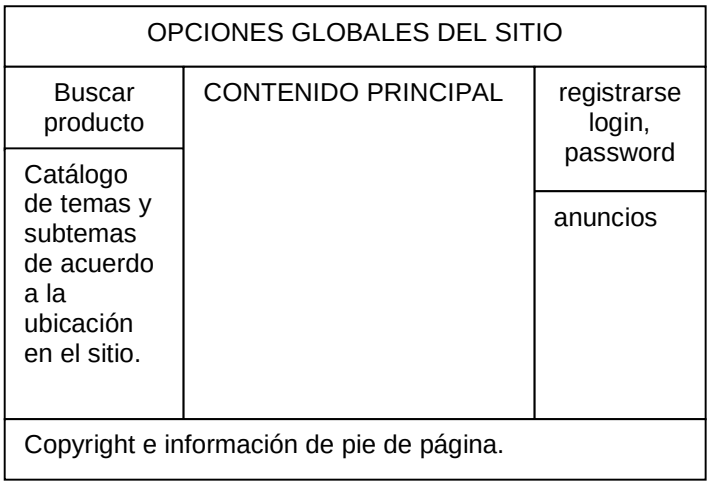

Fig. 2.4.1-9: Ejemplo para ilustrar la navegación

En este ejemplo todo lo que no es el “contenido principal” son las opciones de navegación. Pero incluso la capacidad de organizar las diversas opciones de navegación con el contenido principal forma parte del paquete *navegación-análisis*.

Este paquete describe, por tanto, la integración global del sistema, de los casos de uso (vistos como módulos) y de la navegación.

Esta integración se describe de la siguiente forma:

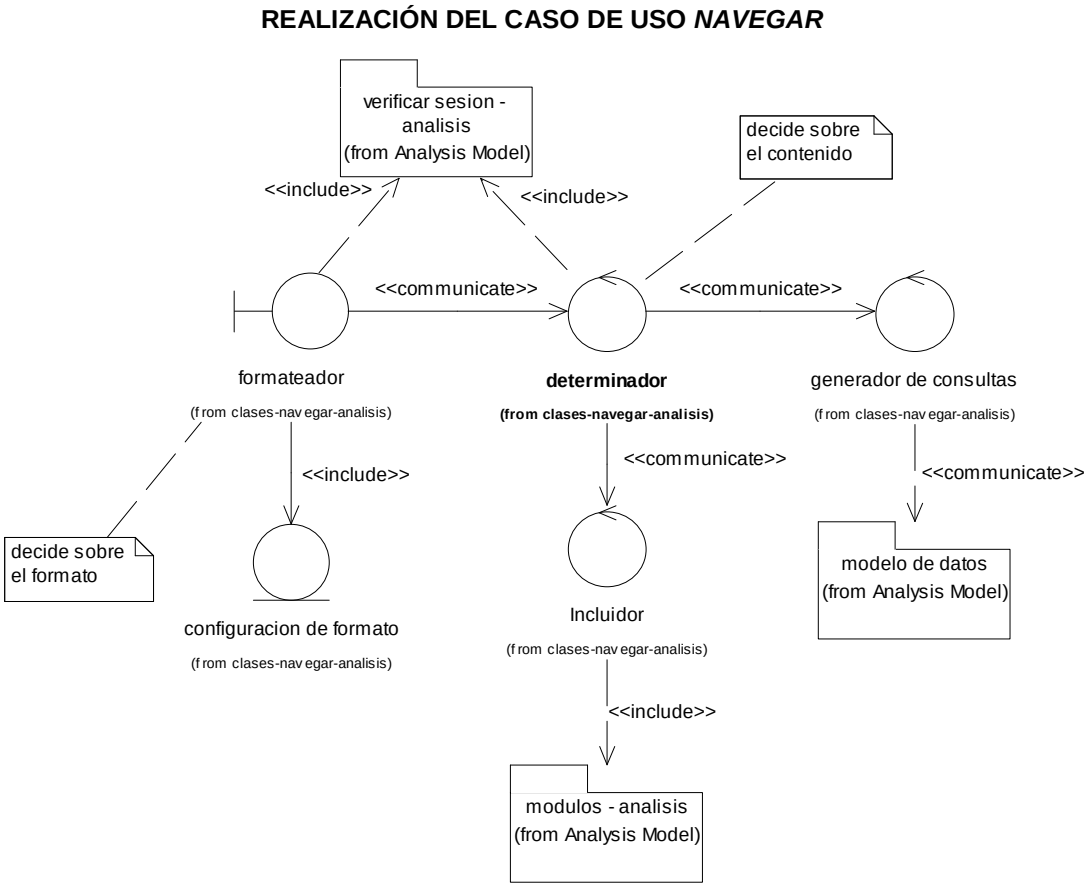

Fig. 2.4.1-10: Realización del caso de uso navegar *e integración de todos los elementos para presentación al usuario*

Explicación:

- **Formateador:** es una clase de salida de resultados, que establece el formato y los estilos y vuelca el contenido en HTML, igualmente crea las ligas a los estilos CSS, muestra los *menús* en formato adecuado, etc.
- **Configuración de formato:** información acerca del formato (e.g. un archivo XML de configuración).
- **Determinador:** clase que determina qué *menús* y qué contenido se va a mostrar, determina el módulo (realización de caso de uso) a utilizarse, etc.
- **Generador de consultas:** genera las consultas necesarias para construir el menú de los temas.
- **Incluidor:** incluye (ejecuta) el caso de uso correspondiente y controla su salida.

Secuencia de Navegar-Análisis

La secuencia es como sigue:

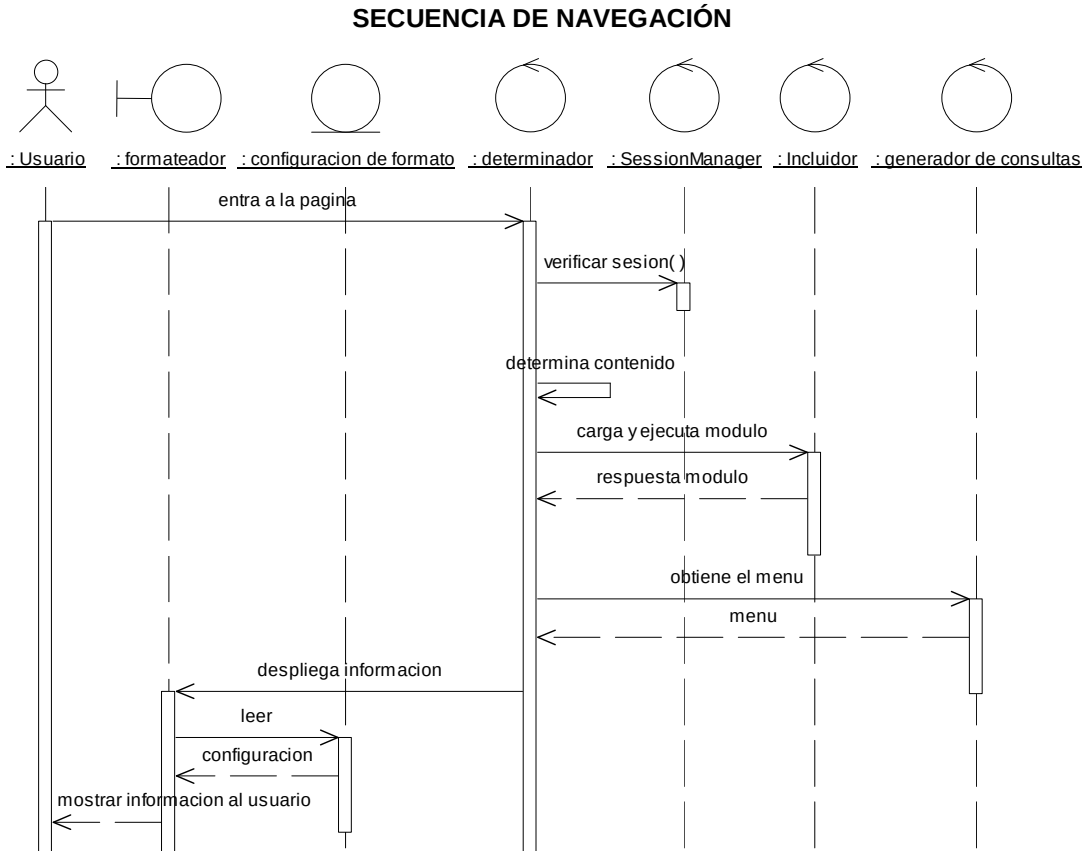

Fig. 2.4.1-11: Secuencia de navegación

Explicación:

1. Un *usuario* se conecta a la página o a una parte del sistema en particular.
2. El *determinador* verifica la sesión mediante un mensaje a *SessionManager*.
3. El *determinador* determina el *módulo* (caso de uso) a realizarse.
4. El *determinador* solicita al *incluidor* ejecute el *módulo* especificado.
5. El *incluidor* ejecuta el *módulo* y regresa el resultado.
6. El *determinador* solicita al *generador de consultas* que obtenga el menú.
7. El *generador de consultas* regresa el menú.
8. El *determinador* envía el contenido (resultado del *módulo*) y el menú al *formateador* para que lo despliegue (lo muestre al usuario).
9. El *formateador* lee la información de formato de la clase *configuración de formato*.
10. El *formateador* envía la información con el formato adecuado al usuario.

2.4.1.4. Diagrama Entidad-Relación

DIAGRAMA ENTIDAD-RELACIÓN

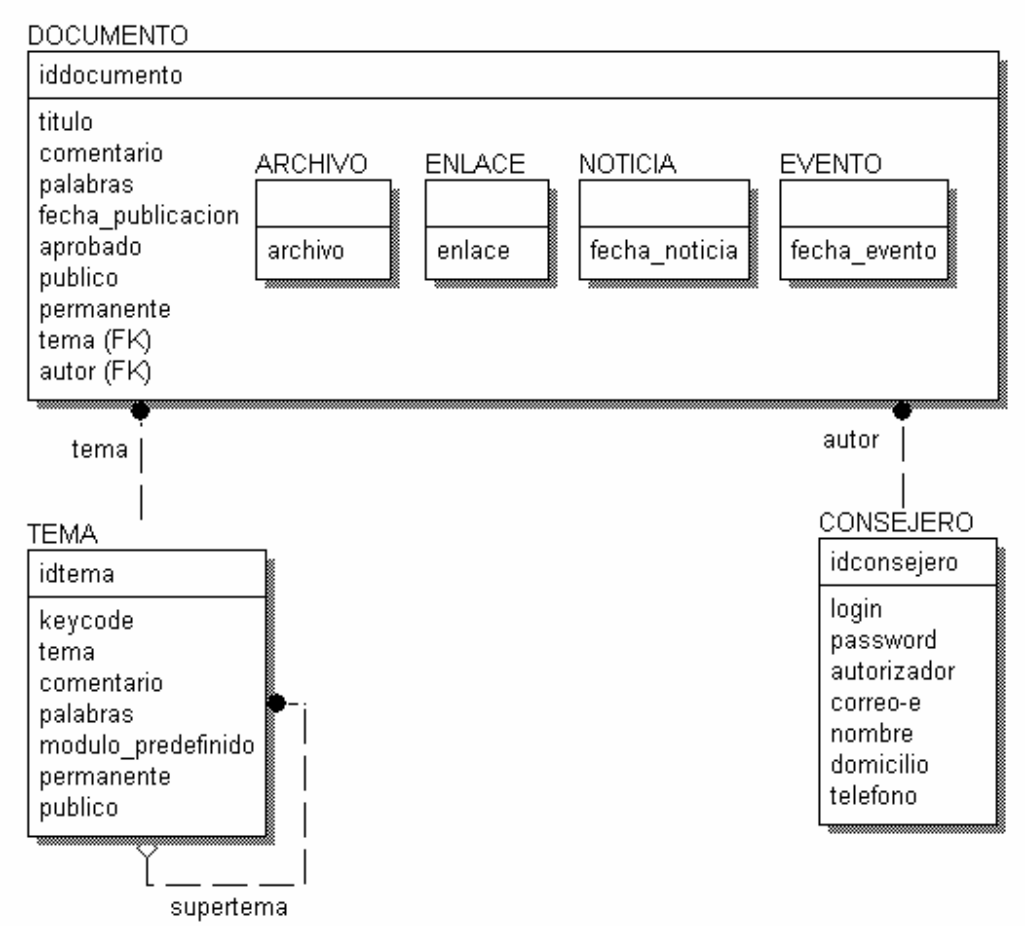

Fig. 2.4.1-12: Diagrama Entidad-Relación

El Diagrama Entidad-Relación corresponde con el paquete de análisis: *modelo de datos* explicado más arriba.

2.4.1.5. Modelo de Diseño

El modelo de diseño está formado por capas y son las siguientes:

CAPAS EN EL MODELO DE DISEÑO

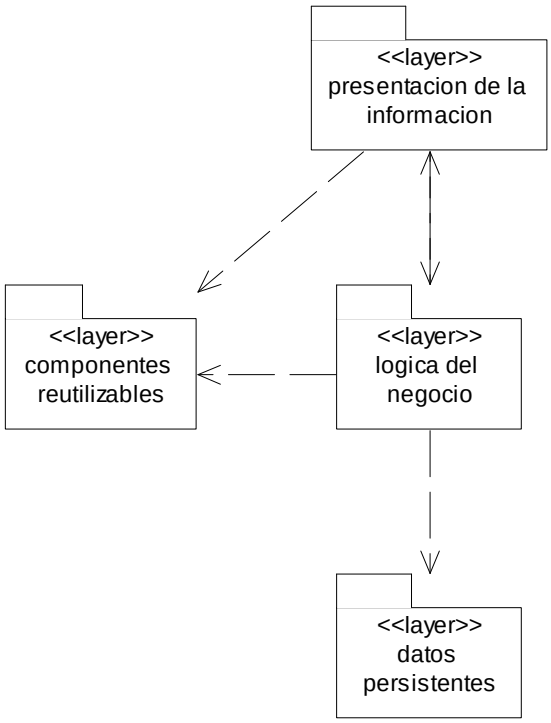

Fig. 2.4.1-13: Capas en el modelo de diseño

Explicación:

- La capa de **datos persistentes** contiene la estructura de la base de datos.
- La capa de **componentes reutilizables** contiene diversas clases que pueden ser usadas en cualquier parte del sistema o en cualquier otro sistema.
- La capa de la **lógica del negocio** contiene las realizaciones de los casos de uso.
- La capa de **presentación de la información** se encarga de dar formato al sitio Web.

2.4.1.5.1. Especificación de las capas

Capa de Datos Persistentes

Se normalizaron algunos campos en la clase consejeros. También se optó por concentrar en una tabla (documentos) todas las subclases y añadir un clasificador (diseño de una sola tabla). Finalmente se añadió el campo *keycode* en la clase temas para hacer referencia al tema de una manera más conceptual y para que al desarrollar el resto del sistema y del sitio no se dependa de la clave primaria asignada por el gestor de bases de datos.

El diseño de la base de datos queda así:

DISEÑO DE LA BASE DE DATOS

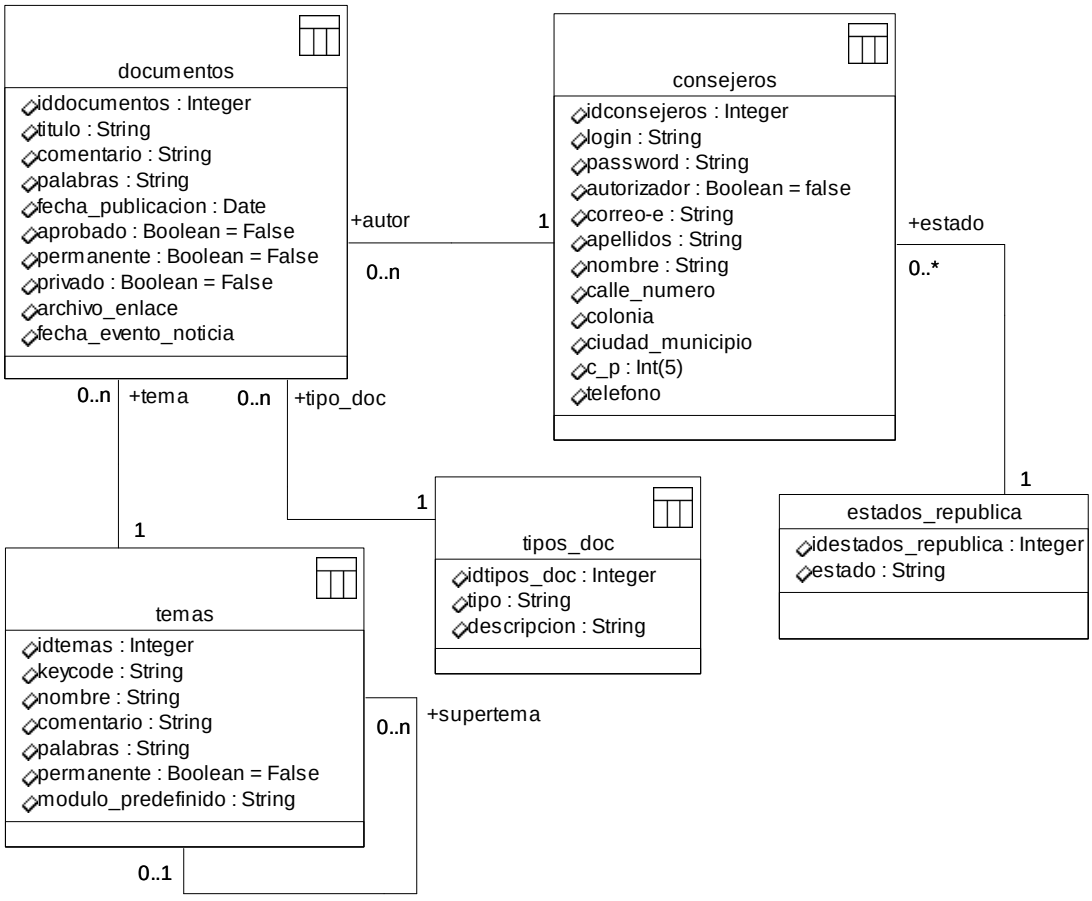

Fig. 2.4.1-14: Diseño de la base de datos

Usando la notación tradicional se expresa de la siguiente forma (considerando MySQL como manejador de bases de datos):

Relación: **ESTADOS_REPUBLICA**

Columna	**idestados_republica**	**estado**
Tipo_llave	PK	
NN/U	NN	NN
Tipo_dato	Integer	Varchar(20)
Ejemplo	1	Campeche

Relación: **TIPOS_DOC**

Columna	**idtipos_doc**	**tipo**	**descripcion**
Tipo_llave	PK		
NN/U	NN	NN	N
Tipo_dato	Integer	Varchar(15)	Varchar(100)
Contenido	1	Archivo	Archivo en Word, pdf. etc.
	2	Evento	Evento programado
	3	Noticia	Noticia de la CMDH
	4	Enlace	Enlace de internet

Relación: **TEMAS**

Columna	**idtemas**	**Keycode**	**nombre**	**comentario**
Tipo_llave	PK			
NN/U	NN	NN/U1	NN	N
Tipo_dato	Integer	Varchar(20)	Varchar(50)	Text
Ejemplo	1	MUJER	Mujer y DH	Derechos de la mujer

Columna	**Palabras**	**permanente**	**modulo_predefinido**	**Supertema**
Tipo_llave				FK1
NN/U	NN	NN	NN	N
Tipo_dato	Text	Bool	Varchar(50)	Integer
Ejemplo	Mujer, derechos	False	ListDocuments	

Relación: **CONSEJEROS**

Columna	**idconsejeros**	**login**	**password**	**autorizador**	**correo_e**
Tipo_llave	PK				
NN/U	NN	NN/U1	NN	NN	NN
Tipo_dato	Integer	Varchar(20)	Varchar(20)	Bool	Varchar(100)
Ejemplo	1	Juan	Juan232#!	False	juan@msn.net

Columna	**apellidos**	**nombre**	**calle_numero**	**colonia**
Tipo_llave				
NN/U	NN	NN	NN	NN
Tipo_dato	Varchar(75)	Varchar(50)	Varchar(100)	Varchar(50)
Ejemplo	Alvarez	Juan	Morelos #13	Independencia

Columna	**ciudad_municipio**	**Estado**	**c_p**	**telefono**
Tipo_llave		FK1		
NN/U	NN	NN	NN	NN
Tipo_dato	Varchar(100)	Integer	Int(5)	Varchar(50)
Ejemplo	Campeche	1	07289	(736)3837-2837

Relación: **DOCUMENTOS**

Columna	**iddocumentos**	**Titulo**	**comentario**	**Palabras**
Tipo_llave	PK			
NN/U	NN	NN	NN	NN
Tipo_dato	Integer	Varchar(20)	Text	Text
Ejemplo	1	Informe	Informe mensual	Informe, meses

Columna	**fecha_publicacion**	**Aprobado**	**permanente**	**privado**
Tipo_llave				
NN/U	NN	NN	NN	NN
Tipo_dato	Date	Bool	Bool	Bool
Ejemplo	2004-10-22	trae	False	False

Columna	**archivo_enlace**	**fecha_evento_noticia**	**autor**	**tema**	**tipo_doc**
Tipo_llave			FK1	FK2	FK3
NN/U	N	N	NN	NN	NN
Tipo_dato	Varchar(100)	Date	Integer	Integer	Integer
Ejemplo	Informe23.doc		1	1	1

Capa de Componentes Reutilizables

Formada por los siguientes paquetes:

PAQUETES DE LA CAPA *COMPONENTES REUTILIZABLES*

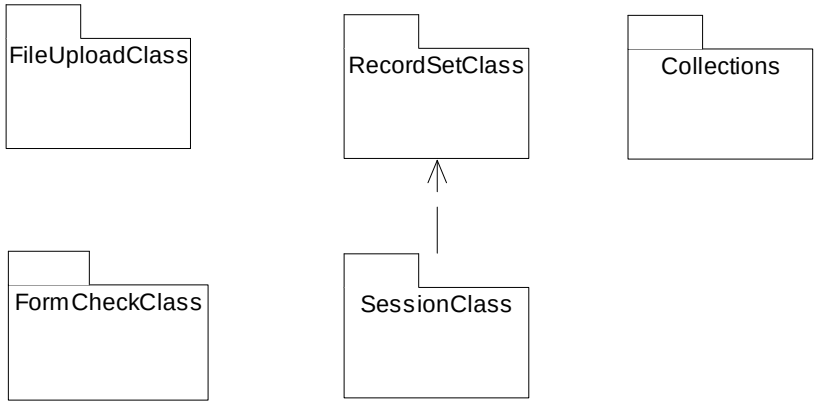

Fig. 2.4.1-15: Paquetes de la capa componentes reutilizables

Explicación:

- **FileUploadClass:** contiene la clase *FileUploadClass* que sirve para controlar los archivos que se envían a servidor mediante formulario Web de manera segura y sencilla.
- **FormCheckClass:** contiene la clase *FormCheckClass* que sirve para validar formularios Web. La clase funciona mediante reglas.
- **RecordSetClass:** contiene la clase *RecordSetClass* que sirve para realizar consultas a la base de datos de forma sencilla y controlar los resultados y los errores.
- **SessionClass*:*** contiene la clase *SessionClass* que se deriva de la clase *RecordSetClass* y que sirve para controlar las sesiones de usuario de manera sencilla.
- **Collections:** contiene las siguientes tres clases: *CollectionClass*, *ArrayCollectionClass* y *ArrayObjCollectionClass*, y sirve para manejar colecciones mediante arreglo de datos de cualquier tipo (*ArrayCollectionClass*) y de objetos en particular (*ArrayObjCollectionClass*).

Capa de la Lógica del Negocio

Describe el funcionamiento global de la siguiente forma:

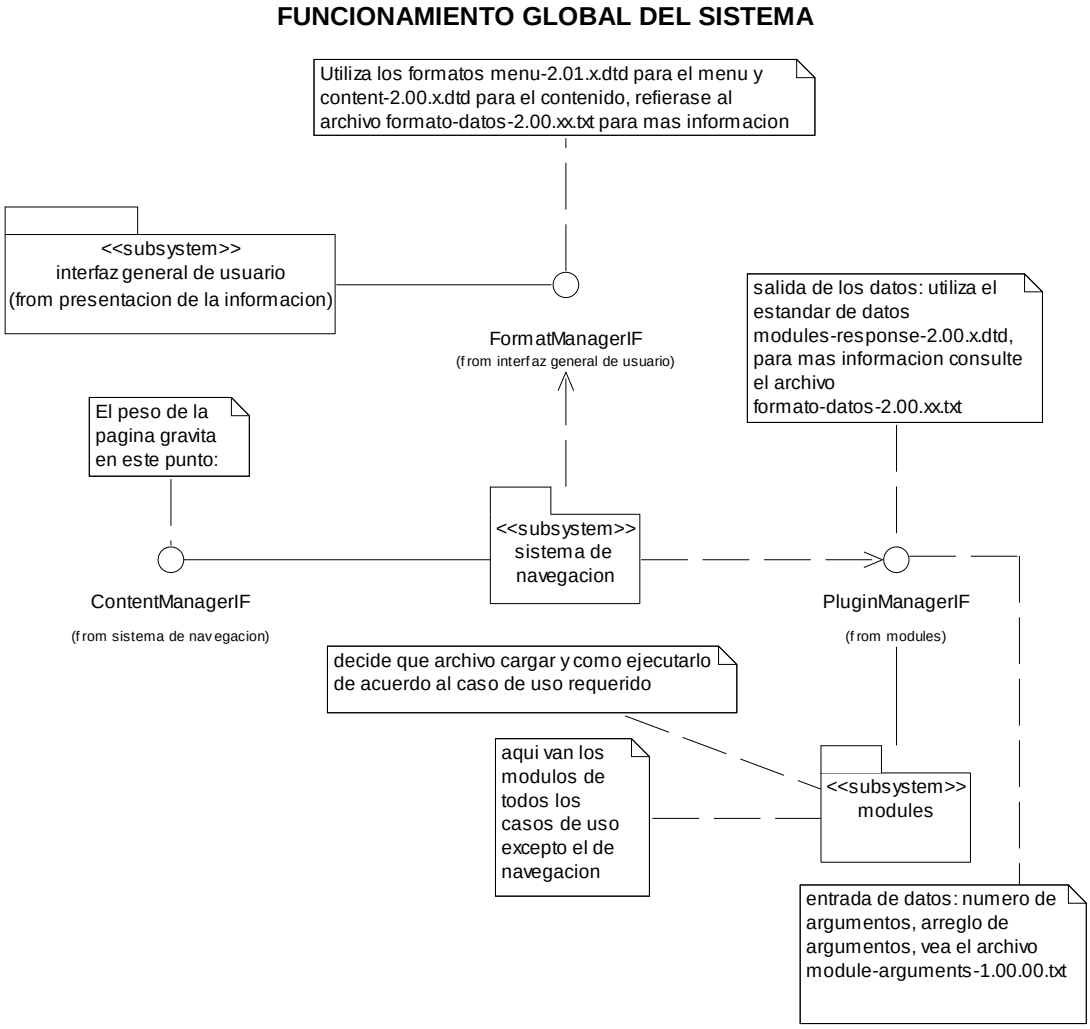

Fig. 2.4.1-16: Funcionamiento global del sistema

Explicación:

La capa contiene dos subsistemas: sistema de navegación y modules, y dos interfaces: *ContentManagerIF* y *FormatManagerIF*.

A su vez se conecta mediante una interfaz (*FormatManagerIF*) con un subsistema de la capa de presentación de la información: *interfaz general de usuario*.

El sistema está planteado como un control central que carga módulos independientes para realizar las peticiones de usuario. El resultado se envía a la *interfaz general de usuario* (capa *presentación de la información*) con el objetivo de darle formato.

La comunicación entre los subsistemas se realiza mediante XML y para los parámetros de los módulos mediante arreglos según se especifica en el diagrama.

La razón por la que se planteó así es para resolver varios problemas, entre otros:

- Independencia entre los menús (temas de navegación y opciones de usuario) y el contenido.
- Tener una infraestructura capaz de soportar nuevas funcionalidades que pueda requerir el usuario y que ni siquiera se hayan planteado, e.g. encuestas, recepción y proceso de noticias de otros sitios mediante RSS, etc.
- Independencia entre los datos y el formato.
- Poder agregar y quitar funciones del sistema sin que afecte el comportamiento global.

El flujo del programa es de la siguiente forma:

1. Una clase o un *script* (por ejemplo *index.php*) utiliza *ContentManagerIF* para crear las instancias de los subsistemas que se necesiten. Configura los parámetros de entrada (por ejemplo los parámetros GET y POST) e invoca el método de ejecución de la interfaz.
2. Entonces el subsistema de navegación determina el tipo de usuario que usa el sistema y de acuerdo a los parámetros determina el módulo (realización de caso de uso) a utilizarse.
3. Mediante la interfaz *PluginManagerIF* se crea una instancia del subsistema modules, y se invoca al módulo correspondiente a la petición de usuario. Se ejecuta el módulo y el resultado es recibido por el sistema de navegación, el resultado es analizado para saber si hay redirección o contenido, si hay redirección se realiza la redirección; si es contenido se genera un XML de salida de contenido.
4. Entonces el sistema de navegación realiza las consultas necesarias a los temas de navegación para construir el menú de navegación, igualmente obtiene de un archivo de configuración el menú con las opciones que tiene cada tipo de usuario (público, consejero, autorizador).
5. Se genera un XML de salida de menú y tanto el XML de contenido como el XML de menú son enviados al subsistema interfaz general de usuario mediante la interfaz *FormatManagerIF* para ser mostrada al usuario.

Especificación de los subsistemas

Modules

Contiene los módulos a utilizar, éstos módulos son totalmente independientes unos de otros y pueden ser estructurados u orientados a objetos.

Además contiene una clase para la gestión de la carga, configuración y ejecución de los módulos como se muestra:

SUBSISTEMA *MODULES*

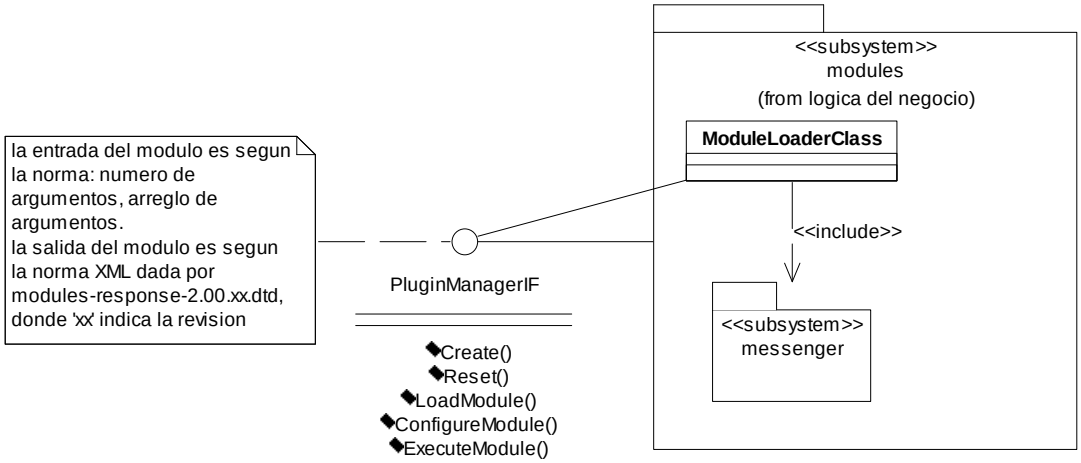

Fig. 2.4.1-17: Subsistema modules

Para esta etapa solo se tiene implementado el módulo *messenger* que sirve para enviar mensajes con un título, e.g. *error de base de datos*. Se implementó este módulo para probar el funcionamiento del subsistema.

La interfaz se explica más adelante.

Sistema de Navegación

Es responsable de lo siguiente:

1. determinar qué modulo va a ejecutarse.
2. Si no hay módulo, obtenerlo del tema en que se encuentra navegando.
3. Invocar al subsistema modules para la ejecución del módulo.
4. Procesar el resultado del módulo.
5. Generar el menú de navegación de acuerdo al tema en que se encuentre el usuario.
6. Generar el menú de usuario de acuerdo a su tipo (público, consejero, autorizador).
7. Generar los XML de salida para enviarlos al subsistema interfaz general de usuario.
8. Ejecutar el subsistema interfaz general de usuario para desplegar el contenido al usuario.
9. Controlar los errores.
10. Realizar redirecciones en caso de que un módulo las especifique.

El subsistema se describe a continuación:

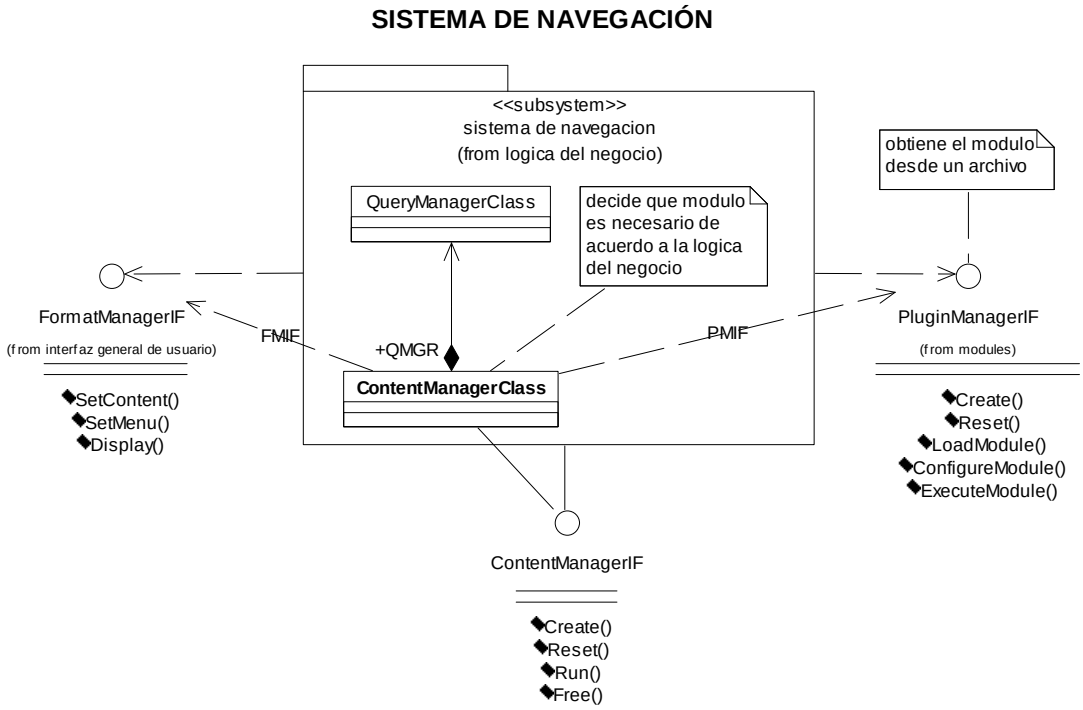

Fig. 2.4.1-18: Sistema de navegación

Explicación:

- **ContentManagerIF:** interfaz para controlar el subsistema externamente.
- **QueryManagerClass:** clase que realiza las consultas a la base de datos para obtener los menús de acuerdo al tema (problema 5 de la lista de arriba).
- **ContentManagerClass:** clase responsable de resolver los problemas 1, 2, 3, 4, 6, 7, 8, 9 y 10 de la lista de arriba. También procesa el resultado de la clase *QueryManagerClass* y le da un formato adecuado para generar el menú de navegación de acuerdo al tema en que se encuentre el usuario.

La forma típica de uso es mediante la interfaz, invocando las operaciones: *Create*, *Reset* y *Run*.

Nota: Estas dos clases son provisionales, mientras este subsistema es optimizado para un mejor rendimiento.

Especificación de las interfases

Las interfases en PHP 4.x no existen, por tanto se implementan mediante objetos, estos objetos se han modelado en una familia (todos se heredan de una clase interfaz genérica) con características comunes, la jerarquía de clases de las interfases usadas en la capa de la lógica del negocio y en la capa de presentación de la información es la siguiente:

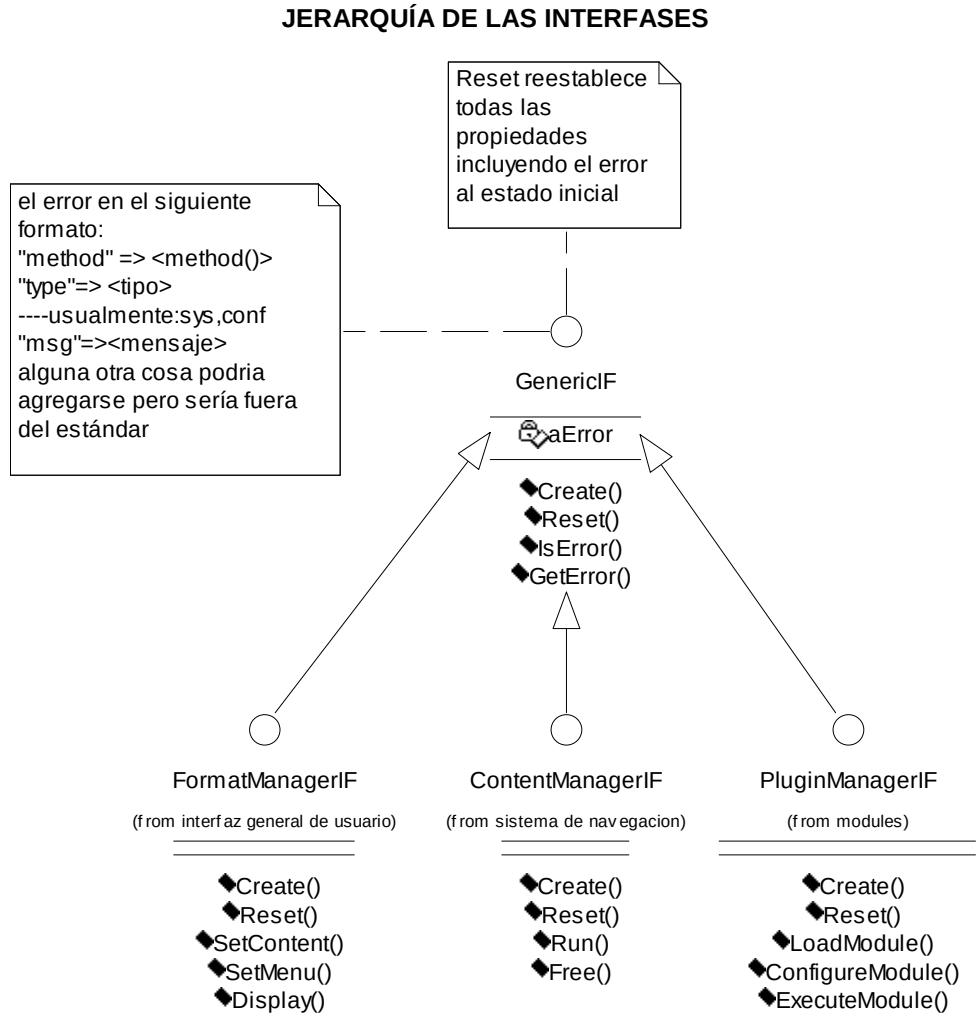

Fig. 2.4.1-19: Jerarquía de las interfases

Explicación:

- **GenericIF** es la interfaz genérica abstracta.
- **FormatManagerIF** es la interfaz del subsistema interfaz general de usuario de la capa presentación de la información.
- **ContentManagerIF** es la interfaz del subsistema sistema de navegación.
- **PluginManagerIF** es la interfaz del subsistema modules.

GenericIF

Propiedades:

1. **aError:** contiene información sobre los errores del subsistema.

Operaciones:

2. **Create:** crea y configura los objetos del subsistema
3. **Reset:** reestablece el subsistema al estado inmediato después de Create.
4. **IsError:** detecta si hay error
5. **GetError:** obtiene el error.

FormatmanagerIF

Operaciones:

1. **SetContent:** establece el contenido de la página.
2. **SetMenu:** establece el menú de la página.
3. **Display:** despliega la información al usuario.

ContentmanagerIF

Operaciones:

- **Reset:** reestablece el subsistema al estado inmediato después de Create y después asigna los parámetros de la página (tema, módulo y parámetros del módulo).
- **Run:** ejecuta las acciónes propias del subsistema para determinar el contenido, el menú, ejecutar el módulo, dar formato al resultado y desplegarlo al usuario.

PluginmanagerIF

Operaciones:

- **LoadModule:** carga el módulo en memoria.
- **ConfigureModule:** configura los parámetros del módulo.
- **ExecuteModule:** ejecuta el módulo y devuelve el resultado.

Capa de Presentación de la Información

Esta capa contiene únicamente un subsistema llamado *interfaz general de usuario*. Este subsistema es responsable de dar un formato y presentación adecuada a la información y de enviarla al navegador del usuario (desplegarla) para que sea vista.

El subsistema se muestra a continuación:

SUBSISTEMA *INTERFAZ GENERAL DE USUARIO*

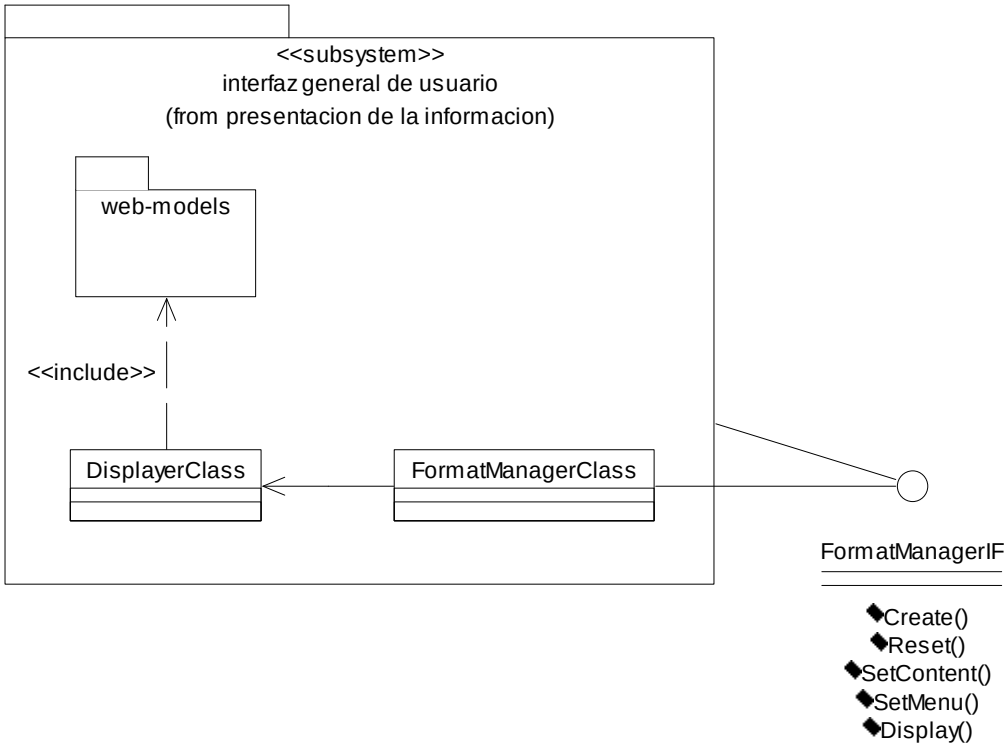

Fig. 2.4.1-20: Subsistema Interfaz General de Usuario

Explicación:

Clases:

- **FormatManagerClass:** clase que implementa la interfaz *FormatManagerIF* y realiza sus funciones. En particular analizar los XML de menú y contenido y transformarlos para que puedan ser desplegados.
- **DisplayerClass:** clase que genera y controla el espacio de objetos visuales de la página.

Paquetes:

- **Web-models:** familia de clases relacionadas entre sí que permiten la construcción robusta de páginas con menús y formularios, así como contenidos, utilizan plantillas HTML para el contenido estático y generan páginas basadas en CSS.

Paquete Web-Models

Consiste en una serie de clases agrupadas en paquetes y que cada clase tiene la capacidad de desplegar un pedazo de la información de la página en formato HTML.

Conceptos

Web-Models es una abstracción de un diseño típico de una página Web, durante su realización se acuñaron una serie de términos cuya definición se presenta a continuación:

1. **Página:** es documento en HTML o XHTML según se definen en la W3C (www.w3c.org). Desde el punto de vista de web-models, es lo que contiene a todos los demás elementos que se muestran. También puede verse como un conjunto de secciones e información adicional extraída de una plantilla.
2. **Sección:** se refiere a una parte visible de una página. En este proyecto se escogió un diseño sencillo que contiene 5 secciones que se muestran en la figura:

DISEÑO DE LA PÁGINA WEB

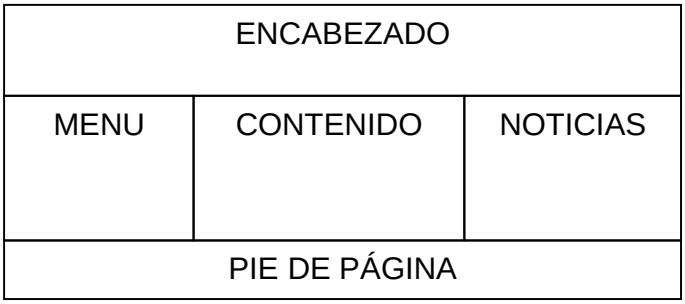

Fig. 2.4.1-21 Diseño de la página Web

Nota: No debe confundirse la sección *menú* con la información *menú*. La sección *menú* almacena la mayor parte de la información *menú* pero en sí misma es sólo un contenedor de información como las demás secciones. La información *menú* que proviene del subsistema *sistema de navegación* denota toda la información que no es *contenido.* Y el objeto *menú* se describe a continuación.

3. **Menú:** El *menú* (entendido como objeto) es el elemento capaz de almacenar la información *menú*. Es decir, los enlaces a los temas de navegación del sitio y a los módulos (casos de uso) del sistema. La forma de realizarlo es mediante enlaces. Un *menú* (objeto) tiene varias presentaciones: vertical para la sección *menú* y para la sección *noticias*; horizontal para la sección *encabezado* y *sólo texto* para la sección *pie de página*.
4. **Modelo:** Es un conjunto de variables con sus respectivos valores, las cuales pueden ser usadas (por referencia) en un formulario. Este concepto es una implementación limitada de lo que se conoce como XForms[2] que es una iniciativa de la W3C para mejorar la forma de realizar formularios Web.
5. **Forma:** Es un conjunto de controles que se relacionan con variables de un *modelo* y que permiten desplegar un formulario Web, igual que el modelo, es una implementación limitada de XForms.
6. **Etiqueta:** Cualquier texto al que se le aplica un *estilo* determinado.
7. **Contenido:** Es el resultado de cualquier módulo (caso de uso) en HTML sin formato.

[2] Para más información acerca de la recomendación de la W3C: XForms visite http://www.w3c.org/MarkUp/Forms/

8. **Estilo:** es un formato que se asigna a un elemento HTML, el estándar a utilizar se llama Cascade Style Sheet (CSS).[3]

Una *sección* es capaz de almacenar formas, etiquetas, *menús* (objetos) y contenidos. Se espera que el *contenido* lo genere el módulo (caso de uso) en cuestión y que el resto lo genere el *sistema de navegación* a lo cual en ese subsistema se le llamó simplemente: *menú*.

Especificación

La explicación de este paquete tendrá lugar sobre la jerarquía de las clases y posteriormente sobre las relaciones:

DIAGRAMA PARA EXPLICAR EL PAQUETE *WEB-MODELS*

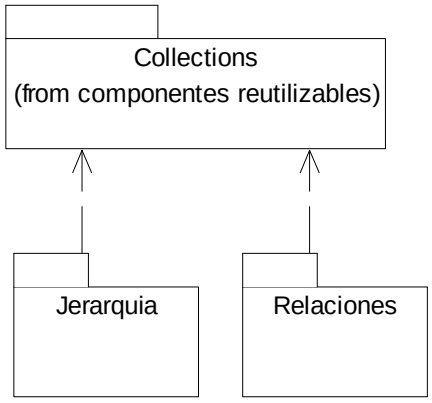

Fig. 2.4.1-22: Diagrama para explicar el paquete web-models

[3] Consulte http://www.w3c.org/Style/CSS/

Jerarquía

La jerarquía de clases del paquete *web-models* se muestra en el siguiente diagrama:

JERARQUÍA DE CLASES DEL PAQUETE *WEB-MODELS*

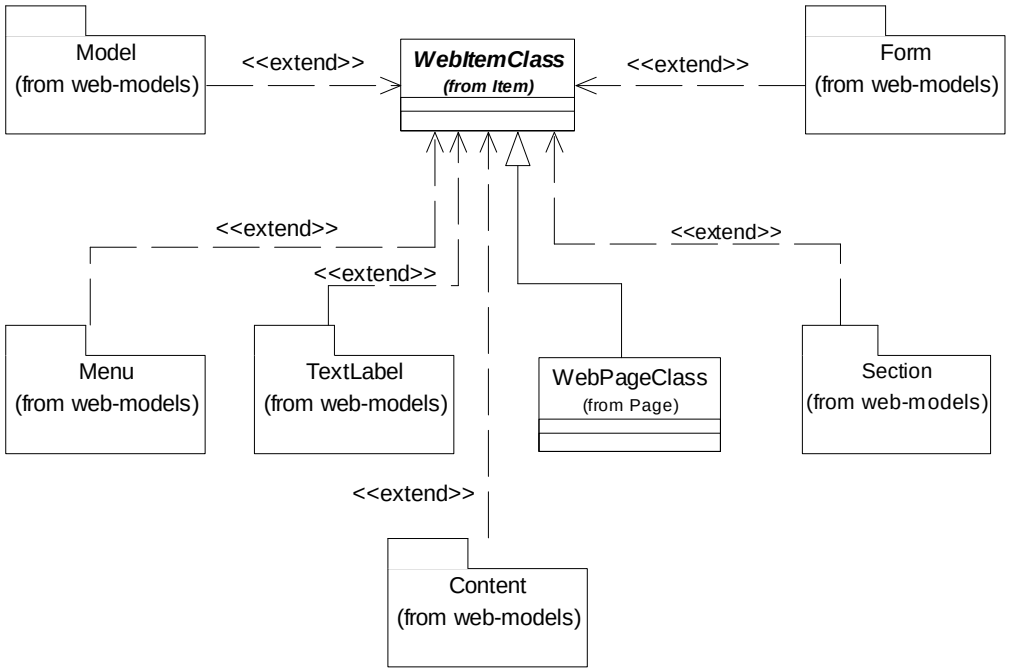

Fig. 2.4.1-23: Jerarquía de clases del paquete web-models

Explicación de las clases:

- **WebItemClass:** clase abstracta de la que se derivan todas las demás.
- **WebPageClass:** clase para almacenar controlar y desplegar páginas Web.

Explicación de los paquetes:

- **Model:** paquete de modelos de formulario.
- **Menu:** paquete de menú.
- **Form:** paquete de formularios.
- **TextLabel:** paquete de etiquetas de texto.
- **Content:** Paquete de contenido Web.
- **Section:** Paquete de secciones de página Web.

Especificación de los paquetes

Model

El paquete *Model* para almacenar modelos de datos de formularios Web contiene tres clases cuya herencia se muestra en el diagrama:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *MODEL*

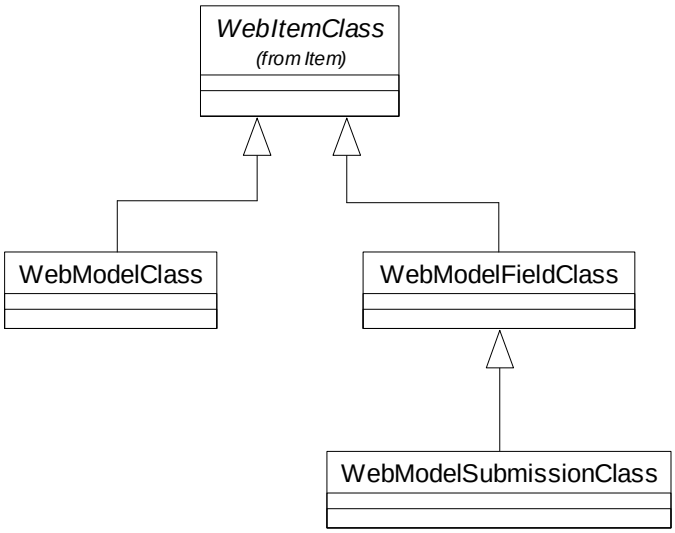

Fig. 2.4.1-24: Jerarquía de web-models*: Paquete* Model

Explicación:

- **WebModelClass:** Clase que almacena y controla el modelo.
- **WebModelFieldClass:** Clase que almacena un campo (variable) del modelo.
- **WebModelSubmissionClass:** Clase que almacena un campo con valores de envío (*submit*) del formulario.

Menu

El paquete *Menu* para almacenar, controlar y desplegar un *menú*, contiene cinco clases como se muestra en el diagrama:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *MENU*

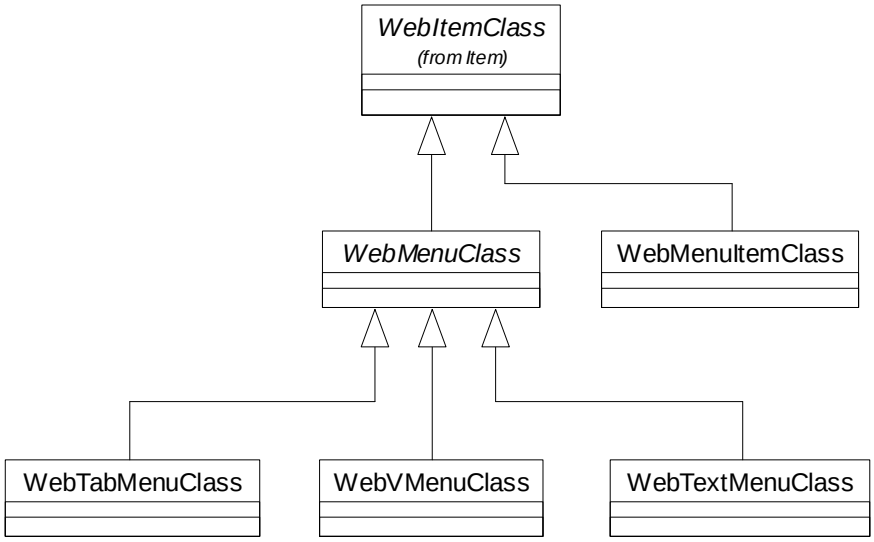

Fig. 2.4.1-25: Jerarquía de web-models*: Paquete* Menu

Explicación:

- **WebMenuClass:** clase abstracta que almacena un menú.
- **WebMenuItemClass:** clase que almacena un elemento de menú.
- **WebTabMenuClass:** clase que despliega un menú en forma horizontal con formato.
- **WebVMenuClass:** clase que despliega un menú en forma vertical con formato.
- **WebTextMenuClass:** clase que despliega un menú en forma te texto sin formato.

Form

El paquete *Form* sirve para almacenar y desplegar un formulario Web. Consta de nueve clases con la jerarquía mostrada en el diagrama:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *FORM*

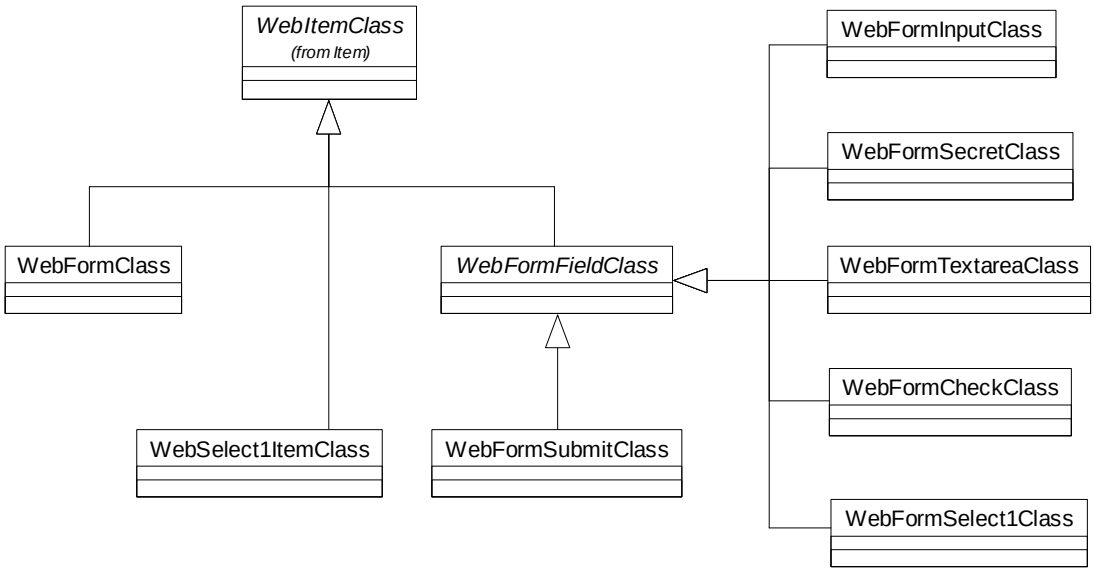

Fig. 2.4.1-26: Jerarquía de web-models*: Paquete* Form

Explicación:

- **WebFormClass:** clase que almacena un formulario Web y lo despliega.
- **WebFormFieldClass:** clase abstracta que representa un control de formulario.
- **WebFormSubmitClas:** clase que almacena un botón de enviar datos (*submit*) de un formulario.
- **WebFormInputClass:** control de entrada tipo texto.
- **WebFormSecretClass:** control de entrada de contraseñas.
- **WebFormTextateaClass:** control de entrada de área de texto.
- **WebFormCheckClass:** control de entrada de tipo activado/desactivado.
- **WebFormSelect1ItemClass:** control de entrada de selección de una opción entre varias.
- **WebSelect1ItemClass:** clase que almacena una opción de selección de un control de selección tipo *WebFormSelect1Class.*

TextLabel

Contiene dos clases cuya jerarquía se muestra:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *TEXTLABEL*

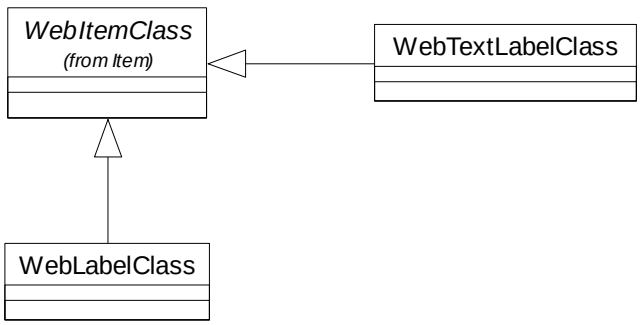

Fig. 2.4.1-27: Jerarquía de web-models*: Paquete* TextLabel

Explicación:

- **WebLabelClass:** contiene una etiqueta de texto.
- **WebTextLabelClass:** agrupa a varias etiquetas de texto aplicándoles un mismo estilo.

Content

Contiene una clase cuya jerarquía se muestra en el diagrama:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *CONTENT*

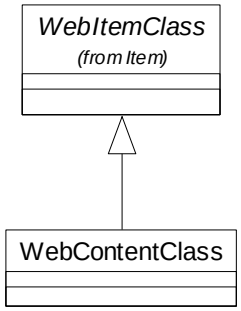

Fig. 2.4.1-28: Jerarquía de web-models*: Paquete* Content

Explicación:

- **WebContentClass:** clase que almacena un contenido Web (contenido HTML que proviene de la ejecución de los módulos en la capa de lógica del negocio).

Section

Este paquete contiene seis clases cuya jerarquía se muestra a continuación:

JERARQUÍA DE *WEB-MODELS*: PAQUETE *SECTION*

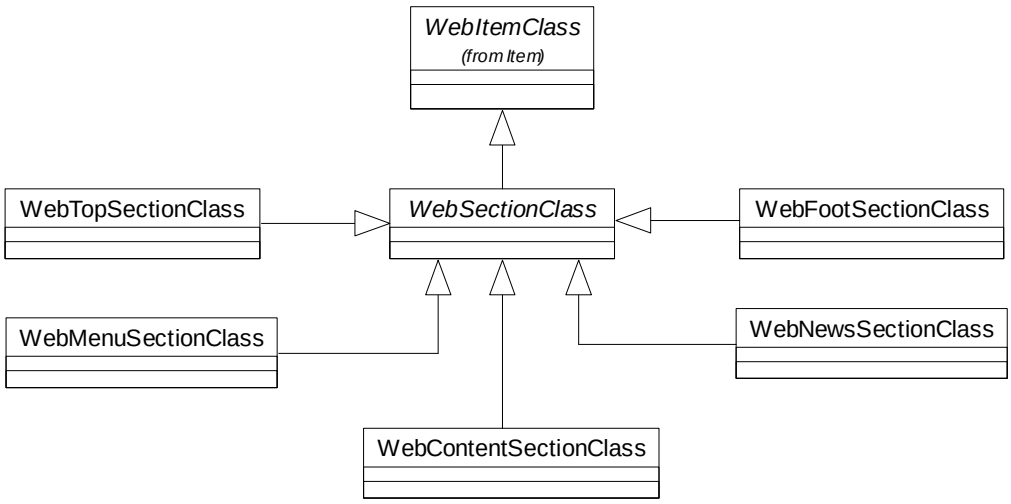

Fig. 2.4.1-29: Jerarquía de web-models*: Paquete* Section

Explicación:

- **WebSectionClass:** Clase abstracta que representa una sección de la página Web.
- **WebTopSectionClass:** Clase que almacena y despliega los elementos de la sección superior de la página (incluyendo el título).
- **WebMenuSectionClass:** Clase que almacena y despliega los elementos de la sección de menú (e.g. lado izquierdo) de la página.
- **WebContentSectionClass:** Clase que almacena y despliega los elementos de la sección de contenido de la página (e.g. al centro).
- **WebNewsSectionClass:** Clase que almacena y despliega los elementos de la sección de noticias (e.g. a la derecha).
- **WebFootSectionClass:** Clase que almacena y despliega los elementos de la sección de pie de página.

Relaciones del paquete *web-models*

Las relaciones se agrupan igualmente por paquetes como se muestra:

PAQUETES PARA EXPLICAR LAS RELACIONES DE *WEB-MODELS*

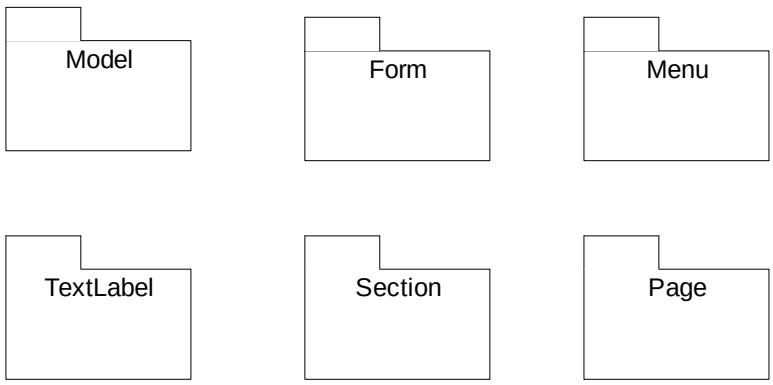

Fig. 2.4.1-30: Paquetes para explicar las relaciones de web-models

Las demás clases y los demás paquetes carecen de relaciones en sí mismos o están representadas sus relaciones en estos paquetes.

Todas las relaciones uno a muchos se realizan mediante colecciones, las cuales son por medio de la clase *ArrayObjCollectionClass* que facilita la colección de objetos en memoria mediante arreglos.

Model

Las relaciones de *Model* se muestran a continuación:

RELACIONES DE *WEB-MODELS*: PAQUETE *MODEL*

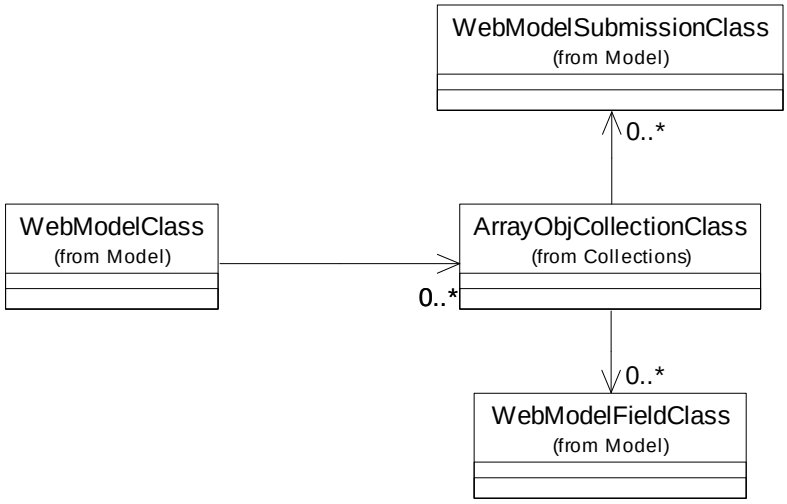

Fig. 2.4.1-31: Relaciones de web-models*: Paquete* Model

Explicación:

- Un objeto clase *WebModelClass* tiene varios objetos clase *WebModelFieldClass* y varios objetos clase *WebModelSubmissionClass*.

Form

Las clases del paquete *Form* están vinculadas con las clases del paquete *Model*, pues Model es el modelo de variables mientras que Form indica la forma de mostrarse al usuario mediante controles. En esto se está realizando una implementación limitada de *XForms* en espera de que esta iniciativa de la W3C madure y sea implementada por los navegadores más comunes.

RELACIONES DE *WEB-MODELS*: PAQUETE *FORM*

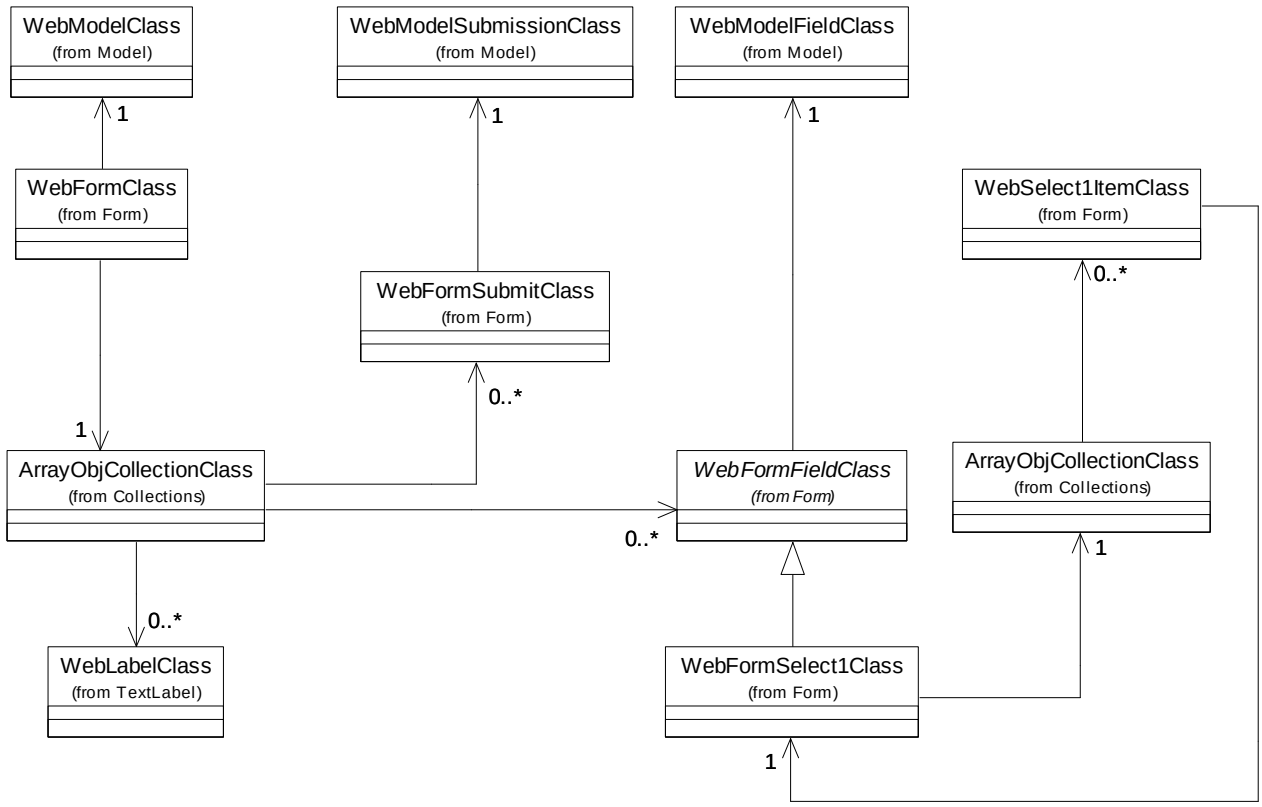

Fig. 2.4.1-32: Relaciones de web-models*: Paquete* Form

Explicación:

- Todo formulario Web debe tener relación con un modelo de datos.
- Un formulario Web es una colección de controles, que son clases derivadas de *WebFormFieldClass*.
- También se puede añadir texto mediante objetos *WebLabelClass*.
- Cada control se asocia con un objeto clase *WebModelFieldClass*.
- Cada objeto clase *WebFormSubmitClass* se asocia con un objeto clase *WebModelSubmissionClass*.
- Los controles tipo *WebFormSelect1Class* coleccionan varios objetos clase *WebSelect1ItemClass* que a su vez tienen relación con el objeto que los posee para determinar cuál está seleccionado por defecto.

Menu

Las relaciones del paquete *Menu* se muestran a continuación:

RELACIONES DE *WEB-MODELS*: PAQUETE *MENU*

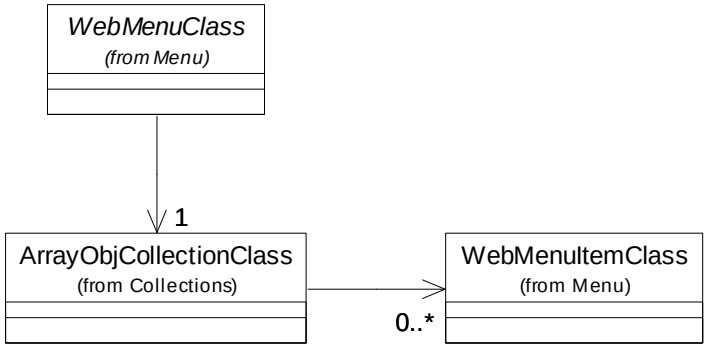

Fig. 2.4.1-33: Relaciones de web-models*: Paquete* Menu

Explicación:

- La clase *WebMenuClass* colecciona varios objetos clase *WebMenuItemClass*.

TextLabel

Las relaciones del paquete *TextLabel* se muestran a continuación:

RELACIONES DE *WEB-MODELS*: PAQUETE *TEXTLABEL*

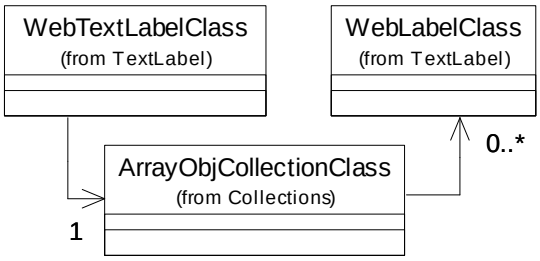

Fig. 2.4.1-34: Relaciones de web-models*: Paquete* TextLabel

Explicación:

- La clase **WebTextLabelClass** colecciona varios objetos clase **WebLabelClass**.

Section

Las relaciones del paquete *Section* se muestran a continuación:

RELACIONES DE *WEB-MODELS*: PAQUETE *SECTION*

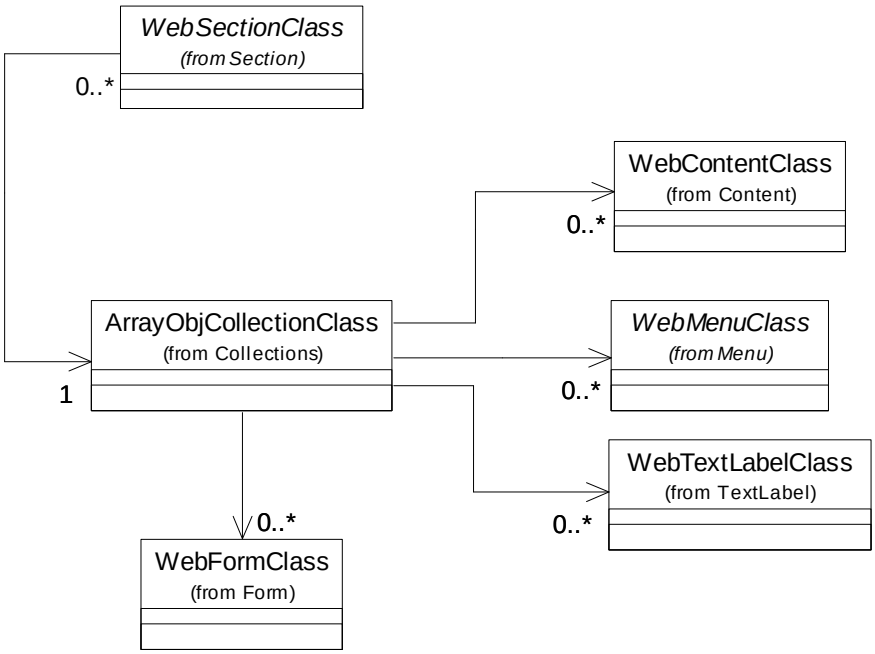

Fig. 2.4.1-35: Relaciones de web-models*: Paquete* Section

Explicación:

- Las clases heredadas de la clase *WebSectionClass* coleccionan varios objetos de las clases *WebContentClass*, *WebMenuClass*, *WebTextLabelClass* y *WebFormClass*.

Page

En este paquete se explican las relaciones de la clase *WebPageClass* y éstas son de la siguiente forma:

RELACIONES DE *WEB-MODELS*: CLASE *WEBPAGECLASS*

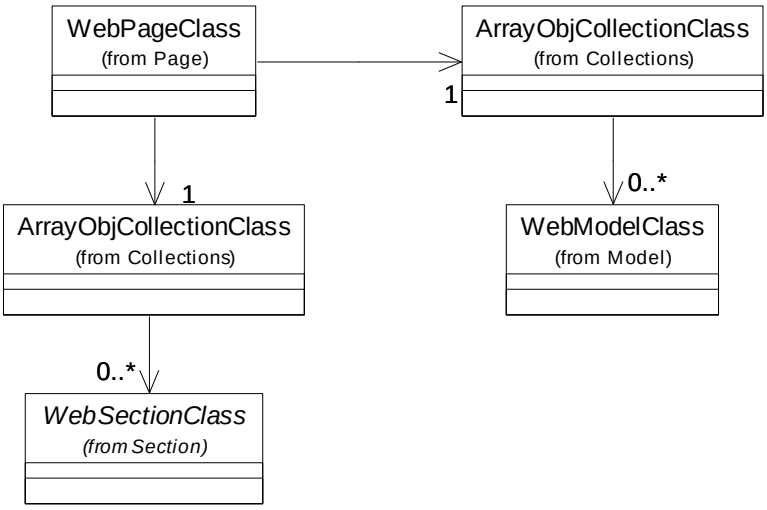

Fig. 2.4.1-36: Relaciones de web-models*: Clase* WebPageClass

Explicación:

- La clase *WebPageClass* colecciona varios modelos (*WebModelClass*) y varias secciones (*WebSectionClass*).
- Más específicamente colecciona un objeto de cada clase derivada de *WebSectionClass*, es decir un objeto de cada una de estas clases: *WebTopSectionClass, WebMenuSectionClass, WebContentSectionClass, WebNewsSectionClass* y *WebFootSectionClass*.

2.4.1.6. Modelo de Distribución

El modelo de distribución describe la forma en que están distribuidos los componentes de software físicamente, es decir, en los diferentes equipos que conforman la red del sistema.

En este caso se trata de una aplicación Web con base de datos, el modelo consiste en un servidor en el que se encuentra la aplicación y la base de datos y varios clientes en los cuales se ejecuta un navegador Web como Internet Explorer, Mozilla o Netscape Navigator.

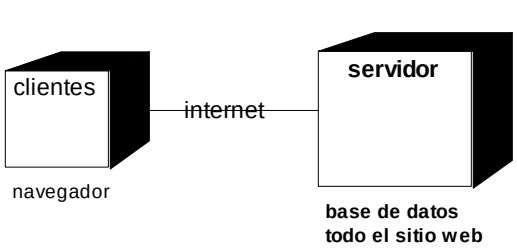

Fig. 2.4.1-37: Modelo de Distribución

Por tanto mostramos el diagrama con propósitos ilustrativos y en las posteriores fases de desarrollo se omitirá.

2.4.1.7. Modelo de Implementación

En este modelo se describe la relación entre los componentes ejecutables o programas fuente y las clases que se implementan.

La implementación obedece a las siguientes reglas:

1. Cada paquete una carpeta.
2. Cada subsistema una carpeta.
3. Cada clase un archivo.
4. Cada interfaz un archivo.

La base de datos es totalmente independiente de estas reglas.

Por tanto el modelo de implementación se divide en dos: el modelo general y los modelos específicos.

2.4.1.6.1. Modelo general de implementación

El modelo general describe la vista global de la implementación y la ubicación de los paquetes de más alto nivel.

Para ello se construyó una tabla, la notación es como sigue: las carpetas terminan con “/”; para indicar el directorio actual se denota con un punto “.”, el punto de partida es la carpeta donde se hospeda el sistema que varía según la configuración del servidor.

La tabla es como sigue:

TABLA 2.4.1-1: Descripción de los elementos del modelo general de implementación

Archivo/Carpeta	**Descripción:**
./	Carpeta raíz
config.php	Archivo de configuración con la información básica de las carpetas, contraseña de la base de datos e información general del sistema.
lib/	Capa de componentes reutilizables
NavigateSystem/	Subsistema “sistema de navegación”.
PresentationLayer/	Capa de presentación de la información.
generic/	Interfaz genérica.
modules/	Subsistema: “modules”.
uploaded/	Carpeta donde se almacenan los documentos que se van a publicar.
index.php	Programa principal.

Las dependencias se ilustran a continuación:

MODELO GENERAL DE IMPLEMENTACIÓN

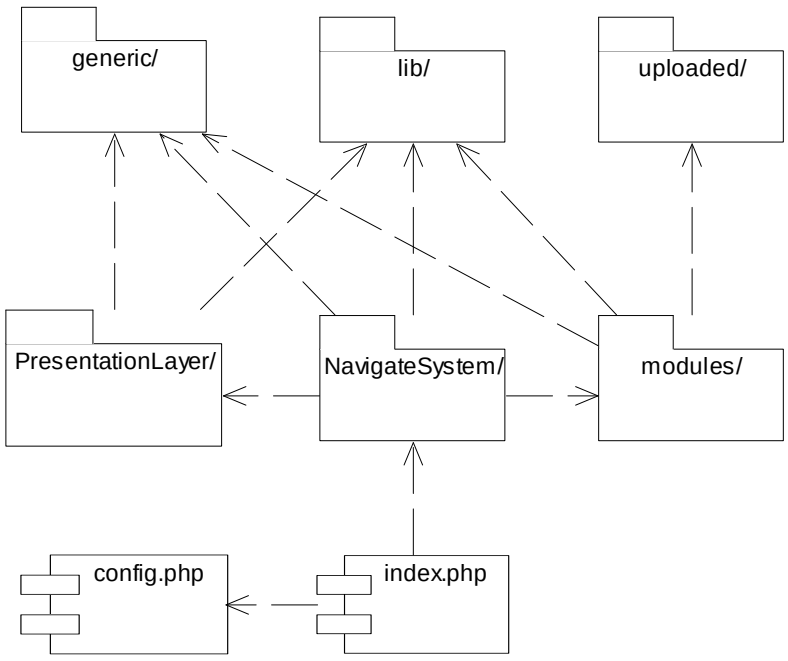

Fig. 2.4.1-38: Modelo General de Implementación

2.4.1.6.2. Modelo de implementación de **generic/**

Este modelo contiene únicamente un componente llamado *GenericIF.php* que implementa la interfaz genérica *GenericIF*.

2.4.1.6.3. Modelo de implementación de lib/

Este modelo contiene la implementación de los componentes reutilizables, los componentes se describen en la siguiente tabla:

TABLA 2.4.1-2: Modelo de implementación en *lib/*

Archivo/Carpeta	Descripción:
RecordSetClass.php	Implementación de la clase RecordSetClass.
SessionClass.php	Implementación de la clase SessionClass.
FormCheckClass.php	Implementación de la clase FormCheckClass.
FileUploadClass.php	Implementación de la clase FileUploadClass.
CollectionClass.php	Implementación de las clases: CollectionClass, ArrayCollectionClass y ArrayObjCollectionClass.php.

Las dependencias se muestran a continuación:

MODELO DE IMPLEMENTACIÓN DE *LIB/*

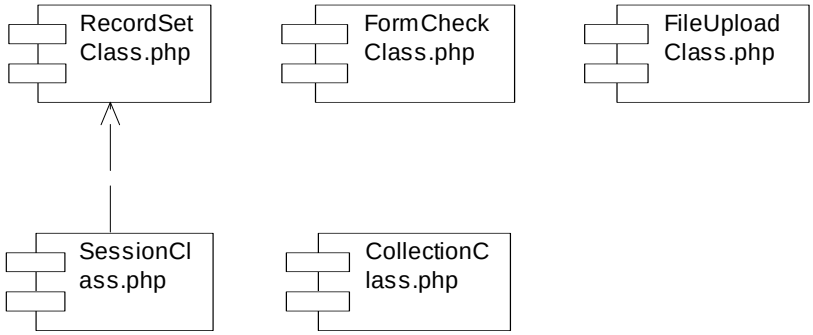

Fig. 2.4.1-39: Modelo de implementación de lib/

2.4.1.6.4. Modelo de implementación de NavigateSystem/

El modelo de implementación de *NavigateSystem* se describe mediante la siguiente tabla:

TABLA 2.4.1-3: Modelo de implementación de *NavigateSystem/*

Archivo/Carpeta	Descripción:
ContentManagerClass.php	Implementa la clase ContentManagerClass.
ContentManagerIF.php	Implementa la interfaz ContentManagerIF.
ContentManagerClass.xml	Archivo de configuración del componente ContentManagerClass.php
QueryManagerClass.php	Implementa la clase QueryManagerClass.
config.php	Archivo de configuración con la información básica de las carpetas, contraseña de la base de datos e información general del sistema.

Las dependencias se muestran en el siguiente diagrama (aquellos elementos que no están en la tabla significa que no pertenecen a esta carpeta):

MODELO DE IMPLEMENTACIÓN DE *NAVIGATESYSTEM/*

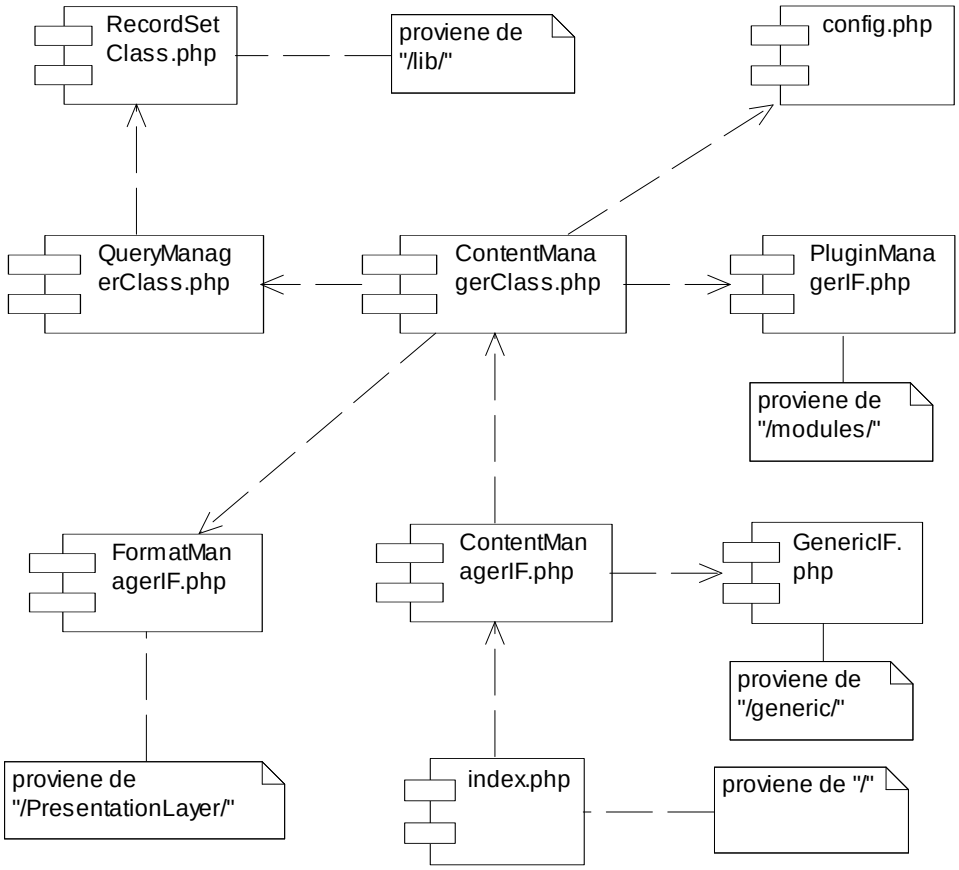

Fig. 2.4.1-40: Modelo de implementación de NavigateSystem/

2.4.1.6.5. Modelo de implementación de modules/

El modelo de implementación de “modules” se describe mediante la siguiente tabla:

TABLA 2.4.1-4: Modelo de implementación de *modules/*

Archivo/Carpeta	**Descripción:**
ModuleSystem/	Carpeta con los programas que implementan la carga y ejecución de los módulos.
messenger/	Módulo de prueba que envía mensajes simples.
modules-list-1.00.00.xml	Archivo con los módulos y sus dependencias de archivos.

Nota: cada módulo nuevo se incorporará a esta carpeta creando una carpeta nueva donde contendrá sus componentes.

Las dependencias son como siguen:

MODELO DE IMPLEMENTACIÓN DE *MODULES/*

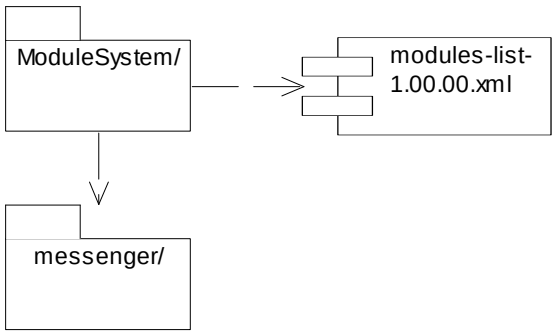

Fig. 2.4.1-41: Modelo de implementación de modules/

Modelo de implementación de *messenger/*

Únicamente contiene un componente: *messenger.php* que implementa una función que envía mensajes simples con un título, los parámetros de entrada se ajustan a la norma *module-arguments-1.00.00.txt* y la salida es un XML con formato *modules-response-2.00.00.dtd*. No tiene dependencias.

Modelo de implementación de *ModuleSystem/*

Contiene los siguientes elementos:

TABLA 2.4.1-5: Modelo de implementación de *ModuleSystem/*

Archivo/Carpeta	Descripción:
config.php	Archivo de configuración con la información básica de las carpetas, contraseña de la base de datos e información general del sistema.
ModuleLoaderClass.php	Implementa la clase ModuleLoaderClass.
PluginManagerIF:	Implementa la interfaz PluginManagerIF.

La dependencia es como sigue:

MODELO DE IMPLEMENTACIÓN DE *MODULESYSTEM/*

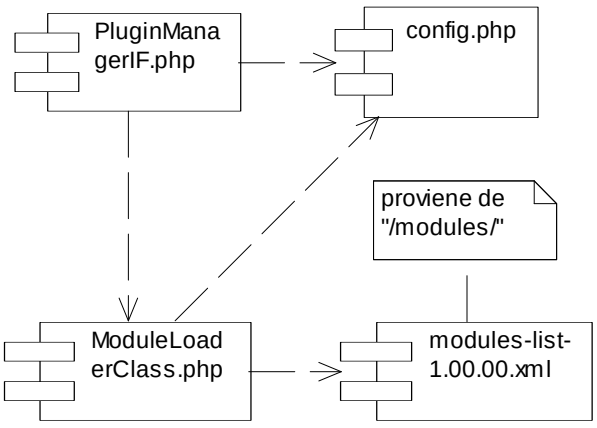

Fig. 2.4.1-42: Modelo de implementación de ModuleSystem/

2.4.1.6.6. Modelo de implementación de PresentationLayer/

La carpeta "*PresentationLayer/*" que contiene la implementación de la capa de presentación de la información (véase modelo de diseño), tiene el modelo de implementación descrito en la siguiente tabla:

TABLA 2.4.1-6: Modelo de implementación de *PresentationLayer/*

Archivo/Carpeta	**Descripción:**
config.php	Archivo de configuración con la información básica de las carpetas, contraseña de la base de datos e información general del sistema.
FormatManagerIF.php	Implementación de la interfaz FormatManagerIF.
FormatManagerClass.php	Implementación de la clase FormatManagerClass.php
DisplayerClass.php	Implementación de la clase DisplayerClass.php
FormatManagerClass.xml	Archivo de configuración del componente FormatManagerClass.php
Content/	Web-models: paquete: Content
Form/	Web-models: paquete: Form
Item/	Web-models: paquete: Item
Label/	Web-models: paquete: Label
Menu/	Web-models: paquete: Menu
Model/	Web-models: paquete: Model
Page/	Web-models: paquete: Page
Section/	Web-models: paquete: Section
templates/	Plantillas HTML

Nota: los paquetes de web-models siguen estas dos reglas:

1. Cada clase en un archivo.
2. El nombre del archivo es el nombre de la clase.

Las dependencias de "*PresentationLayer/*" se muestran a continuación:

MODELO DE DISTRIBUCIÓN DE *PRESENTATIONLAYER/*

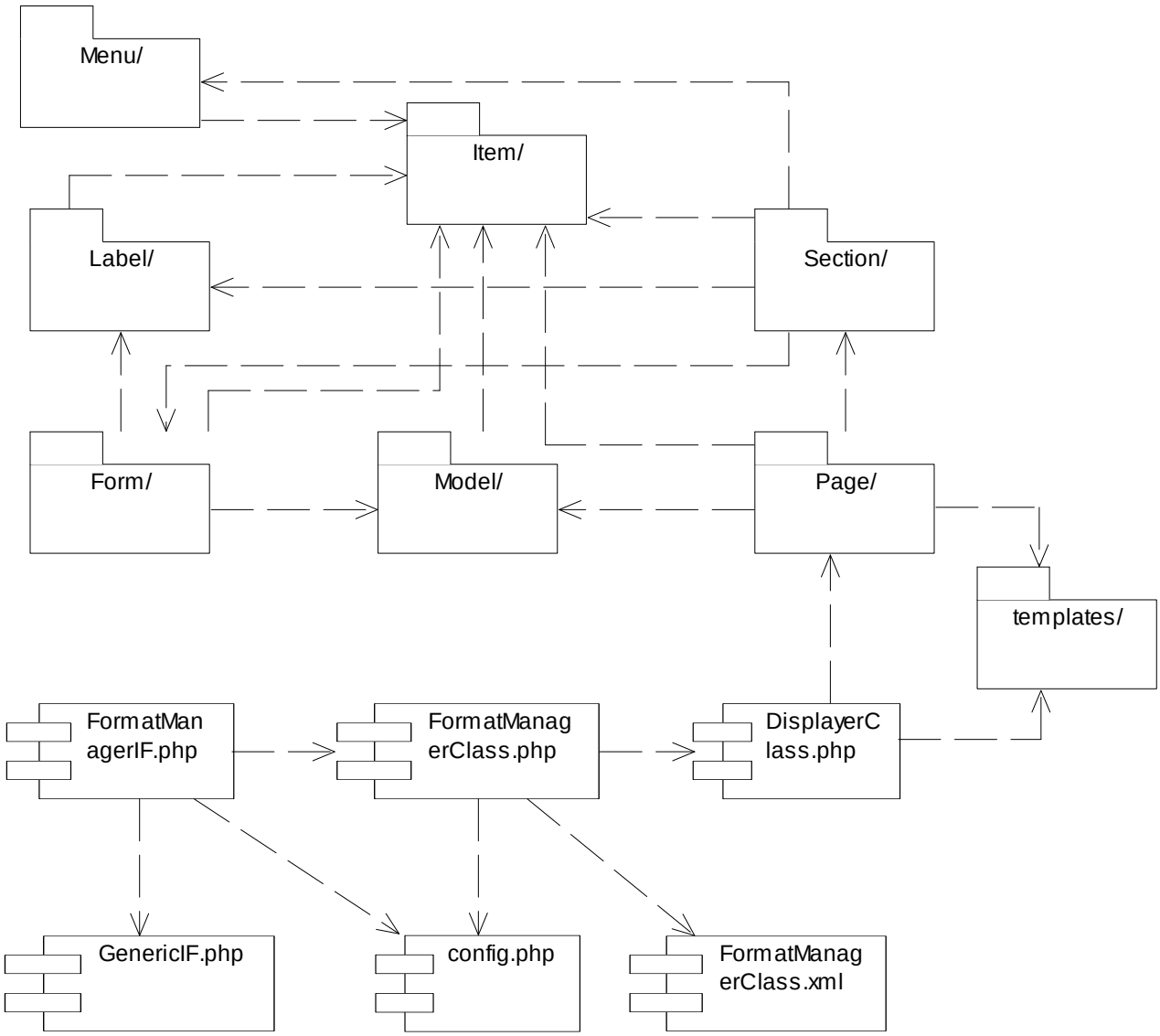

Fig. 2.4.1-43: Modelo de distribución de PresentationLayer/

2.4.2. Primera Iteración

2.4.2.1. Introducción

La fase de elaboración, para este proyecto se tuvo en dos iteraciones; a través de dichas iteraciones se exploraron las posibilidades y las alternativas de diseño que garantizaran la satisfacción de los requisitos del cliente.

Se consideraron además criterios tales como la flexibilidad a los cambios en los requisitos (robustez), independencia y consistencia de los componentes (alta cohesión y bajo acoplamiento) y la separación de los datos y el formato.

En resumen las iteraciones se realizaron de la siguiente forma:

1. Primera iteración: fue de carácter exploratorio, en ella se construyeron y probaron prototipos aislados de partes representativas del sistema y se explotó el recurso a un *patrón genérico* de comportamiento que se había observado en proyectos anteriores. En esta iteración se diseñó la primera versión consistente de la base de datos. Finalmente, a través de las pruebas y de sus limitaciones se obtuvo la información suficiente para plantear el problema de la integración general del sistema.
2. Con la información obtenida de la primera iteración, se hizo un análisis de riesgos centrado en la arquitectura. Se procedió después a atender estos riesgos. En particular, se reestructuraron todos los modelos. La aportación más importante de esta iteración es que a partir de los riesgos y de los requisitos cambiantes del cliente se planteó un diseño capaz de soportar la integración de todos los casos de uso planteados e incluso de soportar requisitos añadidos por el cliente bajo ciertas limitaciones. Al terminar la segunda iteración la arquitectura queda lo suficientemente estable para que los desarrolladores se concentren en los casos de uso sin preocuparse del entorno.

2.4.2.2. Desarrollo de la primera iteración

2.4.2.2.1. Planteamiento de los requisitos

Con respecto a la fase de inicio (inception) se realizaron unas modificaciones en cuanto a la descripción de los actores, la forma en que quedaron al final de esta iteración es la siguiente:

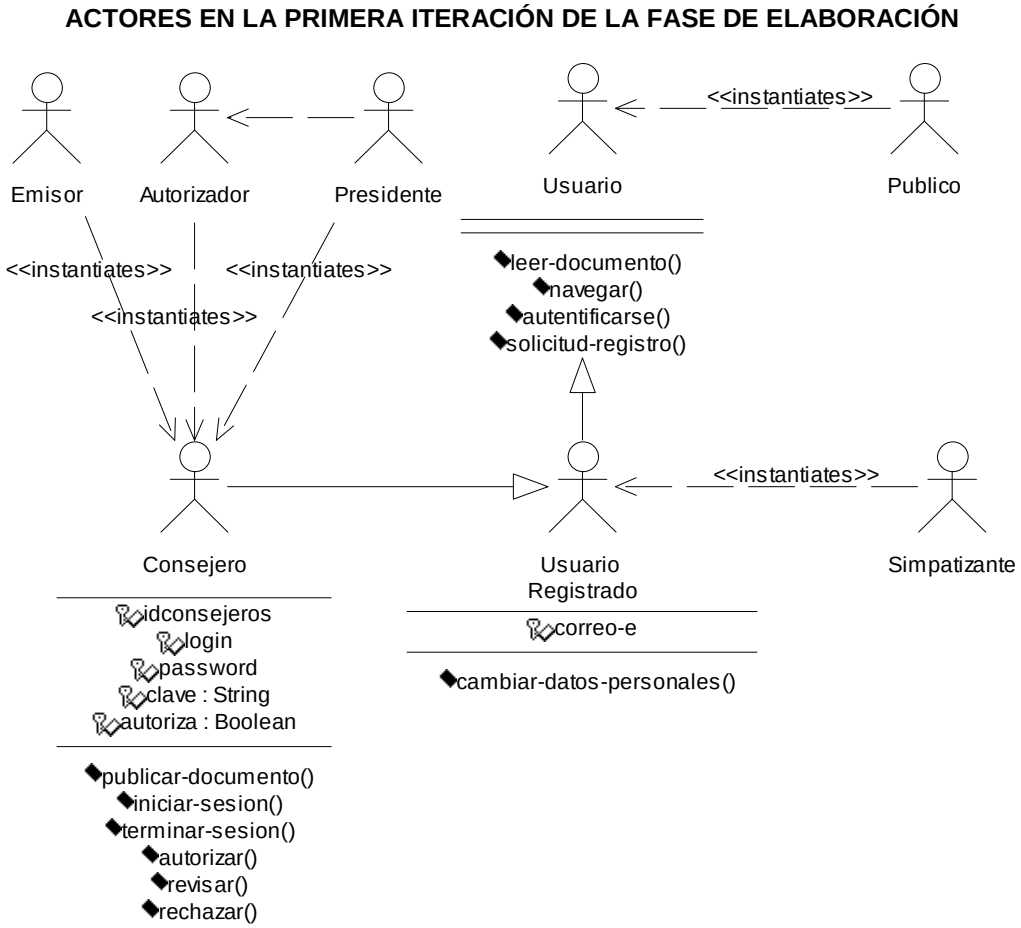

Fig. 2.4.2-1: Diagrama de explicación de los actores en la primera iteración de la fase de elaboración

Explicación:

- Hay un usuario en términos generales, este usuario puede ser público (no está registrado) o puede estar registrado.
- El registro puede ser de dos maneras: como simpatizante (que deje su correo electrónico para recibir actualizaciones del sitio, invitaciones a eventos, etc.) o como consejero (login, password).
- Hay un tipo especial de consejero que es el autorizador, el cual autoriza documentos, temas, etc. El presidente es consejero por antonomasia.
- El emisor es un rol que todo consejero asume cuando publica un documento.
- Este esquema tiene diversas ventajas respecto del anterior, pues centra el actor en su naturaleza más que en los roles que puede adquirir, sin embargo no los desconoce, sino que los considera como *instancias* de la clase. Además evita las redundancias que se daban en el diagrama de la fase de inicio.

El diagrama de casos de uso también se modificó ligeramente y quedó de la siguiente manera:

CASOS DE USO EN LA PRIMERA ITERACIÓN DE LA FASE DE ELABORACIÓN

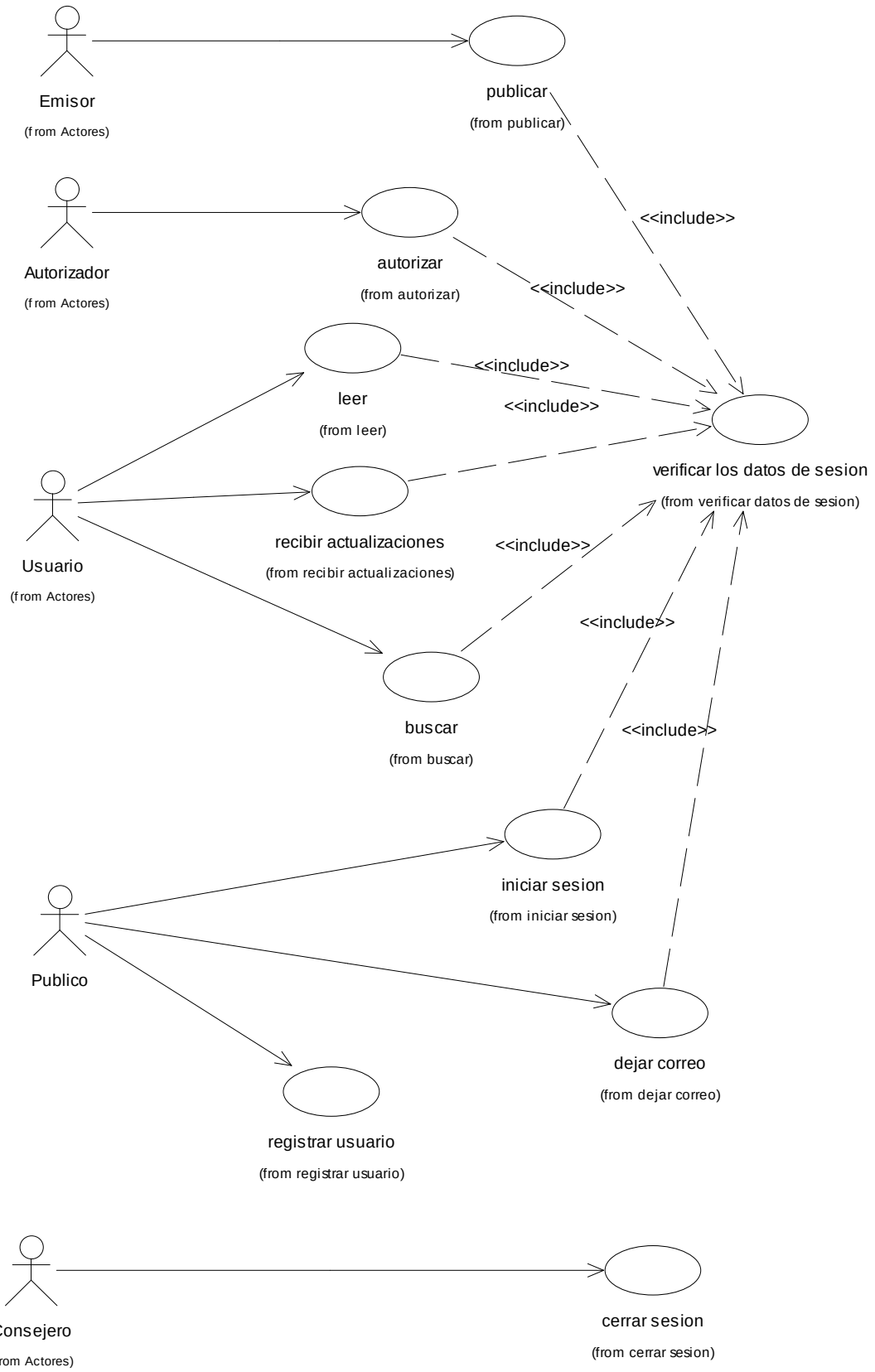

Fig. 2.4.2-2: Diagrama de casos de uso en la primera iteración de la fase de elaboración

Explicación:

- **Publicar:** significa publicar un documento. Por documento se entiende un archivo, un enlace, etc. Por motivos de pruebas de prototipo se considera provisionalmente únicamente un archivo.
- **Autorizar:** autorizar o rechazar un documento, estableciendo el nivel de confidencialidad que tiene.
- **Leer:** (navegar) significa ver los documentos disponibles en un tema, navegar a través de los temas y descargar un documento en particular.
- **Recibir actualizaciones:** cuando un usuario ha dejado su correo electrónico puede recibir periódicamente información acerca de los cambios más recientes en el sitio.
- **Buscar:** realizar una búsqueda de documentos y temas mediante palabras clave.
- Iniciar sesión: autenticarse ante el sistema para que el sistema reconozca al usuario como consejero.
- **Dejar correo:** esta opción permite llevar un registro de usuarios que no son consejeros para que puedan recibir actualizaciones del sitio.
- **Registrar usuario:** un usuario que quiera tener cuenta en el sitio deja sus datos en espera de ser autorizado.
- **Cerrar sesión:** cuando un consejero deja de utilizar el sistema puede cerrar la sesión.

Este diagrama significa dos cambios, por una parte añade la funcionalidad de las actualizaciones de correo electrónico y del inicio y fin de sesión y por otra parte quita "*mostrar menú*" y "*modificar datos personales*".

El caso de "*mostrar menú*" se omite porque es parte de la mayoría de los casos de uso. El caso de "*modificar datos personales*" se considera para otra fase de desarrollo.

De entre todos los casos de uso se seleccionaron los más importantes para intentar implementarlos y así obtener la arquitectura, el resultado fue el siguiente diagrama:

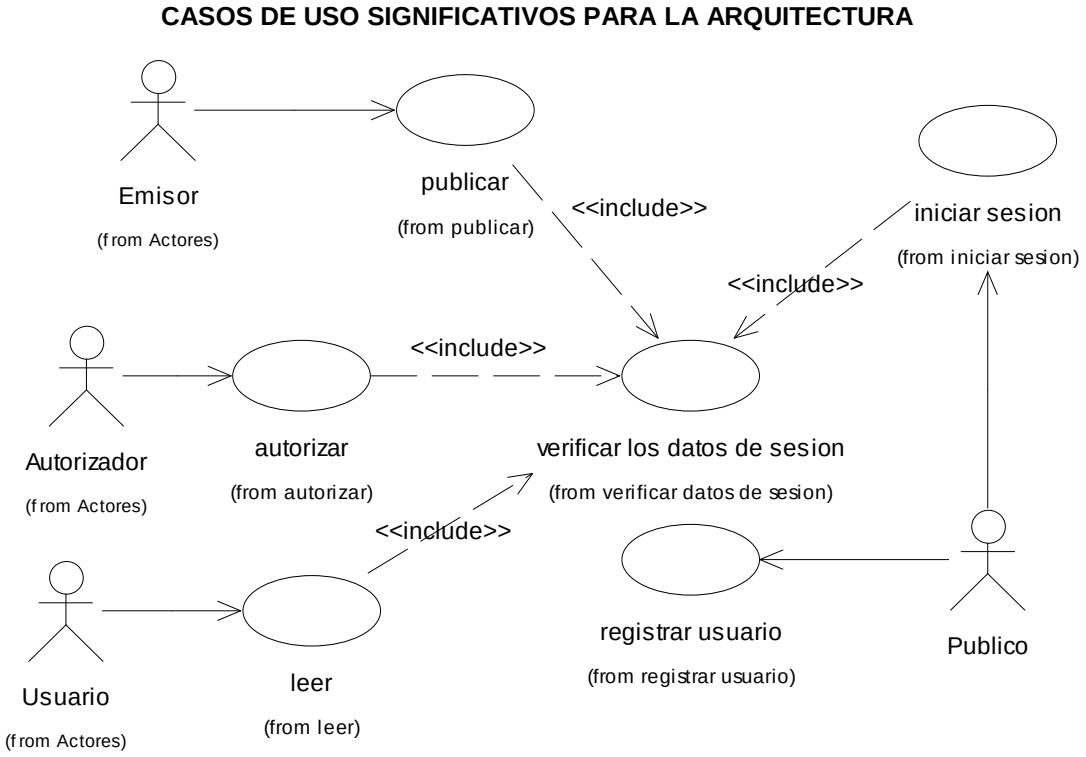

Fig. 2.4.2-3: Casos de uso significativos para la arquitectura (primera iteración de la fase de elaboración)

Por tanto la agenda de prototipos a probar fue como sigue:

1. registrar usuarios nuevos (por lo pronto que queden los usuarios autorizados automáticamente).
2. iniciar, cerrar y verificar sesión.
3. crear temas (para poder publicar documentos deben existir temas dónde asignar la publicación).
4. publicar (por lo pronto que queden los documentos autorizados automáticamente).
5. leer (navegar y mostrar menú y eso).
6. autorizar (entonces desactivar lo de autorización de documentos automática).

2.4.2.2.2. Planteamiento de seguridad

Este esquema trajo consigo un planteamiento de seguridad que había de resolver con el cliente; viendo que para efectos de la comunicación entre los diferentes miembros de la CMDH no era necesario tener 4 niveles de confidencialidad, entonces se determinó que sólo hubiera dos niveles: público y privado. En el nivel público cualquiera podía tener acceso a la información, mientras que en el nivel privado sólo los consejeros. Con esto se logra un objetivo del sistema: garantizar la comunicación interna de los consejeros de la CMDH.

En cuanto al análisis y diseño de múltiples tipos de usuarios, e.g. simpatizantes, etc. se optó por centrarse en los más importantes, y en acuerdo con el cliente se modeló el sistema para consejeros y público en general; dejando para después la inclusión de simpatizantes mediante correo electrónico.

2.4.2.2.3. Planteamiento de la base de datos

El modelo de la base de datos que se obtuvo en esta iteración tuvo como fundamento los siguientes criterios:

1. El sistema almacena documentos, los cuales pueden ser públicos o privados, y pueden estar autorizados o estar pendiente su autorización. Igualmente hay una fecha de publicación.
2. El sistema maneja cuentas de usuarios, los cuales publican documentos, inician y cierran sesión mediante contraseña.
3. El sistema acomoda su información (documentos) en un árbol de temas de navegación para que puedan ser catalogados los documentos y localizados con facilidad.
4. Se debe proveer un sistema de búsqueda que por lo pronto sea sencillo de implementar.
5. Algunos de los usuarios registrados tienen facultades para crear temas, autorizar documentos, etc. (autorizadores).
6. Se buscó que fuera mínimo el modelo principalmente para probar prototipos que aportaran más información; en particular, omitir campos innecesarios.

El modelo de objetos de los datos queda así:

MODELO DE DATOS EN LA PRIMERA ITERACIÓN DE LA FASE DE ELABORACIÓN

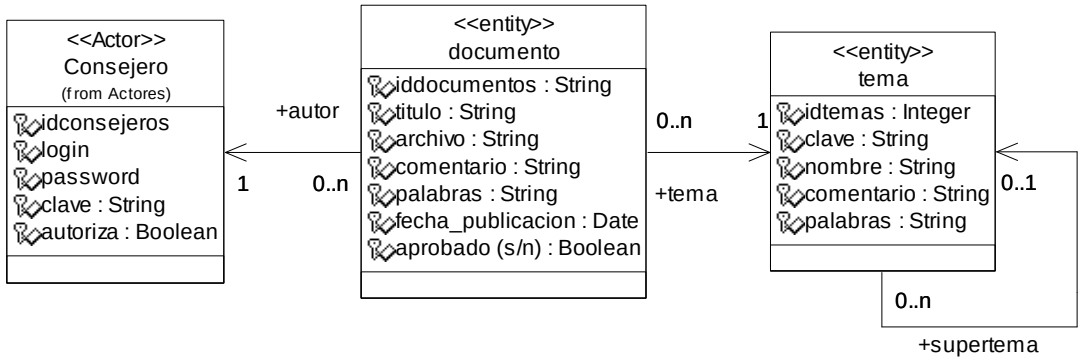

Fig. 2.4.2-4: Modelo de datos en la primera iteración de la fase de elaboración

Como puede notarse en la clase *consejero* hay un campo *password* y otro campo *clave*, este error se advirtió después y se corrigió.

La base de datos se implementó de manera aún más sencilla utilizando un mínimo de atributos para cada tabla con el propósito de realizar pruebas.

Cada tabla fue creada a medida que se implementaban los casos de uso: registrar usuario, crear tema, publicar documento.

2.4.2.2.4. Realización de la agenda de los prototipos

Para realizar los prototipos y así poder elaborar la arquitectura, al menos en su primera versión se estableció el siguiente orden:

1. Patrón genérico para sistemas Web.
2. Registrar usuario.
3. Crear tema.

4. Sesiones
5. Publicar documento
6. Navegar.
7. Autorizar.
8. Correcciones y ajustes.

De esta agenda se lograron construir prototipos parciales hasta "*navegar*". Este último muy incipiente y se decidió cerrar la iteración con la información obtenida.

2.4.2.2.5. Patrón genérico para sistemas Web

Este patrón fue esbozado en la fase de inicio y en esta fase se hicieron algunos ajustes al análisis de manera que quedó planteado de la siguiente manera:

CLASES DE ANÁLISIS DEL *PATRÓN GENÉRICO PARA SISTEMAS WEB*

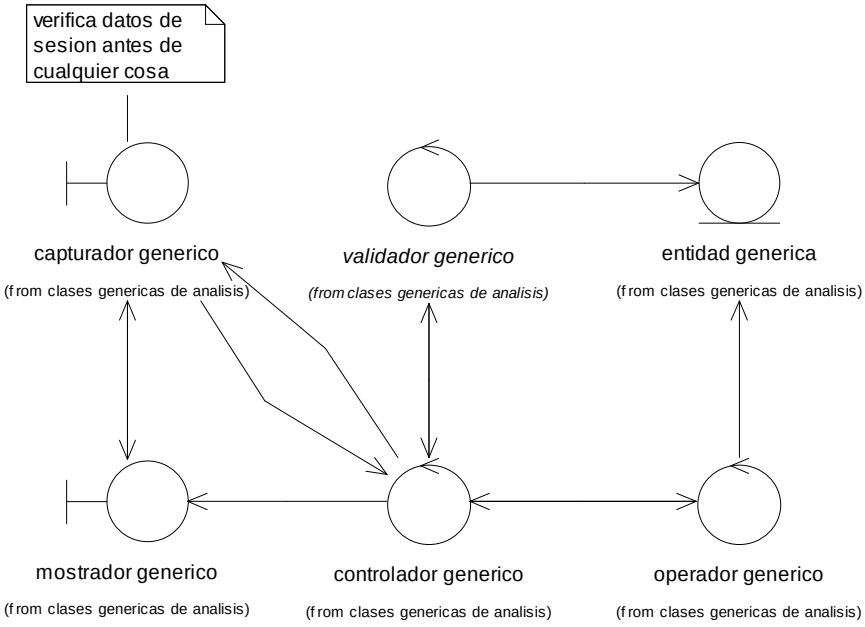

Fig. 2.4.2-5: Clases de análisis del patrón genérico para sistemas Web *(primera iteración de la fase de elaboración)*

Explicación:

- **Capturador genérico:** clase que muestra el formulario de captura de datos, también sirve para mostrar errores en la captura y corregir los datos.
- **Controlador genérico:** clase que controla el flujo de la realización del caso de uso, y toma las decisiones de acuerdo a los resultados entregados por cada clase, e.g. el manejo de los errores.
- **Validador genérico:** clase que valida un formulario Web con base en ciertas reglas.
- **Operador genérico:** clase que realiza la operación con la base de datos.
- **Entidad genérica:** representa cualquier entidad o grupo de entidades de datos persistentes, como tablas de bases de datos, archivos, *cookies*, etc.
- **Mostrador genérico:** clase que muestra resultados.

La secuencia típica de ejecución de este patrón se muestra a continuación:

SECUENCIA DEL *PATRÓN GENÉRICO*

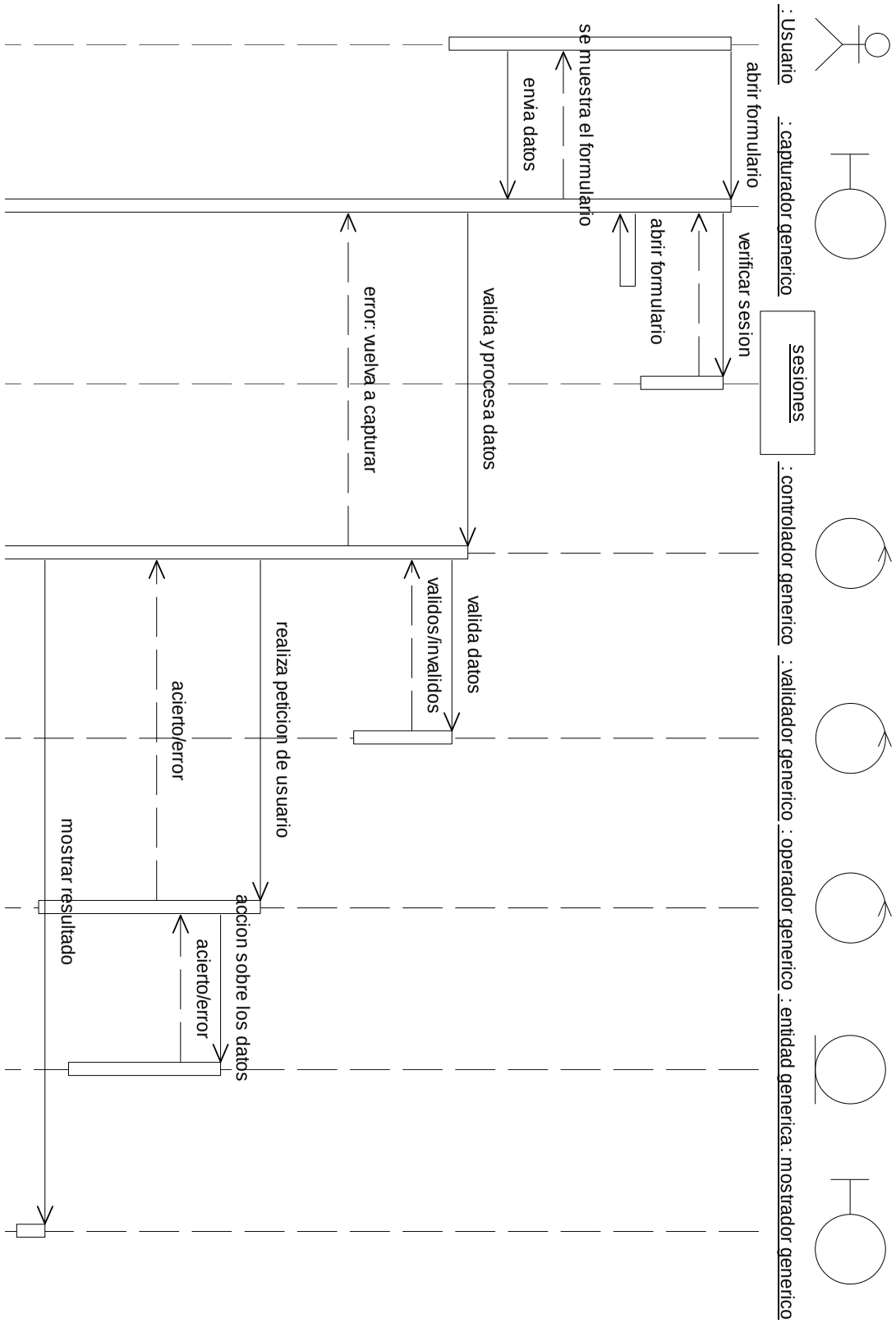

Fig. 2.4.2-6: Secuencia del patrón genérico

Explicación:

1. El usuario ingresa al sistema (*capturador genérico*).
2. El sistema verifica sesión.
3. Si el usuario está autorizado entonces el sistema muestra al usuario el formulario de captura/modificación de información (*capturador genérico*).
4. El usuario captura y envía la información al sistema.
5. La clase *controlador genérico* recibe la información del formulario.
6. El controlador genérico solicita al *validador genérico* que verifique la información.
7. Si la información es inválida entonces el *controlador genérico* solicita al *capturador genérico* vuelva a mostrar el formulario con un informe de los errores de captura.
8. Si la información es correcta, el *controlador genérico* solicita al *operador genérico* realice la petición del usuario.
9. El *operador genérico* inserta/modifica/borra/consulta los datos en la base de datos y devuelve el resultado al *controlador genérico*.
10. El *controlador genérico* solicita al *mostrador genérico* que le muestre al usuario los resultados de su petición.

En esta fase se indagaron los alcances de este planteamiento, para ello se diseñó e implementó este patrón y se aplicó a algunos casos de uso.

El *diseño genérico* resultó de la siguiente manera:

DIAGRAMA DE CLASES DE DISEÑO DEL *PATRÓN GENÉRICO*

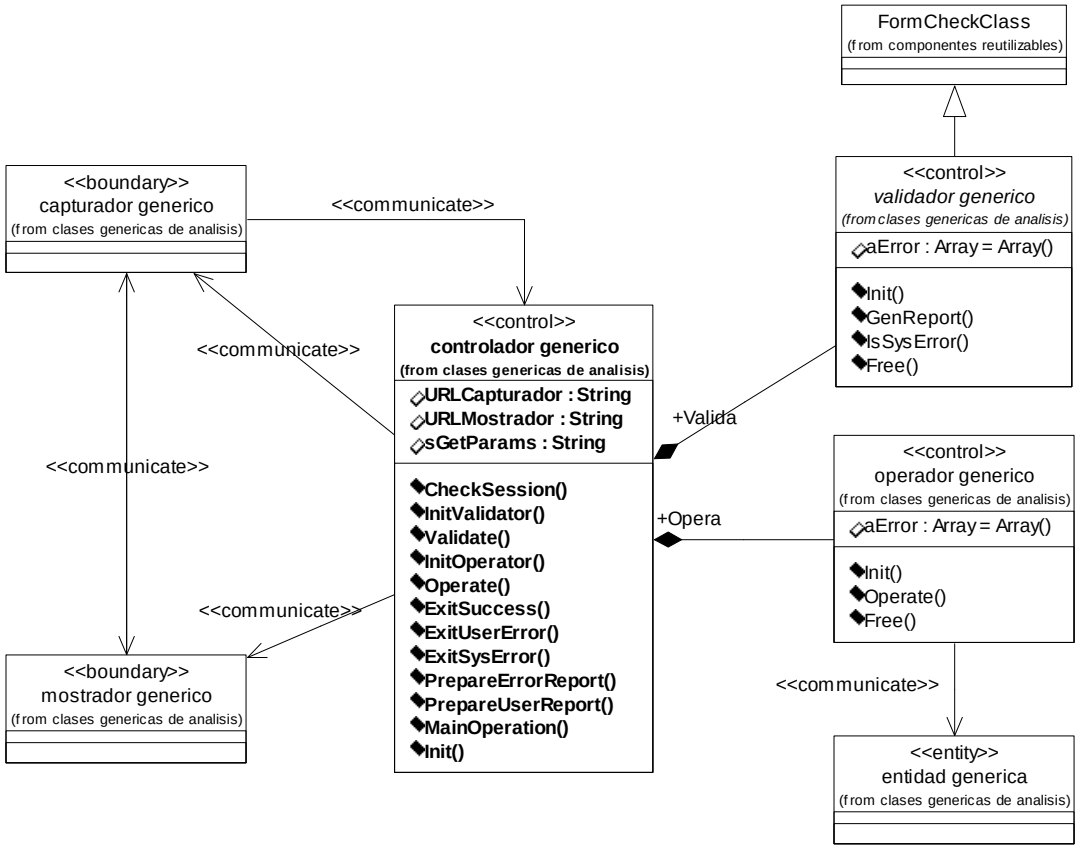

Fig. 2.4.2-7: Diagrama de clases de diseño del patrón genérico

A grandes rasgos cabe señalar que este modelo tiene dos *URL's*: una para el *capturador* y otra para el *mostrador*, se considera que el sistema se ejecuta en tres partes: la primera, para el *capturador genérico* (con su propia URL). La segunda para el *controlador genérico* (con su URL como destino del formulario) y la tercera para el *mostrador genérico* (con su propia URL).

La clase *FormCheckClass* es una clase para validar formularios en general que ya se tenía y que sólo se mejoró y se heredó para características de base de datos, y generación de reportes adecuados al *mostrador genérico*.

Para la codificación, tanto el *mostrador genérico* como el *capturador genérico* no utilizaron clases, esto provisionalmente para ahorrar tiempo y concentrarse en el resto del funcionamiento.

Registrar usuario

Esta realización de caso de uso utilizó el patrón genérico tanto en el análisis como en el diseño, de tal forma que sólo se usaron clases heredadas en lugar de usar las clases genéricas.

La herencia se muestra a continuación:

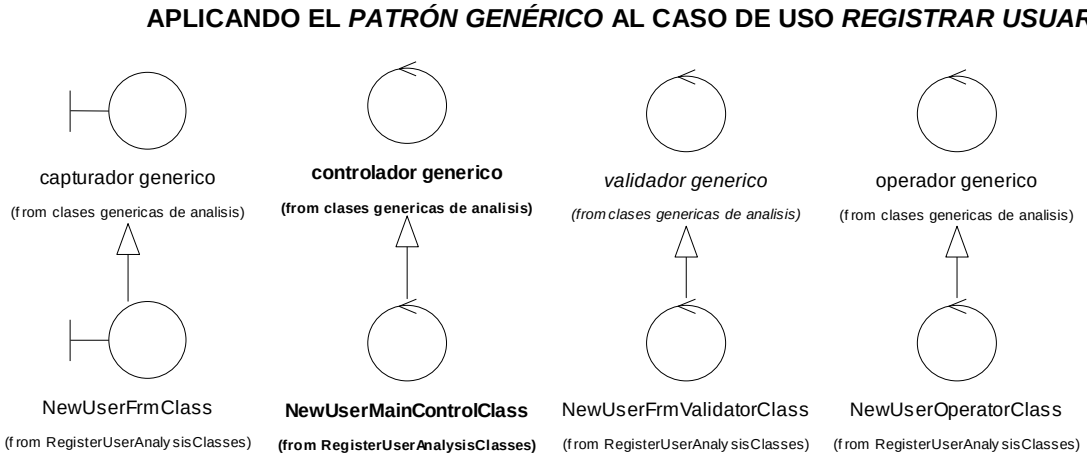

Fig. 2.4.2-8: Herencia para aplicar el patrón genérico al caso de uso registrar usuario

El *mostrador genérico* no necesitó heredarse. La herencia en el *capturador genérico* es sólo en el análisis y el diseño, pues en la implementación provisionalmente se utilizó un *script* para probar el funcionamiento de esta parte del sistema.

El diagrama de clases del análisis queda igual que en el *patrón genérico* pero con las clases heredadas en lugar de las originales y en lugar de la clase *entidad genérica* se utiliza la clase *consejero*.

En cuanto al diseño las únicas modificaciones son que para comunicar a la clase *NewUserOperatorClass* con la clase *consejero.* Se utiliza la clase *RecordSetClass* que es una clase genérica para hacer consultas SQL que ya estaba implementada y a la cual se le hicieron correcciones menores. También se definieron los datos de intercambio en el formulario *NewUserFrmClass*.

La codificación se realizó de manera similar al patrón genérico y se requirió del mismo para herencia de clases; esta primera implementación tomó mucho tiempo en relación al esperado

y las primeras pruebas consumieron también mucho tiempo en la corrección de errores. Pero finalmente funcionó.

Crear tema

Este caso de uso también se apoyó en el *patrón genérico* y la herencia, de manera análoga al caso de uso *registrar usuario* fue como sigue:

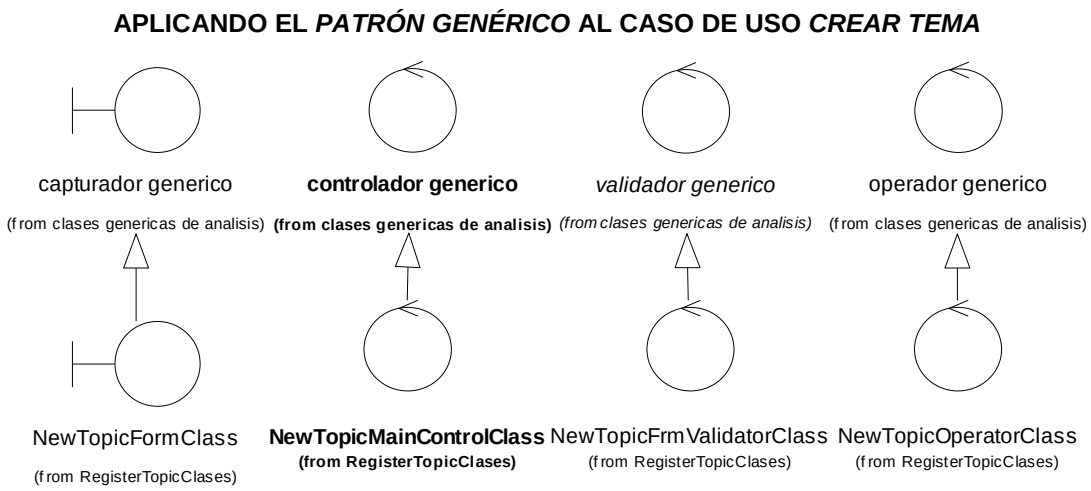

Fig. 2.4.2-9: Herencia para el caso de uso crear tema *en la primera iteración de la fase de elaboración*

El diseño se realizó de manera análoga a *registrar usuario* pero con la clase *tema* en lugar de la clase *entidad genérica*.

La implementación se realizó de manera análoga, la diferencia fue que el formulario realizaba una consulta a la base de datos para obtener una lista de temas para asignar como *supertema* o *tema superior*.

En comparación con la implementación del prototipo del caso de uso *registrar usuario*, *crear tema* se realizó a una gran velocidad, pero aún así no estaba exento de errores ni carecía de complejidad.

Sesiones

En cuanto a las sesiones se tuvieron cuatro actividades:

1. investigar el funcionamiento de las sesiones en general
2. determinar una estrategia para controlar y utilizar las sesiones de manera segura
3. construir un componente para manejo de sesiones
4. la realización del caso de uso *iniciar sesión* a manera de prototipo.

Sesiones en general

Se investigaron 3 formas de manejar las sesiones en PHP4 y son:

1. mediante base de datos.
2. mediante *cookies*
3. mediante *variables de sesión*

Explicación:

- **Base de datos:** la ventaja es que se garantiza que el servidor controla la información de la sesión, la desventaja es que si se interrumpe la conexión queda mucha información basura, además de que se reduce el rendimiento (*performance*) del sistema. También tiene un costo elevado de implementación.
- **Cookies:** tiene la ventaja de que es muy fácil de implementar, pero la desventaja de que está del lado del cliente, ofreciendo un riesgo potencial en la seguridad.
- **Variables de sesión:** es una implementación de PHP en la cual el PHP genera una *cookie* con un identificador único para un archivo (del lado del servidor) en el cual se almacenan las variables de sesión. En caso de que el usuario tenga desactivadas las *cookies* de su navegador, entonces el PHP genera un parámetro GET con el ID de la sesión para todos los enlaces en el sitio Web de forma transparente para el programador.

Dadas las condiciones se eligió usar la funcionalidad de sesiones de PHP pues el mismo lenguaje de programación previene muchas dificultades de manera transparente para el desarrollador. La limitante sería que un mismo usuario podría tener varias sesiones a la vez, esta limitante no es tan grave, si se garantiza que no haya robo de sesión o de contraseña y que una sesión válida sea iniciada únicamente mediante un nombre de usuario (*login*) y contraseña (*password*) válidos.

Estrategia para sesiones seguras

Una cosa es poder almacenar datos de sesión de los usuarios, y otra es garantizar la seguridad, en particular, que ésta información permanece confidencial. Asimismo que el acceso a las zonas restringidas del sistema sea únicamente por aquellos usuarios autorizados.

La pregunta a responder es: ¿de qué forma se puede identificar al usuario mediante variables de sesión y al mismo tiempo, un usuario que pueda acceder a las variables de sesión no pueda falsificar (después de cerrada la sesión) la identidad de nadie?

Para ello se propusieron las siguientes variables:

1. **login:** para identificar al usuario.
2. **key:** clave especial de seguridad.
   `key=md5(login.pasword)` donde "`.`" indica concatenación.
3. **fecha:** fecha y hora de la última vez que se accedió al sistema (este dato se actualiza cada vez que se verifica sesión).

Mediante la fecha se garantiza que de haberse perdido la conexión o de que el usuario haya olvidado accidentalmente cerrar su sesión, ésta caduque y no pueda ser utilizada después de un tiempo razonable para una suplantación.

MD5 (Message Digest version 5)[4] es un algoritmo criptográfico de firma digital que permite cifrar de una sola vía la información. Aunque no es imposible de romper se considera bastante seguro.

[4] Consulte la dirección http://rfc.net/rfc1321.html con la especificación del algoritmo por parte de su autor R. Rivest del MIT (Abril 1992).

La forma de identificar a un usuario sería:

1. El sistema lee la variable de sesión *login*.
2. El sistema verifica en la base de datos el usuario identificado con ese valor.
3. Si existe, se obtiene el valor del password, se obtiene en el servidor una *key* local (key = md5(login **.** password). Se compara con la variable de sesión *key*.
4. La sesión es válida si y sólo si ambas claves cifradas son iguales.

Si un usuario logra obtener las variables de sesión *login* y *key*, es muy difícil que pueda obtener la contraseña a partir de estos datos pues MD5 es un algoritmo de cifrado de una sola vía bastante robusto.

Este mecanismo de verificación de sesión debe ponerse en todas las páginas a las cuales se desea restringir el acceso o de las cuales se desea un comportamiento distinto para usuarios registrados y no registrados.

Componente de sesiones seguras

Este componente se diseñó y se construyó para funcionar en cualquier sitio Web con PHP versión 4 y MySQL.

El diagrama es el siguiente.

CLASE PARA EL MANEJO DE SESIONES SEGURAS

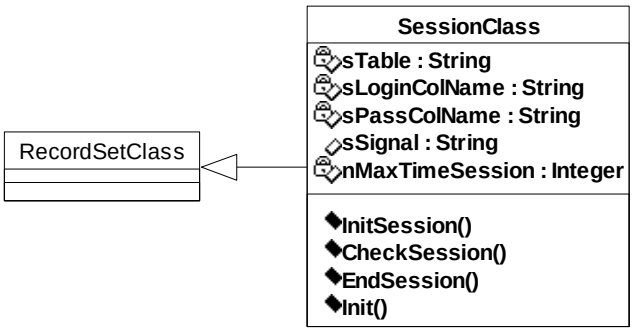

Fig. 2.4.2-10: Clase para el manejo de sesiones seguras

Esta clase está diseñada para que en su configuración se establezca el nombre de la tabla y las columnas *login* y *password*, opera con las variables de sesión *key, login* y *timestamp* (para la fecha y hora). El resultado a la hora de verificar sesión lo guarda en *sSignal*. Si hay error de base de datos se almacena en las propiedades heredadas de *RecordSetClass*.

Caso de uso iniciar sesión

El caso de uso iniciar sesión se basa en el *patrón genérico*, tanto para el análisis, como el diseño y la implementación; la herencia de las clases se muestra a continuación.

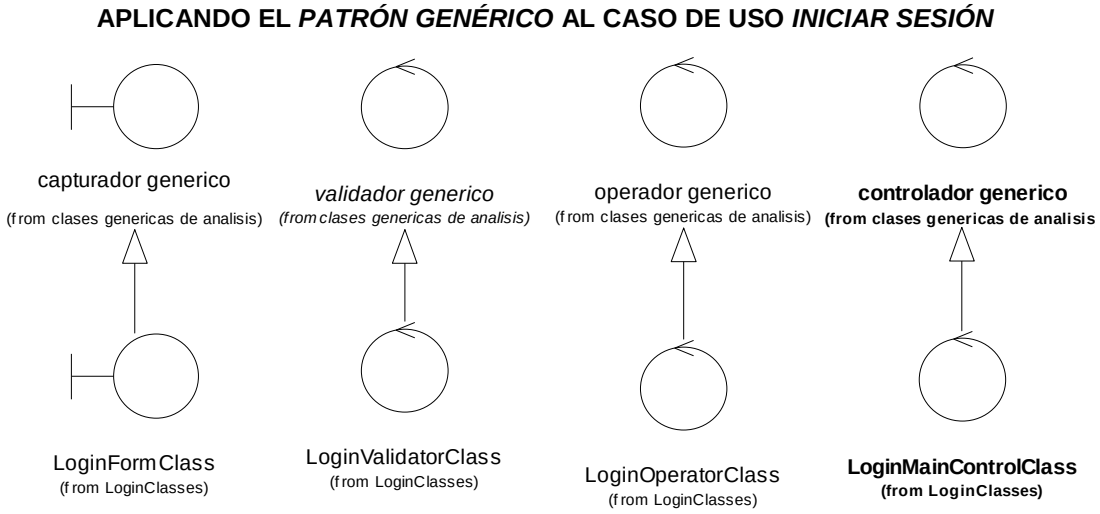

Fig. 2.4.2-11: Herencia de clases del caso de uso iniciar sesión *en la primera iteración de la fase de elaboración*

La diferencia con el patrón genérico de análisis es que la clase *LoginValidatorClass* se conecta con la clase *consejero* y la clase *LoginOperatorClass* utiliza a *SessionClass* para iniciar la sesión.

El diseño es de manera análoga al patrón genérico, con la característica de que el inicio de sesión se realiza utilizando la clase *SessionClass*.

Publicar documento

La realización de este caso de uso ofrecía varias dificultades, en particular, la forma en que un archivo es enviado a un servidor mediante formulario y su control simultáneo con el de la base de datos.

Otra de las preocupaciones es la de la seguridad, en particular, el mantener oculta la carpeta donde se colocan los archivos y los nombres de los mismos.

La estrategia que se siguió fue que todo archivo que se envíe al servidor cambiará su nombre a un número que es el de la clave primaria en la base de datos, esto ayudará a *ocultar* el archivo y también permitirá que puedan publicarse varios archivos con el mismo nombre de diferentes usuarios y que en realidad tienen contenido distinto (e.g. el usuario A envía el archivo *Informe.doc* y el usuario B envía otro archivo *Informe.doc* cuya información es distinta, ambos archivos son necesarios y no deben sobrescribirse).

Con estos requisitos en mente se llegó a la construcción de una clase que permitiera controlar los archivos enviados al servidor de una manera segura, esta clase se muestra a continuación:

FILEUPLOADCLASS

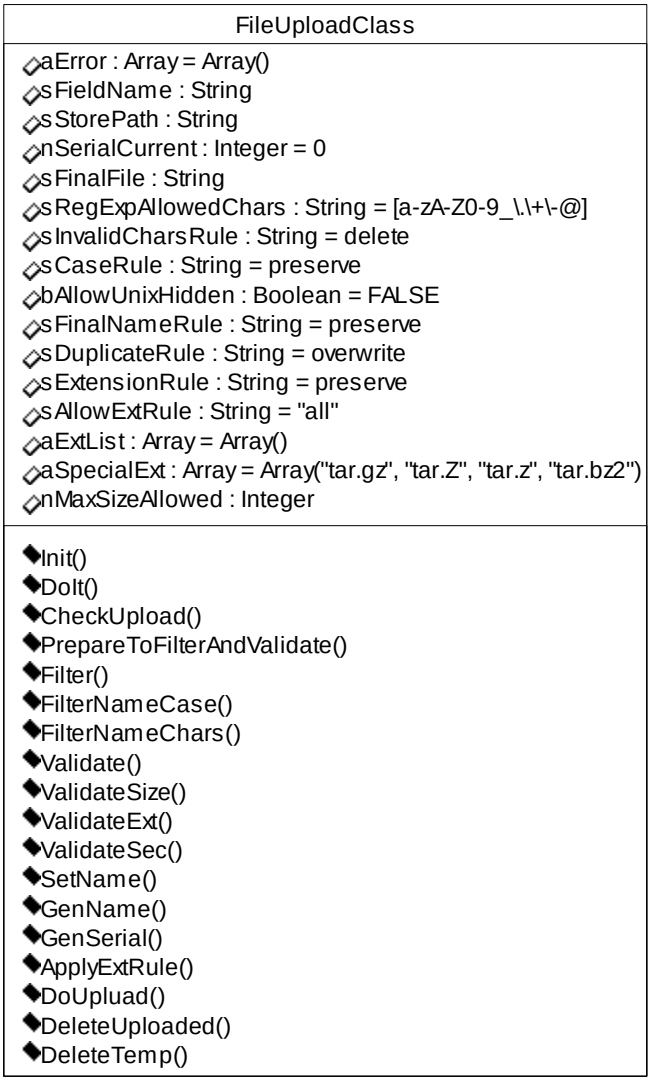

Fig. 2.4.2-12: FileUploadClass

Esta clase se describe posteriormente en detalle, baste decir que está basada en reglas, las cuales consideran los casos de *sobreescritura*, generación del nombre, extensiones permitidas, filtrado de nombre, extensiones permitidas, etc.

Esta clase quedó implementada en su mayor parte, quedando inconclusa la generación de números seriales, lo cual es más seguro generarlos mediante base de datos y asignar este valor mediante *SetName()*.

El análisis aunque basado en el patrón genérico no es tan trivial, pues hay un manejo simultáneo de datos de formulario y de archivos, así como de varias clases de entidad (base de datos) que se sincronizan para lograr el proceso de almacenar el archivo en el servidor.

La herencia de las clases es como sigue:

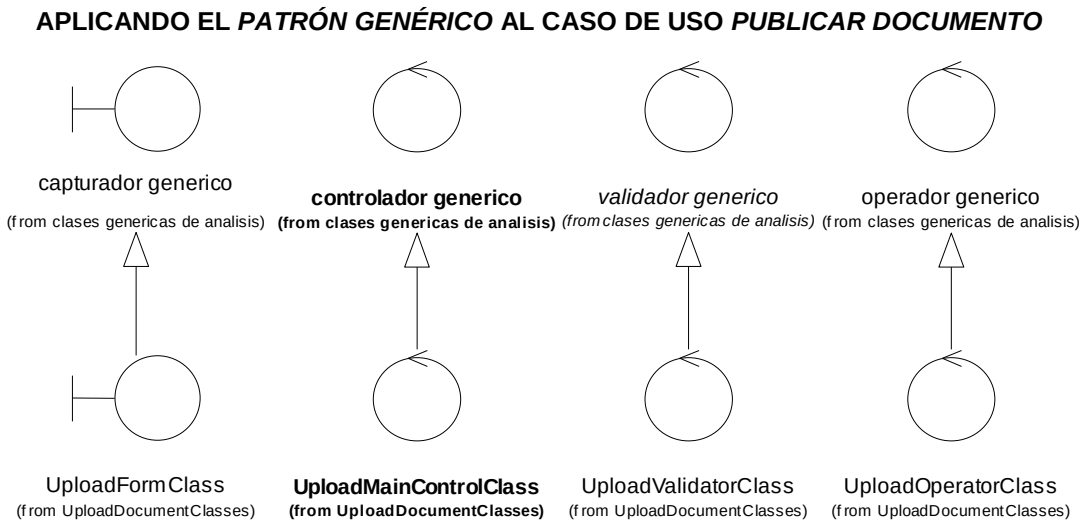

Fig. 2.4.2-13: Herencia de clases del caso de uso publicar documento *en la primera iteración de la fase de elaboración*

Y el diagrama de clases del análisis queda así:

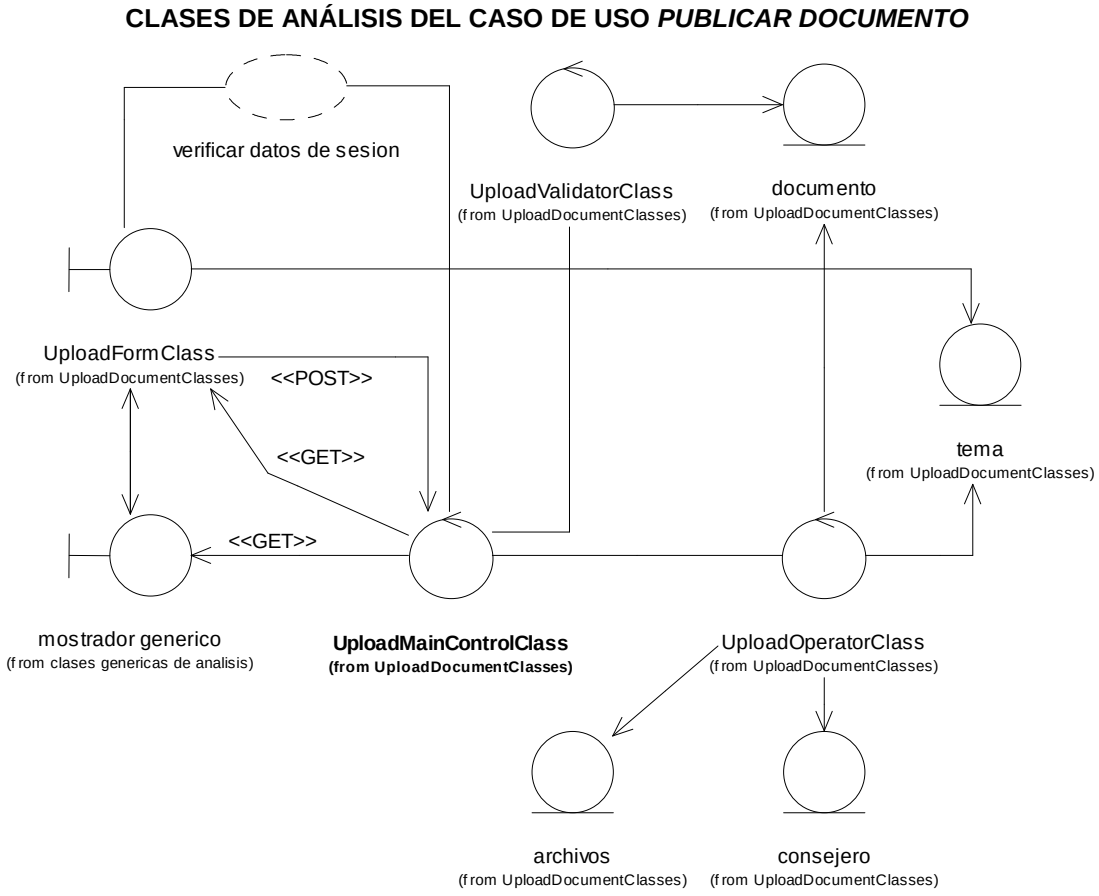

Fig. 2.4.2-14: Clases de análisis del caso de uso publicar documento *en la primera iteración de la fase de elaboración*

La clase *Archivos* representa a la carpeta (o el equivalente) donde se almacenan los archivos.

La secuencia es la misma que en el patrón genérico excepto para *UploadOperatorClass*, en donde primero se consulta a *consejero* y a *tema* para obtener la información que luego se insertará en *documento*. Después se procede a almacenar el archivo en la clase de entidad *archivos*. En caso de que haya algún error se borra tanto el archivo como el registro en la base de datos.

El diseño utilizó *RecordSetClass* para hacer las consultas SQL y usó *FileUploadClass* para colocar el archivo en la carpeta correcta con el nombre correcto.

La implementación tuvo muchas dificultades debido a la complejidad algorítmica y tomó más tiempo de lo esperado.

Navegar

En cuanto a la navegación se tuvieron varias dificultades, la primera de ellas fue tener una noción clara de lo que se entendía por *navegar*.

Hasta este momento el término *navegar* se había planteado de manera vaga y sólo encerraba la noción de usar la información almacenada en el sitio Web, pero a partir de aquí surgió la necesidad de plantearlo de manera más precisa; pues por navegar se entendían varias cosas:

- Mostrar los documentos asociados a un tema.
- Mostrar enlaces a los diferentes temas, subtemas, etc. y opciones de usuario (casos de uso).
- Mostrar o descargar el contenido de un documento.

Además cuando se ejecuta determinado caso de uso de todas formas, en la presentación de la información, se muestran los diferentes temas y opciones a las que el usuario puede ir (navegar).

Finalmente se debe de tomar en cuenta la información confidencial, pues debe ocultarse del usuario no autorizado dicha información.

En primera instancia se decidió indagar el alcance de mostrar los enlaces a los diferentes temas así como un listado de documentos en cada tema.

Para esto se construyó una clase capaz de realizar una serie de consultas a los temas y obtener determinada información de los mismos, así como listar los documentos en un determinado tema, esta clase se llamó *QueryManagerClass*, y se probó mediante un *script* muy sencillo que visualizaba a modo de listado un tema seleccionado, sus subtemas, el tema inmediato superior, los temas raíz y los temas paralelos; así como los documentos de dicho tema.

Entonces surge el problema ¿cómo presentar la información de manera definitiva? Y esta otra pregunta ¿Se puede reutilizar esta clase y otras para la construcción de la navegación de temas en los demás casos de uso?

La respuesta se planteó desde el análisis de la siguiente forma:

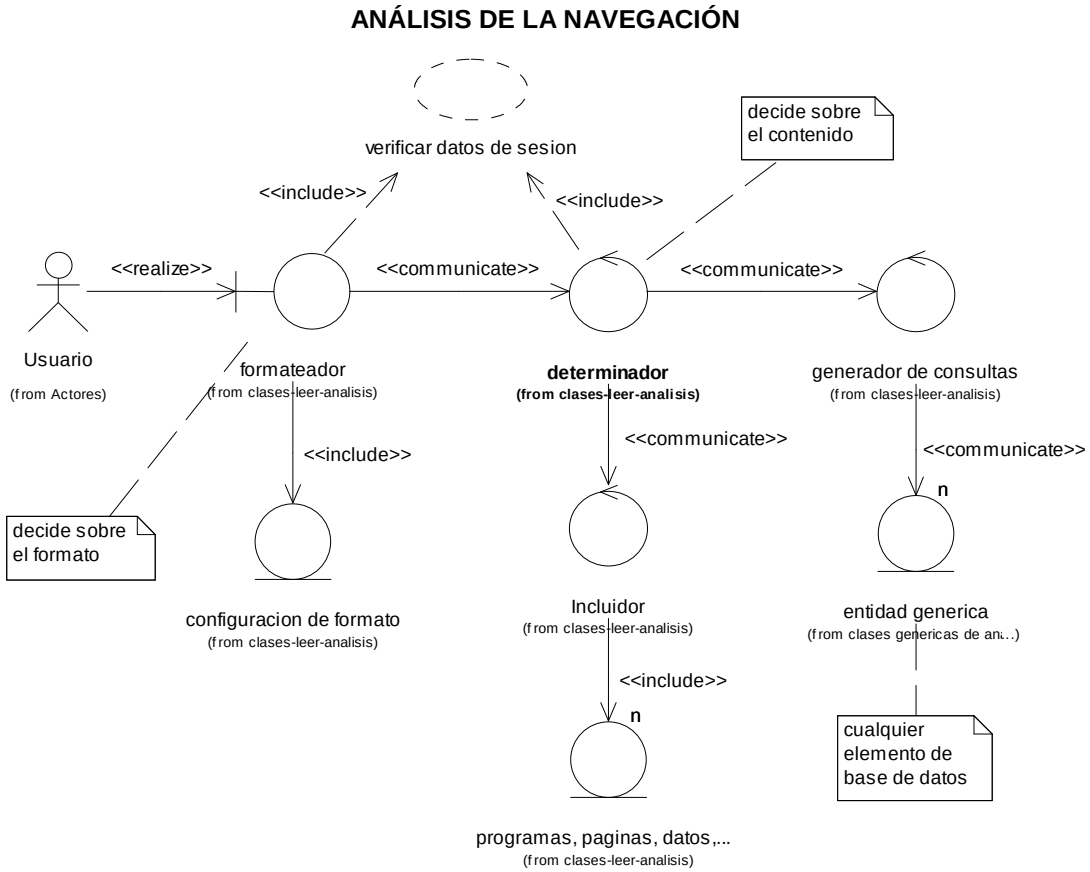

Fig. 2.4.2-15: Análisis de la navegación en la primera iteración de la fase de elaboración

Este modelo es muy similar al ya explicado en la descripción de la arquitectura.

La novedad con este análisis es que distribuye las responsabilidades de formato (*formateador*), contenido (*determinador*) y navegación (*generador de consultas*) para resolver estos problemas de manera independiente. Igualmente incluye la posibilidad de incluir cualquier caso de uso mediante el *incluidor* y darle el formato definitivo.

Se consideraba en ese momento que el listado de documentos (caso de uso de navegación todavía ambiguo) lo realizaría el *generador de consultas*.

El diseño se planteó desde el punto de vista de interfaces y subsistemas de la siguiente manera:

DISEÑO DEL CASO DE USO *NAVEGAR*

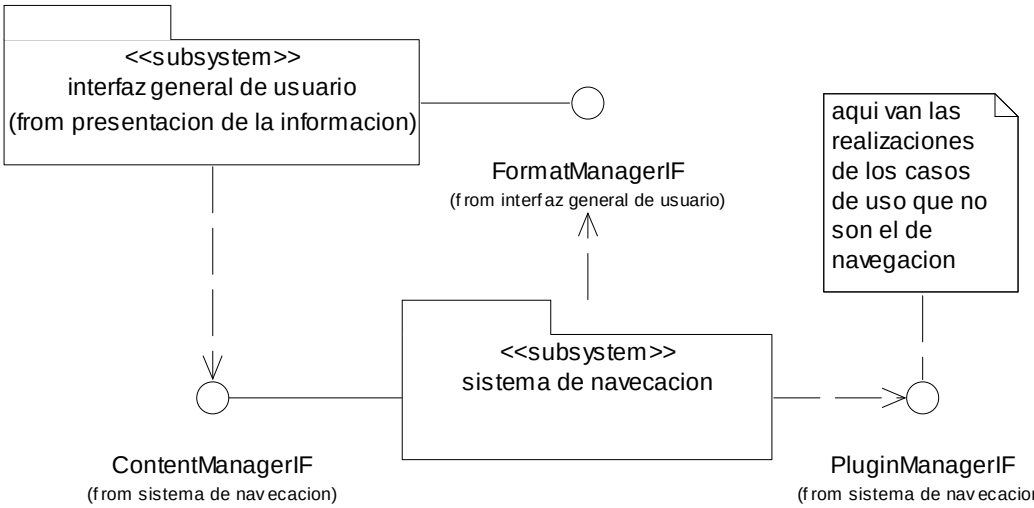

Fig. 2.4.2-16: Diseño del caso de uso navegar *en la primera iteración de la fase de elaboración*

Este diseño abarca mucho más que el caso de uso de la navegación en el sentido de listar documentos, pues abarca también a todos los casos de uso en la forma en que van a mostrar la información y van a mostrar las opciones que puede tener el usuario.

Se dejó abierta la posibilidad de que cualquier subsistema cargara e invocara al otro. Quedaba pendiente el diseño detallado, la implementación y las pruebas para realizar los ajustes necesarios.

Hasta este punto se dio por concluida la primera iteración, pues por una parte se tenía ya el bosquejo central (diseño de navegar) y por otra parte era necesario realizar los flujos de trabajo (requisitos, análisis, diseño, implementación, pruebas) otra vez para poner todo en orden, así como controlar los riesgos que se estaban generando.

2.4.3. Segunda Iteración

2.4.3.1. Introducción

La primera iteración tenía como finalidad explorar las diversas formas en que se podía construir la arquitectura y determinar el alcance de los modelos obtenidos hasta entonces.

En cambio la segunda iteración define la arquitectura en su forma definitiva. Los cambios posteriores a la arquitectura serán de pequeña magnitud, principalmente corrección de defectos que se hayan encontrado o ajustes para lograr mejores resultados.

Al finalizar la primera iteración se tenía bastante información del sistema, pero toda esa información se encontraba en desorden, por tanto, la segunda iteración estaría sujeta a un control más estricto que la primera mediante un plan por escrito.

2.4.3.2. El plan de la segunda iteración

Este plan formuló los siguientes objetivos para la iteración:

1. Obtener la descripción de la arquitectura.
2. Probar la línea base de la arquitectura.
3. Documentar la fase de elaboración.

Para ello se definieron varias etapas en el plan:

1. Etapa de revisión: se revisaron los artefactos producidos en la iteración anterior y en la fase de inicio.
2. Etapa de gestión de riesgos: se definió un plan de riesgos, se investigaron, definieron y documentaron los riesgos, finalmente se redefinió el plan de desarrollo de la segunda iteración de acuerdo a los riesgos encontrados.
3. Etapa de especificaciones: se hicieron las especificaciones de seguridad y de procedimientos de prueba.
4. Etapa de desarrollo de software: se asignaron prioridades de acuerdo con los riesgos y se hicieron los flujos de trabajo del desarrollo de software (análisis, diseño, etc.) con miras a la obtención de la arquitectura.
5. Etapa de documentación de la arquitectura: Se generaron los documentos de la descripción de la arquitectura, las iteraciones y una recopilación de las especificaciones de las diferentes partes del sistema.

2.4.3.3. Gestión de riesgos

Los riesgos son una constante en el desarrollo de cualquier proyecto. Durante esta iteración se asignó un tiempo especial para analizar los riesgos *in extenso* y documentarlos, después de lo cual, los riesgos serían tratados a medida que fueran apareciendo.

Primero se elaboró un plan de riesgos y en ese plan se definió la forma de documentarlos y de tratarlos, este plan se anexa.

Posteriormente se generaron varios documentos con la información de los riesgos:

1. Lista de riesgos: es una tabla sencilla con el nombre del riesgo, su causa, su efecto, su impacto y su probabilidad, así como su estado.
2. Análisis cualitativo: es un documento donde se ordena a los riesgos de acuerdo al producto Probabilidad x Impacto para determinar prioridades.
3. Documentación detallada de los riesgos: los riesgos más importantes de acuerdo con el análisis cualitativo se definieron e indagaron ampliamente para determinar su posible solución.

Los riesgos más importantes en cuanto a la arquitectura que fueron hallados al inicio de la segunda iteración fueron los siguientes:

1. Arquitectura frágil.
2. Alto acoplamiento de objetos, subsistemas, etc.
3. Redundancia.
4. Carencia de seguridad.
5. En el análisis no se gestiona la seguridad.
6. Análisis parcial y poco cohesionado.
7. Modelo de datos incompleto.
8. El modelo de datos no gestiona la seguridad.

9. Análisis y diseño fusionados.
10. Carencia de subsistemas e interfases.
11. Falta de paquetes en análisis.
12. Ambigüedad en los datos a transmitir a la interfaz general de usuario.
13. Ambigüedad en la forma de involucrar los casos de uso y los módulos en el sistema.
14. Formato de resultado de los módulos inexistente.
15. Dificultad para encontrar en qué momento cargar los módulos.
16. Modelo de casos de uso incompleto, vago e inconsistente.

Con esta información se elaboró el resto del plan de la segunda iteración.

2.4.3.4. Especificaciones

En esta iteración se hicieron las siguientes especificaciones previas al desarrollo del software:

1. Especificación de seguridad.
2. Especificación de las pruebas.

2.4.3.4.1. Seguridad

Según el análisis de riesgos el sistema carecía de una especificación de seguridad, y por tanto, no había garantía de que el sistema fuera seguro; por tanto se generó un documento para definir los requisitos de la seguridad. Este documento se anexa.

2.4.3.4.2. Pruebas

Se generó un documento que define los procedimientos de prueba, para garantizar el buen funcionamiento de cada una de las partes del sistema. Este documento se anexa.

2.4.3.5. Etapa de desarrollo del software

En esta etapa, con la información de los riesgos y la seguridad, se procedió a elaborar la arquitectura.

La elaboración de la arquitectura en su forma estable, pasó por los siguientes flujos de trabajo:

1. Requisitos.
2. Análisis.
3. Diseño.
4. Implementación.
5. Pruebas.

La forma que siguieron estos flujos de trabajo no fue exactamente lineal, porque, por ejemplo, cuando se tuvo bien planteado el diseño, se realizó una proyección del tiempo de desarrollo de todos los requisitos y como tomaban demasiado tiempo se hizo una negociación de requisitos para entregar el sistema en dos versiones, una mínima que es la que se presenta en este trabajo, y otra más amplia que se continuará después.

2.4.3.5.1. Requisitos

Lo primero que se hizo fue resolver una ambigüedad en el modelo de los actores, en comunicación con el cliente se suprimieron varios actores innecesarios y se reacomodaron

otros; el resultado fue la simplificación de la seguridad a dos niveles para la lectura: público y privado (consejeros), y tres para la escritura: público, consejeros y autorizadores.

El diagrama se muestra a continuación y es el que corresponde con la versión definitiva en la descripción de la arquitectura.

DIAGRAMA DEFINITIVO DE LOS ACTORES

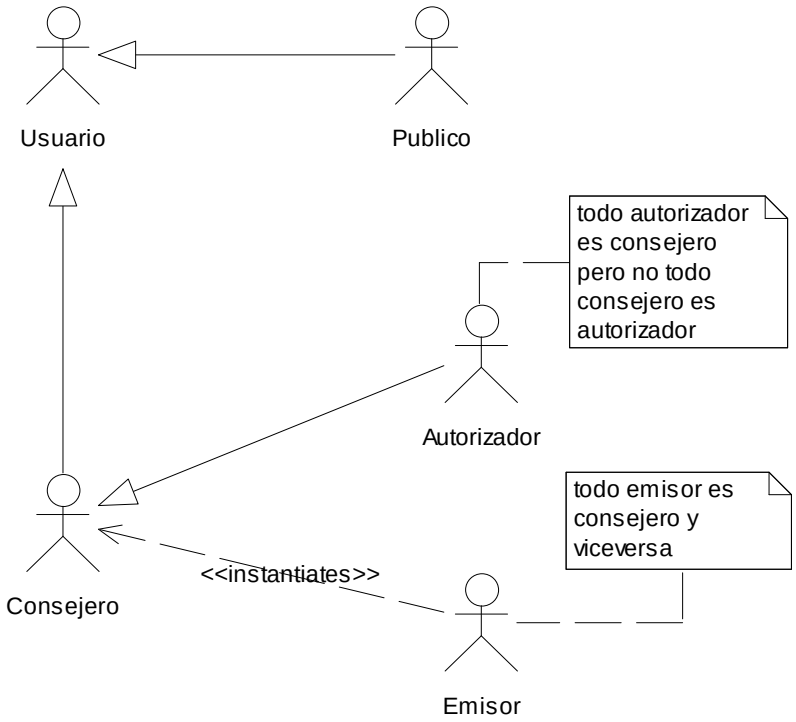

Fig. 2.4.3-1: Diagrama definitivo de los actores

Después se hizo una reasignación de un caso de uso: registrar (dar de alta) un usuario, en conversación con el cliente se llegó al acuerdo de que sólo los consejeros, y de éstos, sólo los *autorizadores* podrían dar de alta a otros usuarios del sistema. La razón de hacerlo así es porque, por una parte ya se tiene la información de los consejeros, y por otra se evita estar rechazando solicitudes innecesarias o con datos falsos. Esta es una modificación de seguridad.

Se realizó una exploración exhaustiva de los requisitos, para lo cual se realizó un sondeo entre varios consejeros de la CMDH para saber qué era lo que esperaban del sitio; con la información obtenida y con los datos que se tenían previamente, se construyó un modelo de casos de uso extendido, el cual contenía 25 casos de uso agrupados en 6 paquetes.

Después de esto se realizó un filtro para obtener los casos de uso significativos para la arquitectura que resultaron ser 8.

Posteriormente se procedió al análisis y al diseño no detallado.

Terminado el diseño no detallado, se hizo una proyección del tiempo de desarrollo del sistema completo, se generó un riesgo dada la cantidad de casos de uso a construir. Por tanto se hizo una negociación con el cliente para realizar el sistema en dos versiones, una mínima que es la que se presenta ahora y otra completa que se realizaría inmediatamente después, pero a manera de pequeñas entregas en donde cada entrega agrega una funcionalidad nueva al sistema ya en uso.

Por tanto se adoptó como definitivo el modelo de casos de uso significativos para la arquitectura, que es el que se presentó en la descripción de la arquitectura.

Aun así, el criterio fue: construir el sistema de tal forma que permita agregar nuevas opciones y casos de uso sin necesidad de rehacer el mismo.

2.4.3.5.2. Análisis

En este flujo de trabajo se realizaron dos cosas:

1. Se modificó la base de datos para que cumpliera mejor con los requisitos y la seguridad: el modelo es el presentado en la descripción de la arquitectura.
2. Se organizó el modelo de análisis en paquetes y dependencias entre ellos.

Después del diseño general y de indagar las características de los subsistemas, se descubrió que el patrón genérico resultaba poco consistente y que ofrecía varias dificultades, entre otras el alto acoplamiento en un solo modelo o diagrama de clases de áreas de formato, base de datos, ejecución de programa e integración del mismo.

Por tanto se puso en duda este patrón, dejándolo como borrador por si se llegara a reutilizar parte del mismo.

El modelo de análisis es el que se muestra en la descripción de la arquitectura.

En cuanto a la flexibilidad para admitir más requisitos no planteados, se modificó el paquete *navegar-análisis* para que fuera un sistema que cargara y ejecutara *módulos* independientes, los cuáles realizarían las peticiones del usuario (casos de uso).

Así las responsabilidades de formato, flujo del programa, etc. quedarían a cargo de las clases de navegación y no de los *módulos* de los casos de uso resolviendo el problema del alto acoplamiento.

2.4.3.5.3. Diseño

El diseño tuvo dos etapas, una general y una específica de cada subsistema.

En la etapa general se refinó el modelo planteado para la navegación en la iteración anterior, hasta llevarlo a su forma definitiva.

Posteriormente se definieron las interfases y el formato de los datos a transmitirse entre las mismas.

La especificación del formato de datos entre las interfases se anexa en su versión definitiva, y fueron los siguientes archivos:

1. modules-response-2.00.00.dtd: respuesta de los módulos al subsistema cargador de módulos (XML).
2. menu-2.00.01.dtd: formato de menu (de navegación, de usuario, formularios predefinidos, etc.) a enviar a la interfaz *FormatManagerIF* (XML).
3. content-2.00.00.dtd: formato para enviar el contenido de una página a la interfaz *FormatManagerIF* (XML).
4. module-arguments-1.00.00.txt: formato de los argumentos de los módulos (arreglos).
5. formato-datos-2.00.00.txt: archivo que explica el formato de los datos en general.

El motivo de que se escogiera el XML es la facilidad con que se definen estructuras complejas en esta tecnología; así como los adelantos que se han hecho al respecto.

La etapa específica de diseño va en paralelo con la implementación y fue por subsistema; se diseñó en detalle el subsistema y sus clases y se implementó y probó. Las correcciones se hicieron tanto al diseño como al código.

Hubo correcciones aplicadas principalmente al formato de los datos, lo cual hizo que se actualizaran las versiones de los documentos. Se anexan las versiones definitivas.

2.4.3.5.4. Implementación

La implementación se centró en los subsistemas planteados en el diseño, únicamente se construyó un módulo para pruebas de concepto (STUB).

Este flujo de trabajo se realizó casi en paralelo con el diseño detallado y se realizó subsistema por subsistema.

2.4.3.5.5. Pruebas

Las pruebas se realizaron de acuerdo a los procedimientos de prueba que aplicaban a este caso, a saber: las pruebas de unidad, las pruebas de integración y las pruebas de sistema.

2.4.3.5.6. Documentación

La documentación de esta iteración está dividida en cuatro partes, la primera es la descripción de la arquitectura al final de la segunda iteración; la segunda y tercera describen las iteraciones y los criterios tomados en ellas; finalmente, la cuarta parte es un compendio de documentos que especifican partes importantes de la arquitectura, como el formato de los datos, y partes importantes del desarrollo como los documentos de riesgos.

Algunos de los documentos serán utilizados en las posteriores fases de desarrollo, en particular, el plan de riesgos (con algunas modificaciones), la especificación de seguridad y los procedimientos de prueba.

2.4.4. Documentación de la arquitectura

2.4.4.1. Introducción

En este apartado colocamos una serie de documentos generados durante la fase de elaboración. Estos documentos se dividen en dos categorías: los documentos referentes al proceso y los documentos referentes al producto. Los documentos referentes al proceso son aquellos que se utilizaron durante el proceso de desarrollo. Los documentos referentes al producto forman parte de la arquitectura del sistema.

2.4.4.1.1. Documentos referentes al proceso

1. Plan de la segunda iteración: es el plan que se trazó al inicio de la segunda iteración de la fase de elaboración (la primera iteración careció de plan escrito).
2. Plan de riesgos: es el plan trazado al inicio para poder gestionar los riesgos.
3. Lista de riesgos (se muestra la versión final): es la lista de riesgos y su estado de solución al final de la fase de elaboración.
4. Análisis cualitativo de riesgos: al inicio
5. Análisis cualitativo de riesgos: al final
6. Descripción detallada de los riesgos.
7. Procedimientos de prueba.

2.4.4.1.2. Documentos referentes al producto

1. Especificación de seguridad.
2. Formato de datos: especifica el formato de los datos entre las interfases de los subsistemas.
3. *modules-response-x.xx.xx.dtd*: respuesta de los módulos desde la interfaz *PluginManagerIF*.
4. *content-x.xx.xx.dtd*: formato de contenido para la interfaz *FormatManagerIF*.
5. *menu-x.xx.xx.dtd*: formato de menú para la interfaz *FormatManagerIF*.
6. *moule-arguments-x.xx.xx.txt*: argumentos para los múdulos desde la interfaz *PluginManagerIF*.

2.4.4.2. Plan de la segunda iteración

2.4.4.2.1. Contenido

- Contenido
- Objetivos
- Actividades

2.4.4.2.2. Objetivos

1. obtener la descripción de la arquitectura
2. probar la línea base de la arquitectura
3. documentar la fase de elaboración

2.4.4.2.3. Actividades

1. establecer el plan de la iteración (este documento)
2. establecer el plan de riesgos
3. gestionar los riesgos (i.e. modificar este plan)
4. obtener los requisitos: casos de uso y requisitos especiales
5. priorizar los casos de uso, paquetes de casos de uso, etc.
6. obtener la vista de arquitectura (VA) del modelo de análisis: paquetes
7. priorizar los paquetes
8. obtener el DER: datos persistentes

(Por cada paquete de análisis y por cada caso de uso)

9. obtener la VA del modelo de diseño: subsistemas, y seguridad
10. obtener el diseño de BD: datos persistentes
11. obtener la VA del modelo de despliegue: general
12. obtener la VA del modelo de implementación
13. obtener la VA del modelo de pruebas: procedimientos
14. probar la línea base de la arquitectura: subsistema por subsistema
15. documentar los resultados
16. gestionar riesgos
17. volver a 3 para cada caso de uso, paquete, subsistema

2.4.4.3. Plan de riesgos de la segunda iteración

2.4.4.3.1. Contenido

- Revisiones
- Contenido
- Objetivos
- Restricciones
- Lista de documentos en los riesgos
- Categorías de los riesgos
- Grado de precisión del conocimiento de un riesgo
- Parámetros a medir en un riesgo
- Graduación de la probabilidad
- Graduación del impacto
- Método de obtención de los riesgos
- Políticas de detalle de los riesgos
- Información en la tabla de la lista de los riesgos
- Información en la tabla del análisis cualitativo
- Estados de solución de los riesgos
- Políticas de gestión de riesgos
- Análisis de causas de un riesgo
- Actualización de los riesgos
- Programa

2.4.4.3.2. Objetivos

1. Identificar los riesgos más importantes.
2. Identificar las consecuencias de los riesgos.
3. Identificar las causas de los riesgos.
4. Responder adecuadamente ante los riesgos.
5. Responder a tiempo a los riesgos.
6. Ajustar el plan del proyecto atendiendo a los riesgos.
7. Tener actualizados los riesgos.
8. Mejorar el plan de riesgos.

2.4.4.3.3. Restricciones

1. Equipo pequeño de desarrollo
2. El tiempo es muy corto
3. Se carece de estadísticas previas.
4. Por tanto se omite el análisis cuantitativo.
5. Igualmente los procesos de riesgos se limitaran a los objetivos y serán lo más simple posible.

2.4.4.3.4. Lista de documentos en los riesgos

1. El plan de riesgos (este documento)
2. La tabla de la lista de los riesgos
3. La descripción de cada riesgo *in extenso*

4. El análisis de las causas de un riesgo (puede estar incluido en la descripción de un riesgo).
5. La tabla del análisis cualitativo
6. Los planes de gestión de los riesgos (pueden estar en la descripción o en el plan general de la iteración).

2.4.4.3.5. Categorías de los riesgos

- PROYECTO, PROGRAMA, AGENDA, ETC.: riesgos relacionados con retrasos/adelantos en el programa y el desarrollo del proyecto (GRUPO I).
- SEGURIDAD, INTEGRIDAD, FUNCIONAMIENTO, CONSISTENCIA: riesgos que están relacionados con estas características en el sistema a desarrollar (GRUPO II).
- ASPECTO, INTERFAZ DE USUARIO: riesgos relacionados con estas áreas en el sistema (GRUPO III).
- REQUISITOS, AMBITO DEL SISTEMA, CALIDAD: riesgos del sistema relacionados con estas áreas (GRUPO IV).
- DOCUMENTACION: riesgos relacionados con la documentación (GRUPO V).

Nota: Debe reducirse al mínimo los riesgos que tengan mas de un grupo en cuyo caso elíjase un grupo principal y en la documentación in extenso del riesgo se anexan los grupos de riesgos relacionados.

2.4.4.3.6. Grado de precisión del conocimiento de un riesgo

Un riesgo puede ser muy preciso o muy ambiguo en su noción; por ejemplo: el retraso es muy ambiguo ya que puede haber un retraso muy grande o muy pequeño, en cambio, que cierta clase tenga un acoplamiento muy alto con elementos fuera de su subsistema o paquete es muy preciso, ya que un cambio en los requisitos implica que deje de funcionar esta clase.

Por tanto la noción de riesgo se clasificará en una escala del 1 al 5, donde 1 es lo más vago y 5 es lo más preciso.

2.4.4.3.7. Parámetros a medir en un riesgo

Parámetros ordinarios (todos los riesgos los llevan):

- nombre del riesgo.
- clave única del riesgo.
- precisión del riesgo.
- categoría del riesgo.
- impacto/prioridad del riesgo.
- probabilidad del riesgo.
- posible causa del riesgo (escueta).
- posible consecuencia del riesgo (escueta).
- estado del riesgo.
- esta detallado el riesgo (si/no).
- indicadores del riesgo.
- Fecha de descubrimiento del riesgo.

Parámetros de detalle (documento separado):

- descripción del riesgo.
- causas del riesgo.
- plan de solución del riesgo.
- contingencia.
- documentos relacionados con el riesgo.
- historial del riesgo.

2.4.4.3.8. Graduación de la probabilidad

- Se refiere a la probabilidad de que el riesgo se haga realidad.
- Es una apreciación subjetiva ya que no existen elementos estadísticos ni datos de proyectos anteriores.
- Se tiene una escala del 0.1 al 0.9 de probabilidad y una resolución de 0.1.
- En casos excepcionales se acepta 0.0 o 1.0 pero se debe documentar el motivo.

2.4.4.3.9. Graduación del impacto

- Se refiere al área del proyecto/sistema que llega a afectar y al grado en que lo hace.
- Es una apreciación subjetiva ya que no existen elementos estadísticos ni datos de proyectos anteriores.
- Se tiene una escala del 0.1 al 1.0 con una resolución de 0.1 que indica 10%.
- En casos relevantes se documentara a que se refiere el impacto, si al alcance del riesgo en el proyecto o a la intensidad con que afecta al mismo.

2.4.4.3.10. Método de obtención de los riesgos

1. Revisar la lista de riesgos anterior.
2. Revisar la línea base anterior.
3. Revisar el plan actual.
4. Anotar los riesgos que surjan en el desarrollo del proyecto.
5. Revisar los riesgos que surgen de las soluciones a otros riesgos.
6. Riesgos derivados de las pruebas.

2.4.4.3.11. Políticas de detalle de los riesgos

Los dos criterios para detallar los riesgos son:

1. cuando el riesgo es muy ambiguo.
2. cuando el riesgo tiene una escala muy alta en el análisis cualitativo (probabilidad x impacto) >= 0.5

Lo que se documenta en el detalle del riesgo (documento aparte):

- Incluir los parámetros ordinarios.
- Los parámetros de detalle.
  (Vea apartado *2.4.4.3.7* de este documento)

2.4.4.3.12. Información en la tabla de la lista de los riesgos

- Los parámetros indicados en el apartado VI de este documento que están etiquetados como "parámetros ordinarios".
- La forma de hacerlo es agrupando los riesgos por categoría.

2.4.4.3.13. Información en la tabla del análisis cualitativo

Se forma con dos matrices, una con los números y otra con las claves que coinciden con los números.

- Las columnas indican la probabilidad.
- Los renglones indican el impacto.
- Las celdas son el producto:
  - En la primera matriz se coloca el valor numérico.
  - En la segunda matriz las claves de los riesgos cuyo valor PxI coincida con ese valor numérico.

2.4.4.3.14. Estados de solución de los riesgos

- Pendiente: todavía no se gestiona el riesgo.
- Resuelto: el riesgo desaparece ya sea en su impacto o en su probabilidad.
- Transferido: se transfiere el riesgo a otra etapa.
- Reducido: se reduce el impacto o la probabilidad del riesgo.
- Sin resolver/aceptado: no se atiende al riesgo.
- Materializado: el riesgo se vuelve realidad.

En el caso de que el riesgo no es resuelva existen dos posibilidades:

- que exista un plan de contingencia.
- que se resista el riesgo pasivamente.

2.4.4.3.15. Políticas de gestión de riesgos

- Cuando el valor PxI es mayor o igual al 0.5 se detalla, se buscan sus causas y se busca solucionarlo de inmediato.
- Cuando el valor PxI esta entre el 0.2 y el 0.5 (sin ser 0.5) se busca una solución poco detallada o se transfiere.
- Cuando el valor PxI es menor a 0.2 se ignora el riesgo.

2.4.4.3.16. Análisis de causas de un riesgo

1. Se anotan las causas directas de un riesgo, si son muy ambiguas se explican.
2. Para cada causa se anotan sus posibles causas siguiendo el mismo procedimiento.
3. Se profundiza hasta encontrar alguna tentativa de solución o se tenga un grado de precisión aceptable.
4. Se intenta buscar similitudes de las causas.
5. Pueden usarse diagramas tipo Ishikawa o bien diagramas de precedencia.

2.4.4.3.17. Actualización de los riesgos

- Proceso ordinario: al finalizar cada artefacto, objeto, etc.
- Proceso extraordinario: cuando se detecte un riesgo.

2.4.4.3.18. Programa

Programa inicial:

1. Examinar los riesgos de la fase de inicio.
2. Eliminar los riesgos no existentes.

-- por categoría de riesgo: ---

3. Examinar los artefactos de la fase de elaboración (hasta ahora).
4. Documentar los riesgos encontrados.
5. Realizar la lluvia de ideas de los riesgos nuevos.
6. Depurar y detallar (donde aplique) la lista de riesgos.
7. Asignar el impacto y la probabilidad.
8. Realizar el análisis cualitativo.
9. Detallar los riesgos importantes (PxI >= 0.5)
10. Indagar las causas de los riesgos importantes.
11. Establecer el estado de los riesgos.
12. Realizar el plan de los riesgos.

Programa de continuidad:

1. Registrar el riesgo con sus parámetros ordinarios.
2. Si lo amerita detallar el riesgo.
3. Realizar el análisis cualitativo.
4. Si lo amerita indagar sus causas.
5. Establecer su estado.
6. Si lo amerita realizar su plan.

Programa del plan de riesgos:

1. revisar los riesgos iniciales
2. revisar los riesgos generados
3. revisar la precisión del riesgo
4. revisar los riesgos solucionados
5. revisar los riesgos transferidos
6. revisar los riesgos reducidos
7. revisar los riesgos ignorados
8. revisar los riesgos materializados
9. revisar el plan de riesgos
10. revisar el plan de solución de riesgos.

2.4.4.4. Lista de riesgos

La lista de riesgos que es el resultado del *plan de riesgos* se muestra a continuación.

TABLA 2.4.4-1: Lista de riesgos encontrados en la fase de elaboración

GRUPO I: proyecto, programa, agenda. Ponderación: 1.0									
clave	**Nombre**	**precisión**	**Impacto**	**probabilidad**	**causa**	**consecuencia**	**estado**	**fecha**	**Detallado**
1	Agenda apretada	5	1.0	0.8	Desconocimiento de los artefactos y actividades a realizar	Retraso, tensión, improvisación, mala calidad.	Pendiente	20040721	Sí
31	no se da seguimiento a los riesgos	5	0.9	0.9	agenda muy apretada para un solo día, falta de estrategia de seguimiento	el análisis de riesgos se vuelve inoperante	Pendiente	40040806	Sí
2	Retraso	1	1.0	0.9	Arquitectura frágil, varias.	Colapso del proyecto	Pendiente	20040721	Sí
3	Poca información del proceso	3	0.8	0.6	Falta tiempo para documentarse. Falta experiencia.	Planes erróneos, retraso.	Reducido	20040721	No

GRUPO II: seguridad, integridad, funcionamiento, consistencia. Ponderación: 0.8									
clave	**nombre**	**precisión**	**Impacto**	**probabilidad**	**Causa**	**consecuencia**	**Estado**	**fecha**	**Detallado**
4	Carencia de arquitectura	5	1.0	0.4	No haberla diseñado	Colapso del sistema, no resistir a los cambios, retraso.	Reducido.	20040721	Sí
5	Arquitectura frágil	4	0.8	0.8	Análisis incompleto	Poca resistencia a los cambios, retrasos, colapsos	Pendiente	20040721	Sí
6	Alto acoplamiento de objetos, subsistemas, etc.	5	0.8	0.8	Análisis incompleto, arquitectura frágil.	Poca resistencia a los cambios, retrasos.	Pendiente	20040721	Sí
7	Redundancia	3	0.8	0.7	Arquitectura frágil.	Ineficiencia del sistema.	Pendiente	20040721	No
8	Carencia de seguridad	4	0.9	0.9	No haberla diseñado	Incumplimiento de objetivos.	Pendiente	20040721	Sí
17	Malas interfaces	5	1.0	0.7	Improvisación	Mala arquitectura.	Pendiente	20040721	Sí
18	Malos subsistemas	5	1.0	0.7	Improvisación	Mala arquitectura	Pendiente	20040721	Sí
20	En el análisis no se gestiona la seguridad	4	0.8	0.8	No contemplar todos los aspectos en el análisis	No realizar la seguridad	Pendiente	20040721	Sí
21	Análisis parcial y poco cohesionado	5	0.8	0.8	Falta de revisión entre los distintos modelos de análisis	Redundancia, difícil integración, problemas en el diseño	Pendiente	20040721	Sí

clave	Nombre	precisión	Impacto	probabilidad	Causa	consecuencia	Estado	fecha	Detallado
22	Modelo de datos incompleto	5	0.9	0.9	Requisitos vagos, análisis parcial	Problemas en el diseño.	Pendiente	20040721	Sí
23	El modelo de datos no gestiona la seguridad	5	0.9	0.9	Modelo de datos incompleto	Problemas de seguridad	Pendiente	20040721	Sí
24	Análisis y diseño fusionados	4	0.8	0.8	Falta de subsistemas e interfases	Alto acoplamiento en el diseño	Pendiente	20040721	Sí
25	Falta de subsistemas e interfases	4	0.9	0.9	No tener una noción clara del sistema en el análisis	Arquitectura débil, retrasos, redundancia	Pendiente	20040721	Sí
26	Falta de paquetes en análisis	5	0.9	0.9	No tener claros los casos de uso	Falta de subsistema e interfases	Pendiente	20040721	Sí
27	Ambigüedad en los datos a transmitir a la interfaz general de usuario.	5	0.8	0.9	No se ha precisado con claridad	No se puede saber qué información y en qué formato ha de transmitirse	Pendiente	20040721	Sí
28	Ambigüedad en la forma de involucrar los casos de uso y los módulos en el sistema	5	0.9	0.9	Falta detallar la clase encargada y los procedimientos para cargar módulos	Inconsistencia en la integración de los módulos, retrasos, arquitectura débil	Pendiente	20040721	Sí
29	Formato de resultado de los módulos inexistente	5	0.9	0.9	No se ha especificado el procedimiento de tratar los módulos	No se sabe qué datos van a entregar los módulos ni en qué formato	Pendiente	20040721	Sí
30	Dificultad para encontrar en qué momento cargar los módulos	5	0.8	0.0	Restricciones del lenguaje de programación	Errores en el código, rediseño del cargador de módulos, etc.	Resuelto	20040721	Sí

GRUPO III: aspecto, interfaz de usuario. Ponderación: 0.6									
clave	nombre	precisión	Impacto	probabilidad	Causa	Consecuencia	estado	Fecha	Detallado
9	Interfaz de usuario separada de programación	5	0.6	0.9	No se ha integrado	Retraso, incumplimiento de objetivo de imagen del cliente	Pendiente	20040721	No
10	Interfaz de usuario muy acoplada	5	0.6	0.6	Arquitectura débil, malas interfaces	Poca resistencia a los cambios, retrasos, redundancias	Pendiente	20040721	No

clave	nombre	precisión	Impacto	probabilidad	Causa	Consecuencia	estado	Fecha	Detallado
11	No apegarse a la accesibilidad	5	0.6	0.6	Improvisación, poco tiempo, arquitectura débil.	Mala calidad en el sitio, lentitud al mostrarse, usuarios insatisfechos.	Pendiente	20040721	No

GRUPO IV: requisitos, ámbito del sistema, calidad. Ponderación: 1.0 para requisitos, 0.6 para calidad.									
clave	**nombre**	**precisión**	**Impacto**	**probabilidad**	**causa**	**Consecuencia**	**estado**	**fecha**	**Detallado**
12	Requisitos funcionales incompletos	5	1.0	0.5	Falta de comprensión al cliente	Hacer un sistema que no haga lo que el cliente pida	Reducido	20040721	Sí
13	Requisitos no funcionales incompletos	5	1.0	0.5	Falta de comprensión del sistema	Hacer un sistema deficiente en su rendimiento.	Reducido	20040721	Sí
14	Falta de estándares para los artefactos	5	1.0	0.4	Poca experiencia en el desarrollo del software.	Hacer artefactos incongruentes y difíciles de entender	Reducido	20040721	No
19	Modelo de casos de uso incompleto, vago e inconsistente	5	1.0	0.7	Falta precisión	No comprender los requisitos funcionales	Pendiente	20040721	Sí

GRUPO V: documentación. Ponderación: 0.4									
clave	**nombre**	**precisión**	**Impacto**	**probabilidad**	**causa**	**Consecuencia**	**estado**	**fecha**	**Detallado**
15	Falta documentación	5	0.4	0.8	No se ha realizado	Que no se pueda usar/comprender el sistema.	Pendiente	20040721	No
16	Carencia de estándares de documentación	5	0.4	0.9	No se ha realizado	Baja calidad en la documentación.	Pendiente	20040721	No

2.4.4.5. Análisis cualitativo al inicio

TABLA 2.4.4-2: Matriz de decisión en el análisis cualitativo (al inicio)

Impacto \ Probabilidad	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
1.00	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.90	0.00	0.09	0.18	0.27	0.36	0.45	0.54	0.63	0.72	0.81	0.90
0.80	0.00	0.08	0.16	0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80
0.70	0.00	0.07	0.14	0.21	0.28	0.35	0.42	0.49	0.56	0.63	0.70
0.60	0.00	0.06	0.12	0.18	0.24	0.30	0.36	0.42	0.48	0.54	0.60
0.50	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
0.40	0.00	0.04	0.08	0.12	0.16	0.20	0.24	0.28	0.32	0.36	0.40
0.30	0.00	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
0.20	0.00	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
0.10	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLA 2.4.4-3: Riesgos acomodados según Impacto x Probabilidad x Ponderación (al inicio)

clave	Riesgo	IxPxP	ponderación	Prob.	impacto	grupo
2	**Retraso**	**0.9**	**1**	**0.9**	**1**	**GRUPO I**
1	**Agenda apretada**	**0.8**	**1**	**0.8**	**1**	**GRUPO I**
19	**Modelo de casos de uso incompleto, vago e inconsistente**	**0.7**	**1**	**0.7**	**1**	**GRUPO IV**
8	**Carencia de seguridad**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
22	**Modelo de datos incompleto**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
23	**El modelo de datos no gestiona la seguridad**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
25	**Falta de subsistemas e interfases**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
26	**Falta de paquetes en análisis**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
28	**Ambigüedad en la forma de involucrar los casos de uso y los módulos en el sistema**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
29	**Formato de resultado de los módulos inexistente**	**0.648**	**0.8**	**0.9**	**0.9**	**GRUPO II**
27	**Ambigüedad en los datos a transmitir a la interfaz general de usuario.**	**0.576**	**0.8**	**0.9**	**0.8**	**GRUPO II**
30	**Dificultad para encontrar en qué momento cargar los módulos**	**0.576**	**0.8**	**0.9**	**0.8**	**GRUPO II**
17	**Malas interfaces**	**0.56**	**0.8**	**0.7**	**1**	**GRUPO II**
18	**Malos subsistemas**	**0.56**	**0.8**	**0.7**	**1**	**GRUPO II**
5	**Arquitectura frágil**	**0.512**	**0.8**	**0.8**	**0.8**	**GRUPO II**
6	**Alto acoplamiento de objetos, subsistemas, etc.**	**0.512**	**0.8**	**0.8**	**0.8**	**GRUPO II**
20	**En el análisis no se gestiona la seguridad**	**0.512**	**0.8**	**0.8**	**0.8**	**GRUPO II**
21	**Análisis parcial y poco cohesionado**	**0.512**	**0.8**	**0.8**	**0.8**	**GRUPO II**
24	**Análisis y diseño fusionados**	**0.512**	**0.8**	**0.8**	**0.8**	**GRUPO II**
12	**Requisitos funcionales incompletos**	**0.5**	**1**	**0.5**	**1**	**GRUPO IV**
13	**Requisitos no funcionales incompletos**	**0.5**	**1**	**0.5**	**1**	**GRUPO IV**
3	Poca información del proceso	0.48	1	0.6	0.8	GRUPO I
7	Redundancia	0.448	0.8	0.7	0.8	GRUPO II

clave	Riesgo	IxPxP	ponderación	Prob.	impacto	grupo
9	Interfaz de usuario separada de programación	0.324	0.6	0.9	0.6	GRUPO III
4	Carencia de arquitectura	0.32	0.8	0.4	1	GRUPO II
14	Falta de estándares para los artefactos	0.24	0.6	0.4	1	GRUPO IV
10	Interfaz de usuario muy acoplada	0.216	0.6	0.6	0.6	GRUPO III
11	No apegarse a la accesibilidad	0.216	0.6	0.6	0.6	GRUPO III
16	Carencia de estándares de documentación	0.144	0.4	0.9	0.4	GRUPO V
15	Falta documentación	0.128	0.4	0.8	0.4	GRUPO V

2.4.4.6. Análisis cualitativo al final

TABLA 2.4.4-4: Matriz de decisión en el análisis cualitativo (al final)

		probabilidad										
		0.00	**0.10**	**0.20**	**0.30**	**0.40**	**0.50**	**0.60**	**0.70**	**0.80**	**0.90**	**1.00**
Impacto	**1.00**	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
	0.90	0.00	0.09	0.18	0.27	0.36	0.45	0.54	0.63	0.72	0.81	0.90
	0.80	0.00	0.08	0.16	0.24	0.32	0.40	0.48	0.56	0.64	0.72	0.80
	0.70	0.00	0.07	0.14	0.21	0.28	0.35	0.42	0.49	0.56	0.63	0.70
	0.60	0.00	0.06	0.12	0.18	0.24	0.30	0.36	0.42	0.48	0.54	0.60
	0.50	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
	0.40	0.00	0.04	0.08	0.12	0.16	0.20	0.24	0.28	0.32	0.36	0.40
	0.30	0.00	0.03	0.06	0.09	0.12	0.15	0.18	0.21	0.24	0.27	0.30
	0.20	0.00	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
	0.10	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TABLA 2.4.4-5: Riesgos acomodados según Impacto x Probabilidad x Ponderación (al final)

cve	**riesgo**	**IP**	**pond**	**Prob**	**impacto**	**prec**	**detail**	**grupo**
2	**Retraso**	**0.9**	**1**	**0.9**	**1**	**1**	**Sí**	**GRUPO I**
1	**Agenda apretada**	**0.8**	**1**	**0.8**	**1**	**5**	**Sí**	**GRUPO I**
22	**Modelo de datos incompleto**	**0.648**	**0.8**	**0.9**	**0.9**	**5**	**Sí**	**GRUPO II**
23	**El modelo de datos no gestiona la seguridad**	**0.648**	**0.8**	**0.9**	**0.9**	**5**	**Sí**	**GRUPO II**
27	**Ambigüedad en los datos a transmitir a la interfaz general de usuario.**	**0.576**	**0.8**	**0.9**	**0.8**	**5**	**Sí**	**GRUPO II**
13	**Requisitos no funcionales incompletos**	**0.5**	**1**	**0.5**	**1**	**5**	**No**	**GRUPO IV**
3	Poca información del proceso	0.48	1	0.6	0.8	3	No	GRUPO I
31	no se da seguimiento a los riesgos	0.45	1	0.5	0.9	5	Sí	GRUPO I
7	Redundancia	0.448	0.8	0.7	0.8	3	No	GRUPO II
21	Análisis parcial y poco cohesionado	0.384	0.8	0.6	0.8	5	Sí	GRUPO II
26	Falta de paquetes en análisis	0.36	0.8	0.5	0.9	5	Sí	GRUPO II
9	Interfaz de usuario separada de programación	0.324	0.6	0.9	0.6	5	No	GRUPO III
4	Carencia de arquitectura	0.32	0.8	0.4	1	5	No	GRUPO II

cve	riesgo	IP	pond	Prob	impacto	prec	detail	grupo
5	Arquitectura frágil	0.32	0.8	0.5	0.8	4	Sí	GRUPO II
6	Alto acoplamiento de objetos, subsistemas, etc.	0.256	0.8	0.4	0.8	5	Sí	GRUPO II
24	Análisis y diseño fusionados	0.256	0.8	0.4	0.8	4	Sí	GRUPO II
14	Falta de estándares para los artefactos	0.24	0.6	0.4	1	5	No	GRUPO IV
11	No apegarse a la accesibilidad	0.216	0.6	0.6	0.6	5	No	GRUPO III
10	Interfaz de usuario muy acoplada	0.216	0.6	0.6	0.6	5	No	GRUPO III
12	Requisitos funcionales incompletos	0.2	1	0.2	1	5	No	GRUPO IV
19	Modelo de casos de uso incompleto, vago e inconsistente	0.2	1	0.2	1	5	Sí	GRUPO IV
17	Malas interfaces	0.16	0.8	0.2	1	5	Sí	GRUPO II
18	Malos subsistemas	0.16	0.8	0.2	1	5	Sí	GRUPO II
8	Carencia de seguridad	0.144	0.8	0.2	0.9	4	Sí	GRUPO II
16	Carencia de estándares de documentación	0.144	0.4	0.9	0.4	5	No	GRUPO V
25	Falta de subsistemas e interfases	0.144	0.8	0.2	0.9	4	Sí	GRUPO II
28	Ambigüedad en la forma de involucrar los casos de uso y los módulos en el sistema	0.144	0.8	0.2	0.9	5	Sí	GRUPO II
20	En el análisis no se gestiona la seguridad	0.128	0.8	0.2	0.8	4	Sí	GRUPO II
15	Falta documentación	0.128	0.4	0.8	0.4	5	No	GRUPO V
30	Dificultad para encontrar en qué momento cargar los módulos	0	0.8	0	0.8	5	Sí	GRUPO II
29	Formato de resultado de los módulos inexistente	0	0.8	0	0.9	5	Sí	GRUPO II

2.4.4.7. Documentación detallada de los riesgos importantes y ambiguos

2.4.4.7.1. CATEGORÍA: GRUPO I: proyecto, programa, agenda

Clave: 2
Nombre: Retraso.
Estado: Pendiente.
Precisión: 1 Impacto: 1.0 Probabilidad: 0.9 Ponderación: 1.0 Producto: 0.9
Fecha de registro del riesgo: 2004-07-21
Descripción:

El retraso es muy ambiguo. Existen retrasos cortos y retrasos largos. Por otra parte el retraso es acumulativo, pues retrasarse en una cosa lleva a retrasarse en otra. Además existe el problema de que por retraso se hagan mal las cosas y en última instancia que al cliente no le sea de utilidad el producto.
Aunque el riesgo no se resuelva o se mitigue se pasará a la siguiente fase/iteración y se le dará seguimiento a través de todas las fases.

Causas:

- Las causas del retraso son múltiples. Están entre otras, una mala planeación, perder de vista los objetivos, una mala arquitectura, modelos incongruentes, una gestión del riesgo inadecuada, elegir una solución demasiado compleja. También está la falta de componentes para determinada solución. Igualmente, el hecho de que el proceso de desarrollo implica muchas actividades desconocidas en el momento de planear. Finalmente la falta de experiencia.
- Entre las causas de la mala planeación está el fijar una agenda muy apretada, el escoger criterios demasiado rígidos, tener demasiadas actividades o que sean muy complejas. Tener en agenda cosas que no sean tan importantes para el resultado, etc.
- Las causas de perder de vista los objetivos son: una agenda demasiado complicada y la distracción en cosas menos importantes, así como no gestionar los riesgos adecuadamente.
- Las causas de la mala arquitectura van desde la no comprensión de los requisitos hasta las malas interfases y subsistemas.
- Los modelos incongruentes se deben a una mala arquitectura o a no considerar todos los aspectos de la arquitectura y de los requisitos.
- La gestión de riesgos inadecuada se debe a desatender los riesgos o atender en demasía riesgos poco importantes.
- Las causas de elegir una solución demasiado compleja son: la falta de experiencia, perder de vista los objetivos, perder de vista la arquitectura, una mala arquitectura y una mala gestión del riesgo.
- La falta de componentes proviene de la falta de experiencia, una variante es, no saberlos elegir correctamente y la otra es construirlos demasiado complicados.

Acciones a considerar:

- Realizar una gestión completa de los requisitos.
- Revisar el plan y simplificar los procesos.
- Establecer los criterios de suficiencia en la documentación y artefactos.
- Mantener al día los riesgos y sus políticas.
- Gestionar una adecuada arquitectura en sus requisitos, paquetes, subsistemas e interfaces.
- Concentrarse en la arquitectura dejando de lado lo que no es de la arquitectura.
- Configurar lo más flexiblemente posible la arquitectura.
- Tomar en cuenta la arquitectura y los requisitos para la congruencia de los modelos.
- Atender únicamente los riesgos más importantes, documentarlos únicamente lo indispensable.
- Concentrarse en los objetivos de la fase.
- Elegir componentes ya hechos, construir los componentes de la manera más simple.
- Hacer referencia a otros riesgos y sus soluciones para escribir menos en la documentación de los riesgos.

Contingencia:

- Hacer una reducción de los requisitos negociando con el cliente.

Cambios:

2004-07-21: se detalló el riesgo.

2004-08-09: se completó la documentación del riesgo.

Clave: 31
Nombre: no se da seguimiento a los riesgos
Estado: Mitigado.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.5 Ponderación: 1.0 Producto: 0.45
Fecha de registro del riesgo: 2004-08-06
Descripción:

Los riesgos quedaron planificados, identificados, analizados, y con propuestas de solución y seguimiento cuando lo marcaba el plan. Una vez que la agenda no se ocupó de los riesgos y debido a los retrasos el seguimiento a los riesgos cayó en desuso. Esto provocó y provoca que varios riesgos que se deseaban evitar no se lograran evitar y que otros riesgos recientemente identificados no sean agregados a la lista ni gestionados.

Causas:

- Agenda muy apretada para un solo día.
- El desarrollo del sistema y la revisión del proceso de desarrollo siguen una misma línea de ejecución, es decir, que no se realiza una actividad de la agenda hasta que se realicen las anteriores aunque haya retrasos.

Acciones a considerar:

- Llevar dos hilos de ejecución independientes, uno para la agenda del proyecto y otro para la gestión de los riesgos.
- El hilo de la agenda del proyecto sigue su propio camino de acuerdo con el plan trazado y con los retrasos que vaya sufriendo.
- El hilo del seguimiento de los riesgos se mantiene por revisiones a intervalos regulares de tiempo sin importar los retrasos de la agenda.

Contingencia:

- Revisar los riesgos al inicio y al final de cada iteración.

Cambios:

2004-08-06: se detalló el riesgo.
2004-08-09: Cambio de estado: pendiente→mitigado
Cambio de probabilidad: 0.9→0.5 (producto: 0.8→0.45)
Mejoras en la redacción.

Clave: 1
Nombre: Agenda Apretada.
Estado: Pendiente.
Precisión: 5 Impacto: 1.0 Probabilidad: 0.8 Ponderación: 1.0 Producto: 0.8
Fecha de registro del riesgo: 2004-07-22
Descripción:

El tiempo programado para hacer las actividades es menor del necesario.

Causas:

- Falta de experiencia.
- Falta de información.
- Esquemas rígidos en la planeación.

Acciones a considerar:

- Revisión de los planes.
- Gestión de los riesgos.
- Evitar la improvisación al planear.
- Buscar asesoría.

Contingencia:

- Hacer una reducción de los requisitos negociando con el cliente.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: se añadió: buscar asesoría en las acciones a considerar.

2.4.4.7.2. CATEGORÍA: GRUPO II: seguridad, integridad, funcionamiento, consistencia.

Clave: 8
Nombre: Carencia de seguridad.
Estado: Resuelto.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.0 Ponderación: 0.8 Producto: 0.0
Fecha de registro del riesgo: 2004-07-22
Descripción:

Hasta ahora existe un motor de sesiones que funciona correctamente, pero que está aislado del resto del sistema. No se han formalizado las políticas ni los servicios de seguridad, así como tampoco se ha establecido ningún modelo de seguridad integral para el sitio.

Causas:

- No se han contemplado todos los requisitos.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Hacer una adecuada gestión de requisitos.
- Continuar con el proceso de desarrollo poniendo énfasis en la especificación formal de la seguridad.

Contingencia:

- Buscar asesoría.
- Negociar con el cliente los requisitos de seguridad.

Cambios:

2004-07-22: se detalló el riesgo.
2004-07-25: Se creó el documento:
ESPECIFICACIÓN-DE-SEGURIDAD-1.00.00.doc
Con las políticas y los modelos de seguridad que se tomarán en cuenta en el sistema.
Estado: pendiente→resuelto.
Probabilidad: 0.9→0.0 (resuelto en el sentido de que hay modelo, no de que la seguridad es infalible)

Clave: 22
Nombre: Modelo de datos incompleto.
Estado: Pendiente.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.9 Ponderación: 0.8 Producto: 0.65
Fecha de registro del riesgo: 2004-07-22
Descripción:

La capa de datos persistentes que en su mayor parte se encontrará en la base de datos presenta muchas ambigüedades y está muy incompleta, de forma tal que no da soporte a todas las necesidades del sistema.

Causas:

- No se han contemplado todos los requisitos.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Hacer una adecuada gestión de requisitos.
- Continuar con el proceso de desarrollo poniendo énfasis en el análisis de los datos.
- Hacer un diagrama entidad relación lo más completo posible pero admitiendo la posibilidad de cambios de última hora.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

Clave: 32
Nombre: el protocolo de comunicación entre los módulos y el sistema de navegación es muy difícil/lento de implementar para la utilidad que produce.
Estado: Pendiente.
Precisión: 4 Impacto: 0.9 Probabilidad: 0.9 Ponderación: 0.8 Producto: 0.65
Fecha de registro del riesgo: 2004-08-11
Descripción:

Se diseño un protocolo y un estándar XML de comunicación entre los módulos y el sistema de navegación. Este protocolo tiene por finalidad evitar que los módulos se metan en problemas de formato y de direcciones Web que si cambian, tendrían que cambiar también en cada módulo, lo cual es alto acoplamiento. La solución planteada es un pseudo-lenguaje que separa los datos del formato, en una primera versión se intentó realizar una completa abstracción, cosa que no se logró; la segunda versión simplificada separaba parcialmente los datos del formato y resolvía completamente el problema de las direcciones Web. **El riesgo está en que implementar esta norma tanto en los módulos como en el sistema de navegación resulta en un retraso que no es aceptable para el desarrollo de este proyecto ni para las ventajas que originalmente ofrecía.**

Causas:

- Carencia de herramientas de análisis léxico/semántico.
- El modelo en sí mismo es impráctico.

Acciones a considerar:

- Posibilidad #1: continuar y tratar de crear las herramientas de análisis léxico/semánt

Contingencia:

- Cambiar de protocolo.

Cambios:

2004-07-22: se detalló el riesgo.

Clave: 23
Nombre: El modelo de datos no gestiona la seguridad.
Estado: Pendiente.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.9 Ponderación: 0.8 Producto: 0.65
Fecha de registro del riesgo: 2004-07-22
Descripción:

La seguridad no se ve reflejada en el modelo de datos persistentes ni en el diagrama entidad-relación.

Causas:

- Falta una visión integral de la seguridad.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Gestionar la seguridad.
- Continuar con el proceso de desarrollo poniendo énfasis en que el modelo de datos gestione la seguridad.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

Clave: 25
Nombre: Falta de subsistemas e interfases.
Estado: mitigado.
Precisión: 4 Impacto: 0.9 Probabilidad: 0.2 Ponderación: 0.8 Producto: 0.14
Fecha de registro del riesgo: 2004-07-22
Descripción:

El diseño hecho hasta ahora es un refinamiento del patrón genérico de realización de casos de uso en análisis. Hay una primera tentativa de subsistemas a partir del caso de uso leer/navegar en donde aparecen dos subsistemas, pero es muy pobre aún y no tiene un alcance muy amplio.

Causas:

- Faltan varios casos de uso importantes y otros son muy ambiguos.
- Falta de paquetes de análisis.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Hacer una adecuada gestión de requisitos.
- Continuar con el proceso de desarrollo poniendo énfasis en los subsistemas e interfases procurando que tengan alta cohesión y bajo acoplamiento.

Contingencia:

- Rediseñar la arquitectura.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→mitigado
Probabilidad: 0.9→0.2
Ya están especificados los subsistemas e interfases más importantes, queda pendiente probar 2 de 3.

Clave: 26
Nombre: Falta de paquetes en análisis.
Estado: mitigado.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.5 Ponderación: 0.8 Producto: 0.23
Fecha de registro del riesgo: 2004-07-22
Descripción:

El análisis hecho hasta ahora se ha basado en los datos persistentes y en un patrón genérico de casos de uso que empieza a ser rebasado.

Causas:

- Faltan varios casos de uso importantes y otros son muy ambiguos.
- Falta analizar las relaciones entre los diversos modelos del análisis.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Hacer una adecuada gestión de requisitos.
- Continuar con el proceso de desarrollo poniendo énfasis en los paquetes de análisis.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→mitigado.
Probabilidad: 0.9→0.5
Ya están los paquetes críticos, falta el resto.

Clave: 28
Nombre: Ambigüedad en la forma de involucrar los casos de uso y los módulos en el sistema.
Estado: mitigado.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.2 Ponderación: 0.8 Producto: 0.15
Fecha de registro del riesgo: 2004-07-22
Descripción:

Desde el punto de vista del diseño, el sistema tiene una parte que presenta los datos al usuario, un núcleo que elige qué información procesar y varios módulos que realizan las peticiones del usuario. El riesgo consiste en que la forma de diseñar/implementar los casos de uso hasta ahora ha sido de manera aislada y no de manera conjunta como un todo.

Causas:

- Está pendiente diseñar la arquitectura en términos de interfases y subsistemas, así como de requisitos no funcionales.

- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Continuar con el proceso de desarrollo poniendo énfasis en las interfases y los subsistemas.

Contingencia:

- Rediseñar la arquitectura.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: Estado: pendiente→mitigado
Probabilidad: 0.9→0.2
Ya es clara la forma de realizar los módulos e integrarlos todos. Queda pendiente normalizar los casos de uso a las nuevas normas.

Clave: 29
Nombre: Formato de resultado de los módulos inexistente.
Estado: resuelto.
Precisión: 5 Impacto: 0.9 Probabilidad: 0.0 Ponderación: 0.8 Producto: 0.0
Fecha de registro del riesgo: 2004-07-22
Descripción:

Desde el punto de vista del diseño, el sistema tiene una parte que presenta los datos al usuario, un núcleo que elige qué información procesar y varios módulos que realizan las peticiones del usuario. El riesgo consiste en que no se ha implementado un formato de los datos que serán transmitidos entre los módulos y el núcleo central, llamado sistema de navegación.

Causas:

- Está pendiente diseñar la arquitectura en términos de interfases y subsistemas, así como de requisitos no funcionales.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Continuar con el proceso de desarrollo poniendo énfasis en las interfases y los subsistemas.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→resuelto
Probabilidad: 0.9→0.0
Se creó la especificación “formato-datos-1.00.x.txt” que actualmente va en su revisión 2. Se creó la especificación: “modules-response-1.00.x.dtd” que actualmente va en su revisión 1.

Clave: 27
Nombre: Ambigüedad en los datos a transmitir a la interfaz general de usuario.
Estado: resuelto.
Precisión: 5 Impacto: 0.8 Probabilidad: 0.0 Ponderación: 0.8 Producto: 0.0
Fecha de registro del riesgo: 2004-07-22
Descripción:

Desde el punto de vista del diseño, el sistema tiene una parte que presenta los datos al usuario, un núcleo que elige qué información procesar y varios módulos que realizan las peticiones del usuario. El riesgo consiste en que el formato de datos entre el núcleo de la información (sistema de navegación) y el sistema que presenta los datos al usuario (interfaz general de usuario) es muy ambiguo, lo cual hace muy dependientes un subsistema de otro.

Causas:

- Está pendiente diseñar la arquitectura en términos de interfases y subsistemas, así como de requisitos no funcionales.
- Es parte del proceso de desarrollo que aún no se cubre.

Acciones a considerar:

- Continuar con el proceso de desarrollo poniendo énfasis en las interfases y los subsistemas.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→resuelto.

Probabilidad: 0.9→0.0

Se creó la especificación “formato-datos-1.00.x.txt” que actualmente va en su revisión 2. Se creó la especificación “content-1.00.x.dtd” que actualmente va en su revisión 0. Se creó la especificación “menu-1.00.x.dtd” que actualmente va en su revisión 1.

Clave: 30

Nombre: Dificultad para encontrar en qué momento cargar los módulos.

Estado: Resuelto.

Precisión: 5 Impacto: 0.8 Probabilidad: 0.0 Ponderación: 0.8 Producto: 0.0

Fecha de registro del riesgo: 2004-07-22

Descripción:

Desde el punto de vista del diseño, el sistema tiene una parte que presenta los datos al usuario, un núcleo que elige qué información procesar y varios módulos que realizan las peticiones del usuario. El riesgo consiste en que los módulos contienen clases que se deben declarar, y no pueden ser declaradas dentro de una función, entonces eso introduce una complicación algorítmica para incluir clases.

Causas:

- Restricciones del lenguaje de programación.
- En el plan se tiene contemplado que una clase cargue los módulos necesarios para utilizarse.

Contingencia:

- Buscar asesoría.

Acciones a considerar:

- Definir correctamente la arquitectura, las interfases y subsistemas para ubicar el problema en un solo punto.
- Buscar sustitutos para la exclusiva carga de los módulos, realizar pruebas e incorporar los datos en la arquitectura.

Cambios:

2004-07-22: se detalló el riesgo.

2004-07-23: se comprobó que el riesgo no existía, entonces la probabilidad pasa de ser 0.9 a ser 0 (cero) y el estado pasa de ser “Pendiente” a ser “Resuelto”.

Clave: 17

Nombre: Malas interfaces.

Estado: mitigado.

Precisión: 5 Impacto: 1.0 Probabilidad: 0.2 Ponderación: 0.8 Producto: 0.16

Fecha de registro del riesgo: 2004-07-22

Descripción:

Las interfaces que se tienen hasta ahora sólo responden a soluciones parciales.

Causas:

- Conocimiento deficiente de los requisitos.
- Análisis deficiente.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Poner atención a los requisitos y al análisis.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: pendiente→mitigado
Probabilidad: 0.7→0.2
Las interfases existentes hasta ahora son claras, están jerarquizadas y estandarizadas, queda pendiente probar 2 de 3.

Clave: 18
Nombre: Malos subsistemas.
Estado: mitigado.
Precisión: 5 Impacto: 1.0 Probabilidad: 0.2 Ponderación: 0.8 Producto: 0.16
Fecha de registro del riesgo: 2004-07-22
Descripción:

Los subsistemas hechos hasta ahora tienen el riesgo de ser ineficaces para responder a los requisitos del sistema.

Causas:

- Conocimiento deficiente de los requisitos.
- Análisis deficiente.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Poner atención a los requisitos y al análisis.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→mitigado
Probabilidad: 0.7→0.2
Se ha estandarizado la arquitectura en cuanto al diseño de subsistemas e interfases principales, el resto se ajustará a las normas. Pendiente realizar pruebas.

Clave: 5
Nombre: Arquitectura frágil.
Estado: mitigado.
Precisión: 4 Impacto: 0.8 Probabilidad: 0.4 Ponderación: 0.8 Producto: 0.26
Fecha de registro del riesgo: 2004-07-22
Descripción:

La arquitectura obtenida hasta ahora no permite muchos cambios en los requisitos y presenta serias dificultades en la integración.

Causas:

- La primera iteración se concentro en analizar los casos de uso por separado y en investigar la eficacia del patrón genérico.
- Falta de subsistemas paquetes de análisis, claridad en los requisitos, etc.
- Falta de una visión integradora.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Gestionar los requisitos de manera más precisa.
- Realizar el análisis usando paquetes de análisis.
- Gestionar el diseño usando subsistemas e interfases.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→mitigado
Probabilidad: 0.8→0.4

El diseño a nivel de interfases y subsistemas de alto nivel ha alcanzado un nivel adecuado de madurez. El subsistema cargador de módulos ha probado ser estable y robusto. La configuración se realiza mediante parámetros y archivos XML. La operación de un subsistema/clase es mediante la interfaz. Queda pendiente revisar la consistencia del modelo de datos (Diagrama Entidad-Relación).

Clave: 6
Nombre: Alto acoplamiento de objetos, subsistemas, etc.
Estado: Pendiente.
Precisión: 5 Impacto: 0.8 Probabilidad: 0.4 Ponderación: 0.8 Producto: 0.26
Fecha de registro del riesgo: 2004-07-22
Descripción:

Los objetos basados en el patrón genérico de análisis y, en general, todos los objetos específicos del sistema son muy dependientes de la estructura de la base de datos y su configuración es mediante el propio código.

Causas:

- La primera iteración se concentro en analizar los casos de uso por separado y en investigar la eficacia del patrón genérico.
- Falta de subsistemas paquetes de análisis, claridad en los requisitos, etc.
- Falta de una visión integradora.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Gestionar los requisitos de manera más precisa.
- Realizar el análisis usando paquetes de análisis.
- Gestionar el diseño usando subsistemas e interfases.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: estado: pendiente→mitigado.
Probabilidad: 0.8→0.4
Ya hay interfases y subsistemas. La configuración se realiza mediante parámetros y archivos XML. La norma “formato-datos-1.00.x.txt” establece formatos XML para los distintos subsistemas en donde se resuelve el problema de los enlaces y direcciones URL internas y externas.

Clave: 20
Nombre: En el análisis no se gestiona la seguridad.
Estado: mitigado.
Precisión: 4 Impacto: 0.8 Probabilidad: 0.2 Ponderación: 0.8 Producto: 0.13
Fecha de registro del riesgo: 2004-07-22
Descripción:

En términos generales no se gestiona la seguridad en el análisis, únicamente se hace mención a que se verifican los datos de sesión pero no hay una especificación de las políticas de seguridad ni de componentes de seguridad adicionales.

Causas:

- Falta implementar la seguridad en general.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Hacer una especificación formal de la seguridad en términos de políticas, servicios y modelos de seguridad.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→mitigado.

Probabilidad: 0.8→0.2

Ya se gestiona la seguridad en los casos de uso que se ajustan a las nuevas normas. Pendiente los demás, pero es parte del mismo proceso de re-análisis, re-diseño que de todas formas se realizará.

Clave: 21

Nombre: Análisis parcial y poco cohesionado.

Estado: mitigado.

Precisión: 5 Impacto: 0.8 Probabilidad: 0.6 Ponderación: 0.8 Producto: 0.38

Fecha de registro del riesgo: 2004-07-22

Descripción:

El análisis en general se ha centrado sobre cuestiones parciales y carece de una visión integral del sistema, provocando que las soluciones sean igualmente parciales y deficientes.

Causas:

- Conocimiento deficiente de los requisitos.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Poner atención a los requisitos.
- Depurar el modelo de análisis eliminando las redundancias y los equivocos.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→mitigado

Probabilidad: 0.8→0.6

Se ha reestructurado el análisis mediante paquetes de análisis que contemplan la integración.

Mejoras en la redacción del riesgo.

Clave: 24

Nombre: Análisis y diseño fusionados.

Estado: mitigado.

Precisión: 4 Impacto: 0.8 Probabilidad: 0.4 Ponderación: 0.8 Producto: 0.26

Fecha de registro del riesgo: 2004-07-22

Descripción:

Con excepción del caso de uso leer/navegar, el diseño no ha sido más que el refinamiento del análisis, el cual no ha sido más que hacer modificaciones sobre un patrón de análisis y de comportamiento, esto le da un alto acoplamiento y una alta cohesión al sistema, así como una gran dificultad para integrar lo ya realizado a un esquema más general.

Causas:

- La primera iteración se concentro en analizar los casos de uso por separado y en investigar la eficacia del patrón genérico.
- Poca comprensión de los subsistemas, las interfases y el diseño.
- Es una parte del proceso de desarrollo que todavía no se cubre.

Acciones a considerar:

- Gestionar los requisitos de manera más precisa.
- Realizar el análisis usando paquetes de análisis.
- Gestionar el diseño usando subsistemas e interfases.
- Continuar con el proceso de desarrollo.

Contingencia:

- Buscar asesoría.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→mitigado

Probabilidad: 0.8→0.4

Ya se hizo la separación en los subsistemas, falta en los casos de uso pero el camino para hacerlo está claramente señalado.

2.4.4.7.3. CATEGORÍA: GRUPO IV: requisitos, ámbito del sistema, calidad.

Clave: 19

Nombre: Modelo de casos de uso incompleto, vago e inconsistente.

Estado: mitigado.

Precisión: 5 Impacto: 1.0 Probabilidad: 0.2 Ponderación: 1.0 Producto: 0.2

Fecha de registro del riesgo: 2004-07-22

Descripción:

El modelo de casos de uso presenta varias ambigüedades, por ejemplo: el caso de uso leer/navegar. Además es muy incompleto pues faltan los casos de uso de edición y eliminación.

Causas:

- Falta de comprensión de los requisitos.
- Presión por el espacio reducido para diagramar los requisitos.

Acciones a considerar:

- Separar el modelo en paquetes por actor.
- Por cada actor anotar todos sus casos de uso de manera precisa.
- Si el modelo se hace muy complejo usar paquetes para manejar la complejidad.
- Documentar ampliamente los casos de uso más importantes.
- Anotar los requisitos no funcionales asociados a cada caso de uso o en general.

Contingencia:

- Refinar los requisitos con el cliente.

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→mitigado

Probabilidad: 0.7→0.2

Ya están capturados la mayoría de los casos de uso. Están separados por paquetes y están especificados y diferenciados unos de otros.

Clave: 12

Nombre: Requisitos funcionales incompletos.

Estado: mitigado.

Precisión: 5 Impacto: 1.0 Probabilidad: 0.2 Ponderación: 1.0 Producto: 0.2

Fecha de registro del riesgo: 2004-07-22

Descripción:

El conocimiento acerca de lo que debe hacer el sistema es deficiente. Se explica con más amplitud en el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”.

Causas:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Acciones a considerar:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Contingencia:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Cambios:

2004-07-22: se detalló el riesgo.

2004-08-09: estado: pendiente→mitigado

Probabilidad: 0.5→0.2

Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Clave: 13
Nombre: Requisitos no funcionales incompletos.
Estado: Pendiente.
Precisión: 5 Impacto: 1.0 Probabilidad: 0.5 Ponderación: 1.0 Producto: 0.5
Fecha de registro del riesgo: 2004-07-22
Descripción:

Los datos adicionales como el rendimiento, capacidad, etc., son ambiguos Se explica con más amplitud en el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”.

Causas:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Acciones a considerar:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”
- Plan de requisitos especiales (pendiente)

Contingencia:

- Vea el riesgo 19: “Modelo de casos de uso incompleto, vago e inconsistente”

Cambios:

2004-07-22: se detalló el riesgo.
2004-08-09: se añadieron acciones a considerar.

2.4.4.8. Procedimientos de prueba

2.4.4.8.1. Contenido

- Pruebas de unidad.
- Pruebas de integración.
- Pruebas de caja negra.
- Pruebas de caja blanca.
- Casos de prueba.
- Pruebas de sistema.
- Errores de complejidad mayor.
- Procedimiento.

2.4.4.8.2. Pruebas de unidad

Cada clase y cada método serán probados individualmente mediante componentes de prueba sencillos para verificar que cumplan con lo especificado. En caso de error, se procede a una depuración y a pruebas de caja blanca hasta encontrar el error y corregirlo.

2.4.4.8.3. Pruebas de integración

Los subsistemas y las interfases serán probados de manera conjunta, si falta alguno se utilizará un *stub* para suplirlo. También se usarán componentes de prueba sencillos que arrojarán la información necesaria para verificar que los subsistemas e interfases cumplen con los objetivos, en caso de error se procederá a depurar y a pruebas de caja blanca hasta encontrar y corregir el error.

2.4.4.8.4. Pruebas de caja negra

En principio todo se prueba con caja negra a menos que se encuentre algún error. Cana negra quiere decir, comparar los resultados obtenidos con los resultados esperados sin importar el funcionamiento interno de aquello que se prueba.

2.4.4.8.5. Pruebas de caja blanca

Únicamente se enfocan al error detectado y a su depuración. La depuración incluye los valores y el comportamiento interno de la clase, sus propiedades, métodos y su entorno en un momento dado.

2.4.4.8.6. Casos de prueba

Cada caso de uso se prueba en sus diferentes modalidades, tanto normales como anómalas como caja negra, si hay discordancia entre lo obtenido y lo esperado, se prueba por subsistemas e interfases, y se rastrea el error, hasta el nivel de método donde sería depuración de caja blanca.

2.4.4.8.7. Prueba de sistema

Se realiza sobre todo el sistema, y se verifican efectos paralelos, se realiza verificando los casos de uso funcionando juntos. Para esto se dispone de un equipo de voluntarios para probar el sistema y hacer los comentarios de los errores que perciban. Se prueba sobre distintos navegadores, distintos sistemas operativos, distintas conexiones, distintas horas del día, etc. Para esta fase se concentra en la arquitectura y la funcionalidad, así como la integridad y seguridad de los datos con preferencia a la estética.

2.4.4.8.8. Errores de seguridad mayor

Cuando un error es de una complejidad mayor que involucra, además del código, elementos de requisitos, análisis y diseño; entonces se gestiona como riesgo y se le da un tratamiento especial en la agenda.

2.4.4.8.9. Procedimiento

1. Elegir el caso de uso a probar.
2. Elegir el subsistema.
3. Elegir la interfaz.
4. Elegir la clase.
5. Elegir el método.
6. Prueba de unidad: método.
7. Prueba de unidad: clase.
8. Prueba de integración: clase, el resto del subsistema y la interfaz es *stub.*
9. Prueba de integración: interfaz, el resto del subsistema y las interfaces exteriores son *stub*.
10. Repetir 6 a 9 con el resto de las clases e interfases del subsistema usando *stubs* donde no existan clases ni interfases desarrolladas.
11. Prueba de integración del subsistema con otro subsistema: usar *stubs* donde haya código incompleto.
12. Repetir 11 hasta completar los subsistemas incompletos y completar el caso de uso.
13. Prueba del caso de uso con sus diversos valores.
14. Probar sistema con los casos de uso disponibles y los usuarios y entornos diversos.
15. Gestionar riesgos y resultados.

2.4.4.9. Especificación de seguridad

2.4.4.9.1. Requisitos de acuerdo a cada servicio de seguridad

Disponibilidad:

La información debe ser disponible las 24 horas del día 7 días de la semana de forma concurrente para todo el que la solicite con las limitaciones de hardware y software del proveedor.

Privacidad:

La información se clasifica en tres grupos de privacidad: contenido público, contenido privado y contenido de configuración/administración. El contenido público carece de restricciones de privacidad. El contenido privado puede ser visto únicamente por los consejeros. El contenido de configuración/administración puede ser visto únicamente por los consejeros autorizadores. Los consejeros son dados de alta previamente. La información de sistema es considerada en otro rubro de clasificación reservada exclusivamente a la cuenta dada por el proveedor de hospedaje Web a la institución.

Integridad:

Respecto a la integridad hay tres grupos: el público, los consejeros y los autorizadores. El público no puede modificar la información. Los consejeros únicamente pueden publicar contenido. Los autorizadores pueden crear, modificar y eliminar información, tanto de usuarios como de temas y documentos.

Autenticidad:

Debe estar garantizada la autenticidad de los documentos publicados en el sentido de qué consejero los publicó y cuándo.

Control de acceso:

Existen diversos servicios para los cuales se requiere la identificación de los usuarios, sean estos público en general (sin identificación), consejeros en general o autorizadores.

No repudio:

En este sentido los requisitos son que todo documento publicado debe pasar por un control de autorización, dicho control lo llevan a cabo los autorizadores. Por tanto, ningún consejero que no sea autorizador puede modificar o suprimir un documento que haya publicado. Por otra parte, la modificación de documentos se realiza en un módulo (caso de uso) distinto de la publicación. Finalmente, el proveedor del servicio ofrece bitácoras con registros sobre las operaciones realizadas a nivel de sistema.

2.4.4.9.2. Ámbito

El ámbito en que se va a gestionar la seguridad es el del software específico del sistema, a saber: la base de datos y los programas y componentes que gestionan el sitio, así como la configuración del sistema software que se pretende construir. Queda fuera de esta gestión, la seguridad a nivel sistemas operativos, firewall, red, sistema manejador de bases de datos, servidor HTTP, etc., que son responsabilidad del proveedor del servicio.

2.4.4.9.3. Modelo

Se propone un modelo híbrido basado en los modelos Bell-La Padula y Biba, para garantizar la privacidad y la integridad; este modelo se presenta en una versión informal y con algunas variantes que a continuación se explican; el propósito es ofrecer una seguridad básica y de bajo costo de implementación pero consistente.

Características del modelo híbrido:

1. Los niveles de privacidad y de integridad se tratan de manera independiente.
2. Los niveles son asignados a roles y a casos de uso y no a usuarios individuales.
3. Cualquier camino fuera del ordinario para modificar el sistema u obtener información debe ser impedido.
4. El grado de formalidad y complejidad del modelo se limitará al alcance de los objetivos.

Roles de seguridad:

1. Roles a nivel sistema de archivos: administrador de sistema y usuarios del sistema.
2. Roles de privacidad: público en general, consejeros, autorizadores.
3. Roles de integridad: público en general, consejeros ordinarios y consejeros autorizadores.
4. Roles de control de acceso: usuarios con sesión y usuarios sin sesión.
5. Roles en el no repudio: consejeros, autorizadores.

Mediante estas características y estos roles establecemos las siguientes reglas, dejando pendiente las tablas del modelo y las distintas graduaciones:

1. **Regla de sistema de archivos:** Sólo el administrador del sistema tiene acceso al sistema de archivos y a la base de datos. La información que cae en este rubro es la configuración del sistema, la contraseña de la base de datos, la organización del sitio como sistema de archivos, etc.
2. **Regla de privacidad:** Los niveles de privacidad: público, privado, configuración/administración; corresponden a los roles: público en general, consejeros, autorizadores. Y operan con el modelo Bell-La Padula donde público es el nivel inferior y configuración/administración es el nivel superior.
   La información que cae en este rubro son los documentos y similares, las cuentas de usuarios, y los temas de navegación.
3. **Regla de integridad:** Los niveles de integridad: público, privado, configuración/administración corresponden a los roles de integridad, público, consejeros ordinarios, consejeros autorizadores; donde público es el nivel inferior y configuración/administración es el nivel superior; el público no puede modificar ningún contenido del sitio. Esta regla opera con el modelo Biba.
   La información de este rubro es la misma que en la regla 2.
4. **Regla de control de acceso:** Se considera consejero y consejero autorizador a aquellos que hayan iniciado sesión, así como el administrador de sistema. Todo usuario sin sesión se considera público en general.
5. **Regla de no repudio:** Un consejero ordinario puede publicar un documento pero no borrarlo ni alterarlo una vez que lo ha enviado. Existe una comisión de autorizadores de confianza que revisan los documentos y los autorizan o los rechazan pero no los alteran. La responsabilidad es compartida. El sistema lleva registro de las acciones de

bases de datos, accesos al sitio y operaciones del sistema de archivos que el administrador examina con motivos de seguridad.

6. **Regla de aplicación de las reglas:** Las reglas se aplican sobre los contenidos a los que se refieren y sobre los casos de uso que tratan esos contenidos.
7. **Regla de frontera:** No hay acceso al sistema de archivos ni a la base de datos ni a los servicios del sistema fuera de los canales ordinarios.

2.4.4.9.4. Propuestas frente a problemas concretos

1. Verificación de la sesión redundante.
2. El modo de transmitir la información de la sesión de un componente a otro es mediante las interfases y canales establecidos.
3. Reducir el uso de *cookies* y variables globales a lugares específicos.
4. Compresión y cifrado de archivos en el servidor.
5. Cifrado de una sola vía de contraseñas en el servidor.
6. Generación aleatoria de contraseñas (si alcanza el tiempo) y gestión de las mismas en un periodo de tiempo determinado.
7. Gestión de riesgos.

2.4.4.10. Propuesta de protocolo de comunicación de datos entre subsistemas

2.4.4.10.1. Revisiones

20040727:
[0.0.0] se creo este documento:
- se establecieron los requisitos.

20040728:
[1.0.0] se llego al estándar completo
- se soluciono el problema de los formularios
- se soluciono el problema de los enlaces
- pendiente la corrección de errores que vayan apareciendo.

20040729:
[1.0.1] cambio de formato de datos.
- se deja protocolo-1.0.0.xsd por:
  . Inmadurez/inestabilidad del estándar
  . Complejidad en los datos.
  . La implementación supone un retraso mayor al beneficio que se espera de acuerdo al objetivo de ahorrar tiempo.
- se cambia a modules-response-1.0.0.dtd por:
  . Estable y maduro.
  . Sencillo de entender e implementar.
- se renuncia a la total separación de formato y datos en su lugar los datos son HTML sin estilo, el cual se usa con *ID* y *class* que varían en las hojas de estilos.
- se salva lo de los enlaces internos y externos tanto en redirecciones, como en enlaces, formularios, *scripts*, etc.
- todo lo que sea texto de usuario se recomienda y es de hecho la configuración por defecto codificado URL

20040812:
[2.0.0]
- cambios en la redacción: mejor conceptualización del requisito de las URL
- cambio en el formato de datos, se detallaron los requisitos para el nuevo formato.

2.4.4.10.2. Contenido

- Ámbito.
- Requisitos
- Solución a problemas concretos.

2.4.4.10.3. Ámbito

El protocolo de comunicación de datos tiene como finalidad la transmisión de información entre los subsistemas a través de sus interfases, a saber, entre los módulos y el sistema de navegación; y entre el sistema de navegación y la interfaz general de usuario.

Más específicamente nos referimos a la información de salida, es decir, de los módulos al sistema de navegación y del sistema de navegación a la interfaz general de usuario, que es la que se muestra al usuario.

Dos son las formas posibles de implementarlo: mediante arreglos asociativos y mediante XML, cada una con sus ventajas y desventajas.

Por lo pronto se deja abierto en lo que se detallan los requisitos.

La especificación se realizará como si fuera XML dado que es más formal hacerlo así, por ejemplo mediante DTD o *Schema*.

2.4.4.10.4. Requisitos

Para conocer los requisitos hay que conocer la información que cada caso de uso va a enviar al cliente y el comportamiento que cada módulo va a tener en cada caso, al menos en sus rasgos más generales.

Esto quiere decir, formularios, listas, textos, redirecciones, enlaces, etc.

En primer lugar, se requiere información general sobre la versión del protocolo de formato y del estándar usado; en caso de ser XML existe una forma de indicarlo para su validación como DTD o *Schema*.

Cuando se invoca un modulo, existen diversos tipos de respuesta, los cuales pueden ser, el error, la redirección y el contenido.

En cuanto al error, algunos módulos lo tratan internamente y lo traducen en redirección o en contenido. De donde solo que no sea posible cargar el módulo existiría esta señal.

Algunos módulos no utilizan este protocolo que es exclusivamente de salida.

De donde podemos obtener los tres grandes tipos de respuesta:

- **error:** no se cargo/ejecuto el modulo o no existe.
- **redirección**: ejecutar otro modulo (o el mismo) con los parámetros indicados a continuación (del módulo a *redireccionar*).
- **contenido**: mostrar al usuario el siguiente contenido el cual es independiente del formato.

Queda pendiente discutir si existe la respuesta "vacía".

El "error" debe tener un tipo, un modulo, un texto y si se desea depurar, tiene un campo adicional libre para añadir toda la información que sea necesaria de los subsistemas, clases, variables, etc., y así poder dar seguimiento al error.

La "redirección" puede ser interna o externa, si es externa se trata de una pagina de Internet en otro servidor, y si es interna se trata de un módulo.

Cuando la "redirección" es externa se requiere la URL completa incluyendo los parámetros GET de la forma en que se escribiría en el navegador.

Cuando la "redirección" es interna, los parámetros son: el modulo, el tema de navegación y los parámetros propios del modulo con sus respectivos valores. Cuando el modulo falta, se asume el modulo por defecto (navegar) y cuando falta el tema se asume el tema raíz.

El contenido es la parte más amplia y difícil de especificar.

Ante todo, el contenido debe estar lo más separado que se pueda del formato.

El contenido usado por los casos de uso es:

- textos.
- enlaces externos.
- enlaces de módulos.
- formularios etiquetados.
- listas.
- detalle de elementos.
- tablas inteligentes.
- Etiquetas.
- Títulos.
- Estadísticas y graficas (pendiente especificar).
- anuncios (pendiente especificar).
- *Resaltadores* (aplican solo a texto y enlaces).

Todos los elementos se pueden identificar con un ID y pueden tener una clase, que serán usados en el estilo *css*, esto es lo único que se usara para indicar el formato.

El problema de las direcciones URL consiste en que un modulo no debe alterar su código si se decide cambiar el sistema de navegación o el servidor o las carpetas donde se encuentran las cosas a las que desea acceder el módulo.

Por tanto se llega al concepto de estado de la siguiente manera:

- un estado es una forma única de operación del sistema.
- un estado posee un único tema de navegación
- un estado posee un único módulo a ejecutar
- un estado posee una única información de usuario
- un estado posee parámetros adicionales

Esta información es la que cada modulo transmite al sistema de navegación cuando transmite una URL.

2.4.4.10.5. Solución a problemas concretos

Hacer un esquema (*Schema*) o un formato DTD completo de los objetos capaz de incrustar un modelo avanzado de formularios tipo *XForms* y luego procesarlo desde servidor resulta una tarea que quita demasiado tiempo, quizá más tiempo del que se supone que ahorraría la creación de este protocolo y separar en módulos los casos de uso.

Por tanto para este proyecto se desiste de ese estándar en lo que madura la idea. En su lugar se usa un DTD muy sencillo que salva el problema de las direcciones URL y de los estilos CSS, así como la respuesta principal del modulo, a saber, si fue error, redirección o contenido.

El estándar dado en `modules-response-1.0.x.dtd' resuelve teóricamente el problema, pero su implementación introduce muchas complicaciones tanto en los módulos como en el sistema de navegación, por tanto, es necesario crear una nueva versión que sea fácil de implementar en la práctica y adoptar medidas que resuelvan el problema del acoplamiento entre formato y datos y entre módulos y servidor (*URI/URL's*).

De acuerdo al concepto de estado explicado en el apartado anterior, y dadas las complicaciones que presenta el dar el estado abstracto desde el modulo y que el sistema de navegación lo traduzca en direcciones reales; se sugiere enviar al modulo la información para transformar el enlace o la URL en dirección efectiva a partir de la información de estado; i.e. la dirección base del servidor y el programa navegador mas algunos parámetros adicionales (por lo pronto ningún parámetro adicional).

Para distinguir entre enlaces internos y externos se pueden usar atributos ID que además se pueden acoplar con hojas de estilo CSS para lograr cosas como: mostrar enlaces externos en ventanas distintas del navegador.

Resta solucionar el problema de cargar un HTML dentro de un XML, esto se puede solucionar mediante una traducción de los símbolos HTML que tienen significado en XML, a saber "<",">", ","&".

Esto se realiza de manera transparente para el desarrollador mediante la *API* llamada *domxml* que es un estándar de la W3C y que está implementada en PHP.

Este estándar se limita por el momento a la comunicación entre los módulos y el sistema de navegación pudiendo ser extensible a la interfaz general de usuario posteriormente.

2.4.4.11. modules-response-2.00.00.dtd

Listado 2.4.4-1: Archivo DTD para especificar el formato de respuesta de los módulos (XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 sencillo version 1.0 todo el contenido deberia ser codificado URL
 para evitar problemas con entidades HTML en lo que se prepara
 una version avanzada de DTD o Schema para los datos
-->
<!--
revisiones:
==========
  fecha: 20040812
  version/revision: 2.0.0
  notas: se reformo el elemento `content' y el elemento `redirection'.
  autor: Gustavo Serrano
```

```
  correo: fru <elfrucool@tutopia.com>

  fecha: 20040808
  version/revision: 1.0.1
  notas: se añadio el campo `title` al elemento `content`
  autor: Gustavo Serrano.
  correo: fru <elfrucool@tutopia.com>

  fecha: 20040729
  version/revision: 1.0.0
  notas: se creo este documento.
  autor: Gustavo Serrano.
  correo: fru <elfrucool@tutopia.com>

-->
<!ELEMENT response (error | redirection | content)>
<!ELEMENT error (type, module, msg, debug)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT module (#PCDATA)>
<!ELEMENT msg (#PCDATA)>
<!ELEMENT debug ANY>
<!ELEMENT redirection (#PCDATA)>
<!ELEMENT content (title?,body)>
<!ELEMENT body (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!--URI/URL significa que esta codificado
el contenido lo cual es por defecto-->
<!ATTLIST response
  version CDATA #FIXED "2.0"
>
<!ATTLIST msg
  encoded (URI | URL | NO | simple) "simple"
>
<!ATTLIST content
  version CDATA "2.0"
>
<!ATTLIST title
  encoded (URI | URL | NO | simple) "simple"
>
<!ATTLIST body
  version CDATA "2.0"
  <!--simple:"<"=>"&lt;",">"=>"&gt;","""=>"&quot;","&"=>"&amp;"-->
  encoded (URI | URL | NO | simple) "simple"
  id CDATA #IMPLIED
  class CDATA #IMPLIED
>
<!ATTLIST redirection
  type (internal|external) "internal"
  encoded (URI | URL | NO | simple) "simple"
>
```

2.4.4.12. content-2.00.00.dtd

Listado 2.4.4-2: Archivo que especifica el formato de contenido a transmitir a la interfaz *FormatManagerIF* en XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 este DTD es para comunicar el sistema de nagevacion con
```

```
 la interfaz general de usuario en lo que se refiere al contenido
 central.
-->
<!--
revisiones:
==========
  fecha: 20040817
  version/revision: 2.0.0
  notas: compatible con la version 2.0 de modules-response
  autor: Gustavo Serrano.
  correo: fru <elfrucool@tutopia.com>

  fecha: 20040770
  version/revision: 1.0.0
  notas: se creo este documento.
  autor: Gustavo Serrano.
  correo: fru <elfrucool@tutopia.com>

-->
<!ELEMENT content (#PCDATA)>

<!ATTLIST content
  version CDATA "2.0"
  encoded (URI | URL | NO | simple) "simple"
  id CDATA #IMPLIED
  class CDATA #IMPLIED
>
```

2.4.4.13. menu-2.01.00.dtd

Listado 2.4.4-3: Formato de datos para transmitir el menú de navegación y todo lo que no es el contenido a la interfaz *FormatManagerIF* en XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
  menu DTD:
  version: 2.01.00
  inicio: 20040801
  modificado: 20040920
-->
<!--
  cambios:
  20040920: cambios sustanciales en form y en atributos de version
  20040918: correcciones menores
  20040830: modificaciones a model y a submit en los atributos
  20040817: se creo la version 2.0 simplificada
  20040809: se agrego title, se modifico menuset
-->
<!-- DOCTYPE menuset: (seccion principal) -->
<!ELEMENT menuset (title,(menugroup* | textlabel* | model* | form*)*)>

<!-- bloques principales -->
<!ELEMENT title (#PCDATA)>
<!ELEMENT menugroup (type, title, menuitem*)*>
<!ELEMENT textlabel (type, label*)>
<!ELEMENT type (#PCDATA)>

<!-- bloque de menu -->
<!ELEMENT menuitem (text,link)>
```

```
<!ELEMENT text (#PCDATA)>
<!ELEMENT link (#PCDATA)>

<!-- bloque de textlabel -->
<!ELEMENT label (#PCDATA)>

<!-- bloque de formulario -->
<!ELEMENT model ANY>
<!ELEMENT form (type, method, title?,
            (label | input | secret | textarea |
             check | select1)*,
            submit*)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT submit (#PCDATA)>
<!ELEMENT input (#PCDATA)>
<!ELEMENT secret (#PCDATA)>
<!ELEMENT textarea (#PCDATA)>
<!ELEMENT check (#PCDATA)>
<!ELEMENT select1 (label,item*)>
<!ELEMENT item (text, value)>
<!ELEMENT value (#PCDATA)>

<!-- seccion de atributos -->
<!ATTLIST model
  id CDATA #IMPLIED
>
<!ATTLIST title
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST menuset
  version CDATA #FIXED "2.1"
>
<!ATTLIST link
  encoded (URI | URL | NO | simple ) "simple"
  id CDATA #IMPLIED
  class CDATA #IMPLIED
>
<!ATTLIST label
  encoded (URI | URL | NO | simple ) "simple"
  id CDATA #IMPLIED
  class CDATA #IMPLIED
>
<!ATTLIST text
  encoded (URI | URL | NO | simple ) "simple"
  id CDATA #IMPLIED
  class CDATA #IMPLIED
>
<!ATTLIST field
  ref NMTOKEN #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST input
  ref NMTOKEN #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST secret
  ref NMTOKEN #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
```

```
<!ATTLIST textarea
  ref NMTOKEN #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST check
  ref NMTOKEN #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST select1
  ref NMTOKEN #REQUIRED
  presentation (big | medium | small) "small"
>
<!ATTLIST item
  selected (YES | NO) "NO"
>
<!ATTLIST menugroup
  section CDATA #IMPLIED
>

<!ATTLIST textlabel
  section CDATA #IMPLIED
>

<!ATTLIST form
  section CDATA #IMPLIED
  idref CDATA #REQUIRED
>

<!ATTLIST submit
  idref IDREF #REQUIRED
  encoded (URI | URL | NO | simple ) "simple"
>
<!ATTLIST value
  encoded (URI | URL | NO | simple ) "simple"
>
```

2.4.4.14. Especificación del formato de los parámetros de entrada de los módulos (module-arguments-1.00.00.txt)

2.4.4.14.1. Contenido

- Requisitos.
- Especificacion.

2.4.4.14.2. Requisitos

El sistema central de navegación (subsistema: *NavigateSystem*) requiere un unico orden para transmitir los parámetros a los módulos, ya que debe ser completamente independiente de los módulos establecidos y por establecer.

Los módulos, dependiendo de sus necesidades concretas, necesitan tener la información de la sesión, del tema y determinados parámetros que están dados en la URL por parámetro=valor.

2.4.4.14.3. Especificación

Son 3 los parámetros transmitidos a los módulos en términos de `*argv*' y estos son:

- argv[0]: el nombre del modulo.
- argv[1]: los parámetros del modulo en un arreglo asociativo de la forma parámetro => valor.
- argv[2]: la información de la navegación y de la sesión en un arreglo asociativo con la siguiente forma:
  - userid => *UserID*: ID del usuario.
  - topicid => *TopicID*: ID del tema.
  - topickeycode => *keycode*: clave alfanumérica del tema.
  - urlbase => *urlbase*: dirección base del programa ejecutor del sistema de navegación.

2.5. FASE DE CONSTRUCCIÓN

2.5.1. Introducción

En la fase de construcción desarrollamos el sistema hasta llevarlo a su capacidad operativa inicial, es decir, que el sistema cumple con los requisitos acordados con el cliente al menos en su forma básica, y el sistema puede ser utilizado ya por el cliente, aunque de manera restringida, ya que está sujeto a observación tanto para apreciar la satisfacción del cliente como para detectar posibles defectos que no hayan podido hallarse durante las pruebas; este proceso de ajuste por el uso del cliente y corrección de fallas que no fueron encontradas en el proceso de pruebas corresponde a la fase de transición.

Para este proyecto la fase de construcción se llevó a cabo en dos iteraciones, durante la primera se tuvo el desarrollo más intenso; mientras que la segunda iteración operó sobre los riesgos generados, la gestión de la seguridad, la interfaz de usuario y terminar los casos de uso incompletos.

2.5.2. Primera Iteración

2.5.2.1. Introducción

La primera iteración de la fase de construcción se centró en el desarrollo de todos los casos de uso como quedaron definidos en la fase de elaboración.[1]

Este desarrollo consistió en la realización de los casos de uso en el diseño, el código y las pruebas.

El resultado fue una primera versión (beta) del sistema con la capacidad de realizar todos los casos de uso pero con ciertas limitaciones que se tratarán más adelante.

Para lograr esta versión se trabajó, en primer lugar sobre la arquitectura; además de corregir errores y mejorar la estructura de la base de datos, se hizo un modelo de los datos desde un enfoque orientado a objetos mediante colecciones de objetos que tuvieran persistencia en la base de datos.

Una *colección persistente en base de datos* es un objeto que tiene la capacidad de manipular objetos de determinado tipo y por otra parte realizar consultas a una base de datos para almacenar los datos (propiedades) de los objetos que almacena, esto permite un bajo acoplamiento con la base de datos.

Además del modelo se crearon diversos componentes que facilitaban la construcción de las tareas repetitivas de los módulos y otros para tareas específicas como publicar un archivo o descargarlo.

Partiendo del diagrama central de la arquitectura,[2] se trabajó en el subsistema llamado *modules* en el concepto de que un módulo realiza una tarea específica; así para cada caso de uso se hicieron diversos módulos según las tareas específicas que se necesitaron.

---

[1] Ver la sección *2.4.1. Fase de Elaboración: Descripción de la Arquitectura* del presente trabajo.
[2] *Ibid.*

Merece especial atención el concepto de *rotación automática* que significa el desplazamiento de los documentos viejos al ingresar los documentos nuevos al sistema en el sentido de su *eliminación automática*.

Este concepto es correcto con los documentos aprobados pero ¿qué sucede con los documentos que se envían y quedan pendientes de aprobar?, ¿conviene que un documento que todavía no ha sido aprobado provoque la eliminación de un documento aprobado?, ¿qué sucede si dicho documento es rechazado?

Estas preguntas nos llevaron a una conclusión: la rotación automática opera en el caso de uso *autorizar documento*; en cambio en el caso de uso *publicar documento* es necesario un candado de seguridad que prevea la saturación del espacio y se tomen medidas al respecto; este candado se planteó y se llegó a una solución hasta el nivel de código y pruebas.

Para la *rotación automática* se plantearon y diseñaron las diversas soluciones y sus excepciones, quedando pendiente la implementación para la siguiente iteración.

Las limitaciones concretas del sistema son:

- Rotación automática incompleta (falta implementación y pruebas).
- No está implementada la seguridad.

2.5.2.2. Rotación automática

El principal problema con la rotación automática es ¿cuántos y cuáles documentos hay que eliminar para tener suficiente espacio para el buen funcionamiento del sistema? Otra pregunta no menos importante es ¿cuándo se deben eliminar esos documentos? Y otra es ¿cuánto espacio es considerado *suficiente* sin dejar el sistema demasiado vacío pero con suficiente margen para permitir la publicación de documentos nuevos? Finalmente ¿qué acciones además de eliminar documentos conviene realizar en la rotación automática?

Estas preguntas (y otras similares) además de la pregunta *¿cómo realizarlo?* son las que nos llevaron a plantear la siguiente solución, a nivel de diseño, de la rotación automática.

Es evidente que los documentos a eliminar serán los más antiguos; y no es tan evidente que no serán eliminados aquellos que estén marcados como *permanentes* (a menos que sean eliminados manualmente por un usuario autorizado que es el *autorizador*). Lo que es menos evidente de todo es la cantidad de ellos; para arrojar alguna luz en esta investigación establecemos dos criterios; el primero es el *tope* a partir del cual se requiere la eliminación automática, y el segundo es la *velocidad* con la cual se espera que el cliente llegue a ese tope.

El tope es una constante establecida por el proveedor, que en este caso es de 500MB, ahora bien, resulta conveniente tener un margen de operación debido a que la autorización no se realiza automáticamente sino cuando un *autorizador* hace uso del sistema, además porque el sistema *crece* por diversos caminos y no únicamente mediante la publicación de documentos. De aquí que el criterio establecido es que el tope sea el 90% del espacio concedido por el proveedor.

La velocidad del cliente para acabarse el espacio depende únicamente del cliente y de aquellos usuarios a los que el sistema permita introducir información persistente; en este caso son los *consejeros*.

Conversando con el cliente y realizando una proyección del posible crecimiento institucional (porque el número de sus miembros puede variar) calculamos una tasa de información de

10MB mensuales, lo cual quiere decir que el sitio estará agotando su espacio aproximadamente en dos años.

De acuerdo con este análisis resulta muy eficiente hacer la eliminación de los documentos antiguos por meses en lugar de analizar los documentos uno a uno. La proporción es: de varios cientos (los documentos) a 24 (los meses) y un documento tan antiguo como un año y medio ya se considera obsoleto (a excepción de si es marcado como *permanente*).

La siguiente pregunta es ¿cuánto espacio hay que liberar?, nosotros hemos considerado que lo conveniente para que el sitio posea información y que permita almacenar más información sea eliminar el 10% o sea que si se estaba en el tope del 90% se llegue a un estado de 80%, lo cual en meses es aproximadamente 3 o 4.

Dado que no es lo mismo un promedio que la realidad, es decir, que el flujo de los documentos al servidor es irregular a través del tiempo es necesario hacer un cálculo mensual y realizar la suma hasta obtener el 10% del espacio en disco, entonces se determina el mes en que se obtuvo esa suma y se procede a la eliminación de los documentos anteriores.

TABLA 2.5.2-1: Ejemplo del gasto por mes del espacio en el servidor

Mes	Total
200504	8MB
200505	12MB
200506	7MB
…	…

Resta solamente preguntarnos ¿qué acciones además de la eliminación implica la *rotación automática*? Una de estas acciones sería informar a los *autorizadores* con cierta anticipación, por ejemplo, por correo electrónico, de aquellos documentos que serán eliminados; esto se puede realizar si el cálculo del mes se realiza cuando el sistema se encuentra al 88% de su capacidad (lo cual da un margen de aproximadamente 2 o 3 semanas) y la eliminación se realiza cuando el sistema llega al 90%.

Otra acción conveniente es almacenar el mes calculado (cuando se había llegado al 88%) y así estar seguros de enviar el correo a los *autorizadores* una sola vez y no repetir el cálculo cada vez que un autorizador realiza el caso de uso *autorizar documento*; así cuando se llegue al 90% tampoco se realizará el cálculo sino que se procederá directamente a la eliminación.

De aquí surge la necesidad de almacenar el mes de la eliminación de los archivos viejos; si no existe ningún mes, entonces se hace el cálculo y se envía el aviso a los autorizadores, si existe el mes pero no se ha llegado al 90% se ahorra el cálculo y no se envía el correo. Si existe el mes y se ha llegado al 90% se ahorra el cálculo y se hace la eliminación automática. Existe la posibilidad (aunque se espera que no suceda por el margen de 2 o 3 semanas) de que se llegue al 90% sin hacer el cálculo, en ese caso se haría el cálculo, se enviaría el correo informando de lo que ya está sucediendo y se eliminarían los documentos.

ACTIVIDADES DE LA *ROTACIÓN AUTOMÁTICA*

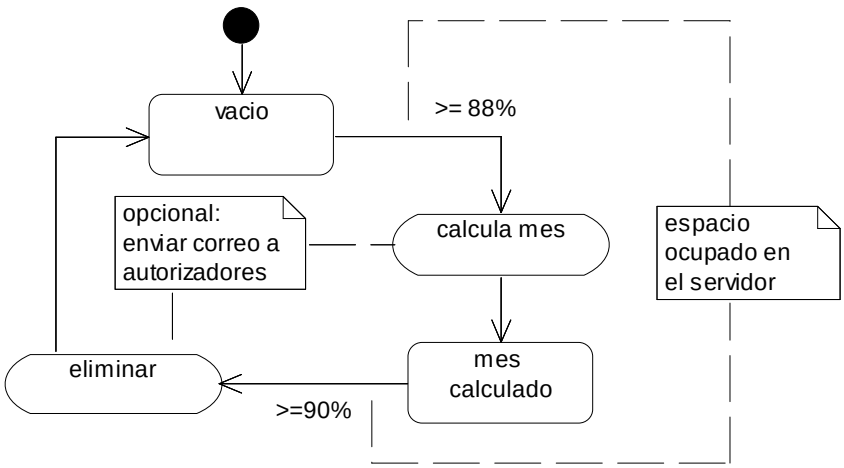

Fig. 2.5.2-1: Actividades de la rotación automática

El *mes* desde el punto de vista de los objetos resulta en un objeto con la propiedad *mes* que incluye el año y es almacenado en una colección (se explica más adelante) de meses, esta colección tiene persistencia en base de datos.

Desde la base de datos se debe de crear la tabla para almacenar el mes.

El caso de uso *autorizar documento* debe incluir una parte que controle la *rotación automática* usando el cálculo del *mes* o eliminándolo de acuerdo a los criterios aquí establecidos.

Para el presente trabajo *no se implementará el servicio de notificación por correo electrónico* quedando propuesto para futuras versiones.

2.5.2.3. Componentes reutilizables nuevos

En la construcción hubo necesidad de desarrollar componentes reutilizables nuevos. Estos componentes pueden ser usados en otros sistemas sin ningún cambio.

A continuación se describen los tres componentes nuevos desarrollados en esta iteración:

- YAFUClass (YetAnotherFileUploader)
- FileDownloadClass
- DbCollectionClass

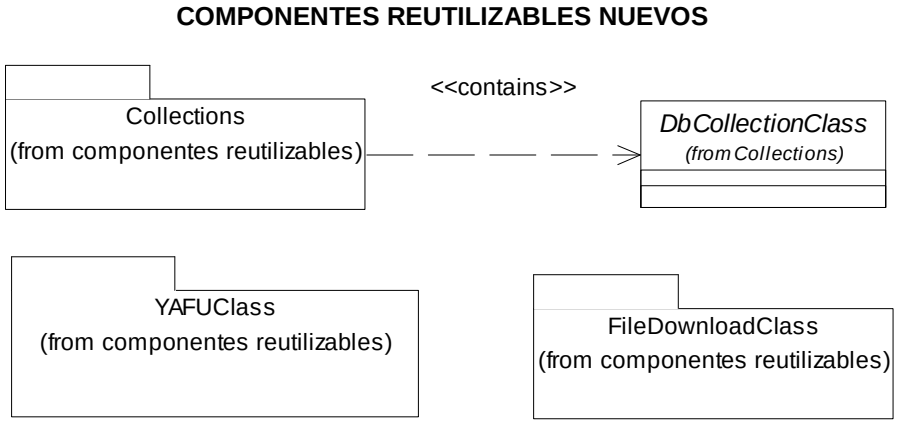

Fig. 2.5.2-2: Componentes reutilizables generados en la primera iteración de la fase de construcción

2.5.2.3.1. YAFUClass (Yet Another File Uploader Class)

Cubre la necesidad de recibir archivos enviados por los usuarios a través de formularios web y guardarlos en el servidor. Por este medio los usuarios pueden ingresar nuevos archivos al sistema, el archivo será recibido y procesado por el sistema una vez que haya sido completado su envío.

Reemplaza al componente reutilizable *FileUploadClass* que es deficiente y muy complicado de utilizar.

2.5.2.3.2. FileDownloadClass

Sirve para que el usuario descargue vía web archivos. Los archivos en el sistema están en una carpeta del servidor a la cual el cliente no tiene acceso, de manera que protege los archivos. Para mayor seguridad los archivos tienen nombre clave, solo mediante la base de datos se pueden reestablecer sus nombres originales para ser enviados al cliente.

Esta clase permite la descarga del archivo enviando al cliente los encabezados HTTP adecuados y posteriormente los datos.

2.5.2.3.3. DbCollectionClass

Esta clase es una colección de objetos con persistencia en base de datos. Encapsula las consultas SQL de los objetos que colecciona, de esta forma se hace independiente el programa de las consultas SQL sobre un objeto determinado. Es abstracta debe heredarse para poderse utilizar.

2.5.2.4. Cambios en la arquitectura

Los cambios en la arquitectura fueron tres:

1. Cambios en la base de datos.
2. Paquete MRMaker.
3. Paquete data_models.

Excepto los cambios en la base de datos, los demás cambios fueron dentro de la *capa lógica del negocio*[3] y dentro de la misma, en el subsistema *modules*.

SUBSISTEMA *MODULES* DE LA FASE DE CONSTRUCCIÓN

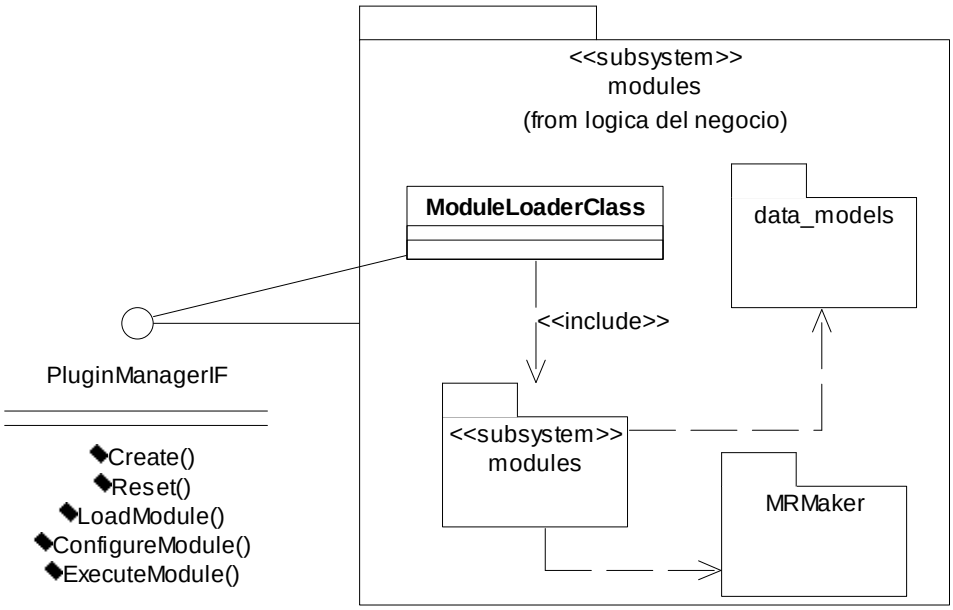

Fig. 2.5.2-3: Subsistema modules *de la fase de construcción*

2.5.2.4.1. Cambios en la base de datos

En la *capa de datos persistentes*[4] la clase *documentos* ahora tiene el atributo *filesize* que va a contener el tamaño del archivo, esto con el objeto de hacer los cálculos de espacio en disco mediante el motor de SQL y ahorrar tiempo.

También se agregó la clase *meses* que almacena el año y mes a partir del cual se deben eliminar los archivos más viejos en caso de que la *rotación automática* así lo determine, el propósito es evitar hacer el cálculo todas las veces.

[3] Ver la sección *2.4.1. Fase de Elaboración: Descripción de la Arquitectura* del presente trabajo.
[4] *Ibid.*

MODIFICACIONES A LA BASE DE DATOS

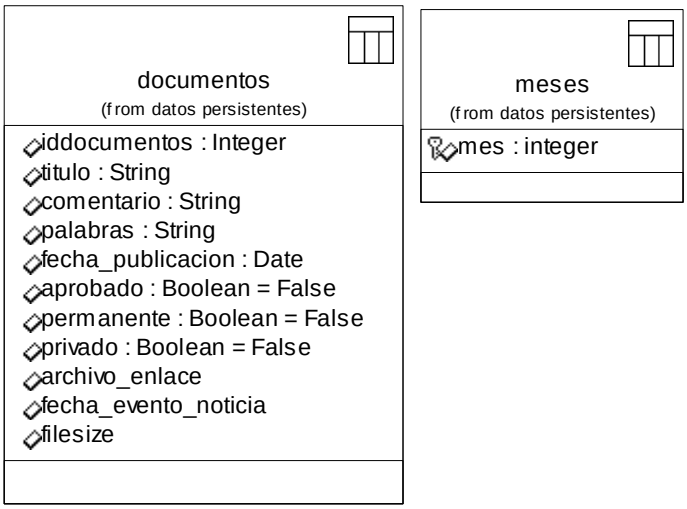

Fig. 2.5.2-4: Modificaciones a la base de datos

Las relaciones quedan igual que antes y la clase *meses* no se relaciona con el resto del modelo en esta capa.

La clase *meses* solamente se diseñó y quedó pendiente su implementación.

2.5.2.4.2. Paquete MRMaker

Contiene la clase MRMakerClass que sirve para generar las respuestas de los módulos de acuerdo al estándar *modules-response-2.00.x.dtd*[5] (XML) de manera sencilla. Presenta otra ventaja: independencia del código y el estándar XML, ya que puede incluso sustituirse XML por otra forma de transmitirse los datos, por ejemplo, objetos y el código de los módulos permanecer sin cambio alguno.

2.5.2.4.3. Paquete data_models

Contiene el modelo de la base de datos desde un enfoque orientado a objetos utilizando el concepto de *colección* como se explica más arriba.

Este paquete está formado por varios paquetes como se muestra en el diagrama:

[5] Ver la sección *2.4.4. Fase de Elaboración: Documentación de la fase de Elaboración* de este trabajo.

COLECCIONES PERSISTENTES DEL PAQUETE *DATA_MODELS*

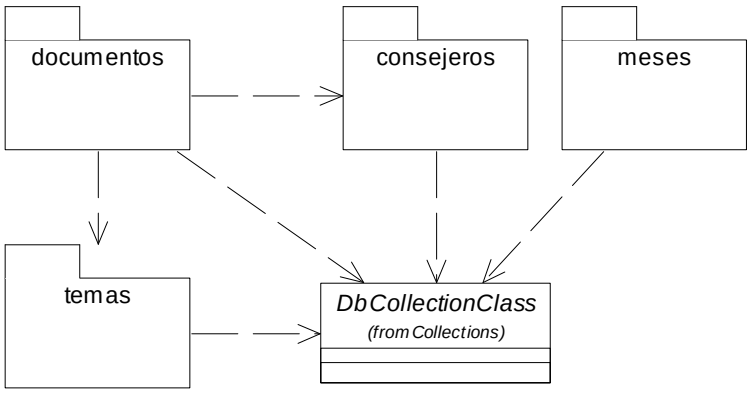

Fig. 2.5.2-5: Colecciones persistentes del paquete data_models

Explicación:

- **DbCollectionClass:** clase para hacer colecciones persistentes en base de datos.
- **meses:** Contiene la clase *MesClass* que representa el mes a partir el cual se deberán eliminar los documentos antiguos y *MesesClass* que es la colección de objetos clase *MesClass* para darles persistencia (por base de datos), aunque en este caso sólo se almacenará un mes, por simplicidad se diseñó de esta forma. **Nota:** todavía no está implementado este paquete.
- **consejeros:** Contiene la clase *ConsejeroClass* que representa a un *consejero* (o a un *autorizador*) en el sistema, y la clase *ConsejerosClass* que es la colección de objetos clase *ConsejeroClass* con persistencia mediante base de datos.
- **temas:** Contiene la clase *TemaClass* que representa a un *tema* de navegación y la clase *TemasClass* que es la colección de objetos clase *TemaClass* con persistencia mediante base de datos.
- **documentos:** Contiene la clase *DocumentoClass* que representa un *documento* en el sistema, y la clase *ArchivoClass* que representa a un *archivo* que es una especie de *documento*; también contiene la clase *DocumentosClass* que es la colección de *documentos* y la clase *ArchivosClass* que es la colección de *archivos*. La razón de estas clases es porque el *documento* según el *modelo de análisis*[6] es una clase abstracta, por tanto su colección también es abstracta, de donde para cada especie de *documento* se requiere una colección específica; para este trabajo sólo se trabajarán *archivos*, por tanto sólo es necesario estas clases; posteriormente en las mejoras para el cliente se agregarán las demás y una *colección maestra* que tenga la capacidad de operar con todas las clases de objetos simultáneamente.

[6] Vea la sección *2.4.1. Fase de Elaboración: Descripción de la Arquitectura* del presente trabajo.

La jerarquía de las colecciones queda así:

JERARQUÍA DE LAS COLECCIONES

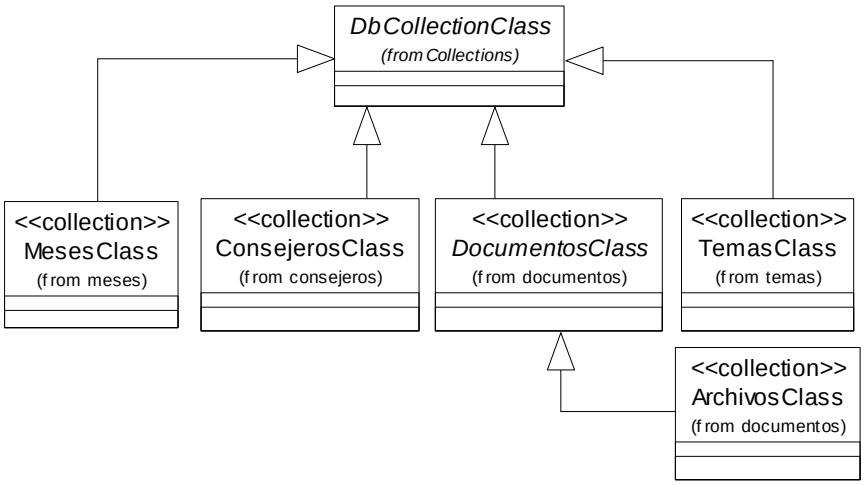

Fig. 2.5.2-6: Jerarquía de las colecciones

Las relaciones de las colecciones quedan así:

RELACIONES DE LAS COLECCIONES

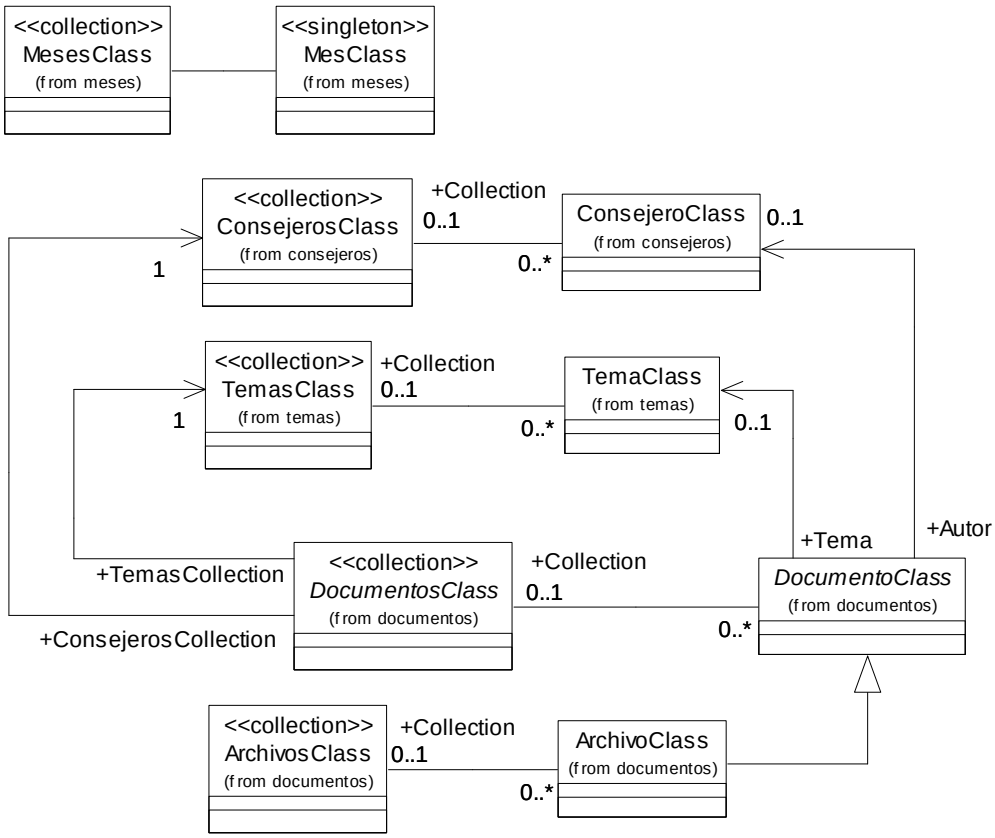

Fig. 2.5.2-7: Relaciones de las colecciones

Una característica importante de las colecciones (*ConsejerosClass, TemasClass, DocumentosClass* y *ArchivosClass*) es la capacidad que tienen de calcular el espacio que ocupan sus datos (incluyendo los archivos) en el servidor; a esto hemos llamado *entropía de la colección* en el sentido de ser un gasto (en bytes) en el servidor que es mayor a la sola suma del tamaño de los archivos que se encuentran en él (comparación termodinámica).

Para obtener esta *entropía* (que no debe confundirse con la entropía de la teoría de la información)[7] se utiliza la operación *get_enthropy()* de cada colección; esta operación consulta la base de datos para medir el tamaño de las tablas que dependiendo de la instalación en el servidor y de la cantidad de registros estos datos varían y no necesariamente en proporción al tamaño y número de registros.

Además, en el caso de la colección *ArchivosClass* se calcula el tamaño ocupado por los documentos en el servidor, para hacer esta operación más rápida, se almacena el tamaño de cada archivo en un campo de la base de datos (por tanto en una propiedad del objeto *ArchivoClass*) y así se obtiene el total del espacio usado mediante una suma ejecutada por el manejador de la base de datos; finalmente, se suma el espacio de la tabla y la suma del tamaño de los archivos recientemente obtenida y ese es el resultado de la operación *get_enthropy()* para la colección *ArchivosClass*.

2.5.2.5. Construcción de los casos de uso

Los casos de uso que se construyeron son siete, para cada uno se generó un paquete como se muestra en el diagrama.

PAQUETES CON LAS CONSTRUCCIONES DE LOS CASOS DE USO

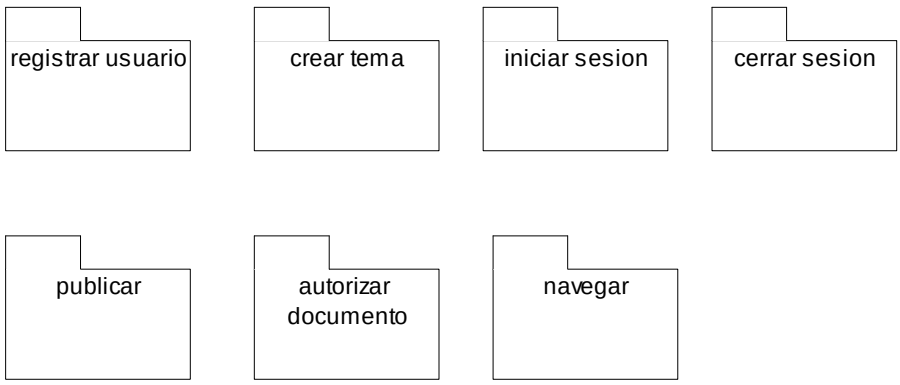

Fig. 2.5.2-8: Paquetes con las construcciones de los casos de uso

Los casos de uso se realizan mediante módulos que se ejecutan en la arquitectura, estos módulos pueden ser utilizados en más de un caso de uso y cada módulo está contenido en un paquete.

Se construyeron algunos módulos especiales:

- **messenger:** Muestra mensajes del sistema. Por ejemplo, si llegara a ocurrir un error con la base de datos se utiliza para informar detalles. En el desarrollo es útil para depuración.

[7] Lucena López, Manuel C.; *Criptografía y Seguridad en Computadores*; Universidad de Jaén; 2ª Ed. 1999; p.36.

- **Formularios:** Son parte de la interfaz para el usuario. Permite al usuario introducir información al sistema. Están programados para reportar errores aunque la validación es independiente de estos módulos. Módulos que son formularios: *FrmDelete, FrmListADocument, FrmListDocuments, FrmLogin, FrmNewTopic, FrmRegisterUser, FrmUploadFile*.
- **Listados:** Con estos módulos el sistema muestra al usuario los datos contenidos de manera ordenada en tablas. Los hay de dos formas: los que muestran una vista global sobre una *colección*, por ejemplo, el listado de los consejeros; y los que muestran el detalle sobre un objeto de la *colección*, por ejemplo, la vista en detalle de un *consejero*. Módulos que son listados: *FrmListADocument, FrmListDocuments, ListATopic, ListAUser, ListTopics, ListUsers*.
- **Download:** descarga documentos al equipo del cliente, no devuelve ningún resultado al sistema después, verifica la seguridad porque un documento privado sólo puede ser descargado por un consejero.
- **Seguridad:** El sistema debe diferenciar a los usuarios permitiendo o no el acceder a otros módulos. Si algún módulo esta restringido este verifica los atributos del usuario y se ejecuta en caso de estar autorizado, si por el contrario el usuario no esta autorizado el sistema lo redirecciona a un lugar público. **Nota:** aún no esta implementado en todos los módulos.

2.5.2.5.1 Caso de uso Registrar Usuario

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *REGISTRAR USUARIO*

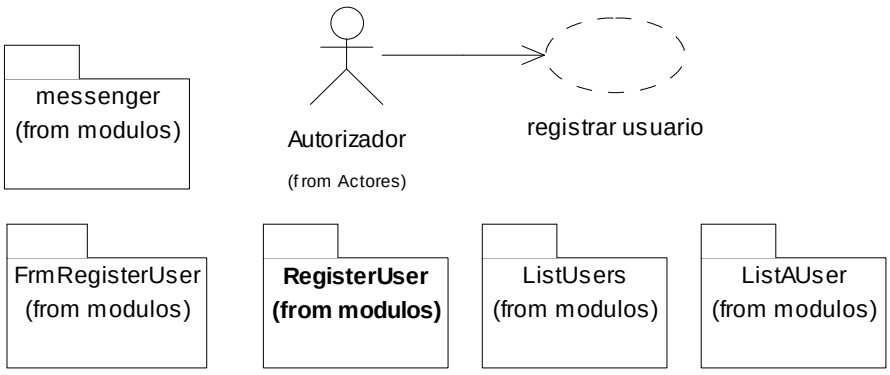

Fig. 2.5.2-9: Módulos del caso de uso registrar usuario

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **FrmRegisterUser:** módulo que muestra el formulario de captura de los datos del consejero, también muestra los errores en caso de una captura incorrecta.
- **RegisterUser:** módulo que realiza el caso de uso, es decir, registra el consejero en el sistema.
- **ListUsers:** módulo que lista los consejeros registrados en el sistema con algunos de sus datos.
- **ListAUser:** módulo que muestra los datos de un consejero en particular.

El módulo que realiza el caso de uso es *RegisterUser* mientras que los demás son módulos complementarios ya sea para capturar la información o para mostrar los resultados de la operación; por tanto sólo se detallará este módulo.

Descripción del módulo *RegisterUser* (Registrar Usuario)

El módulo *RegisterUser* (Registrar Usuario) se realiza mediante la siguiente secuencia:

SECUENCIA DEL MÓDULO *REGISTERUSER* (REGISTRAR USUARIO)

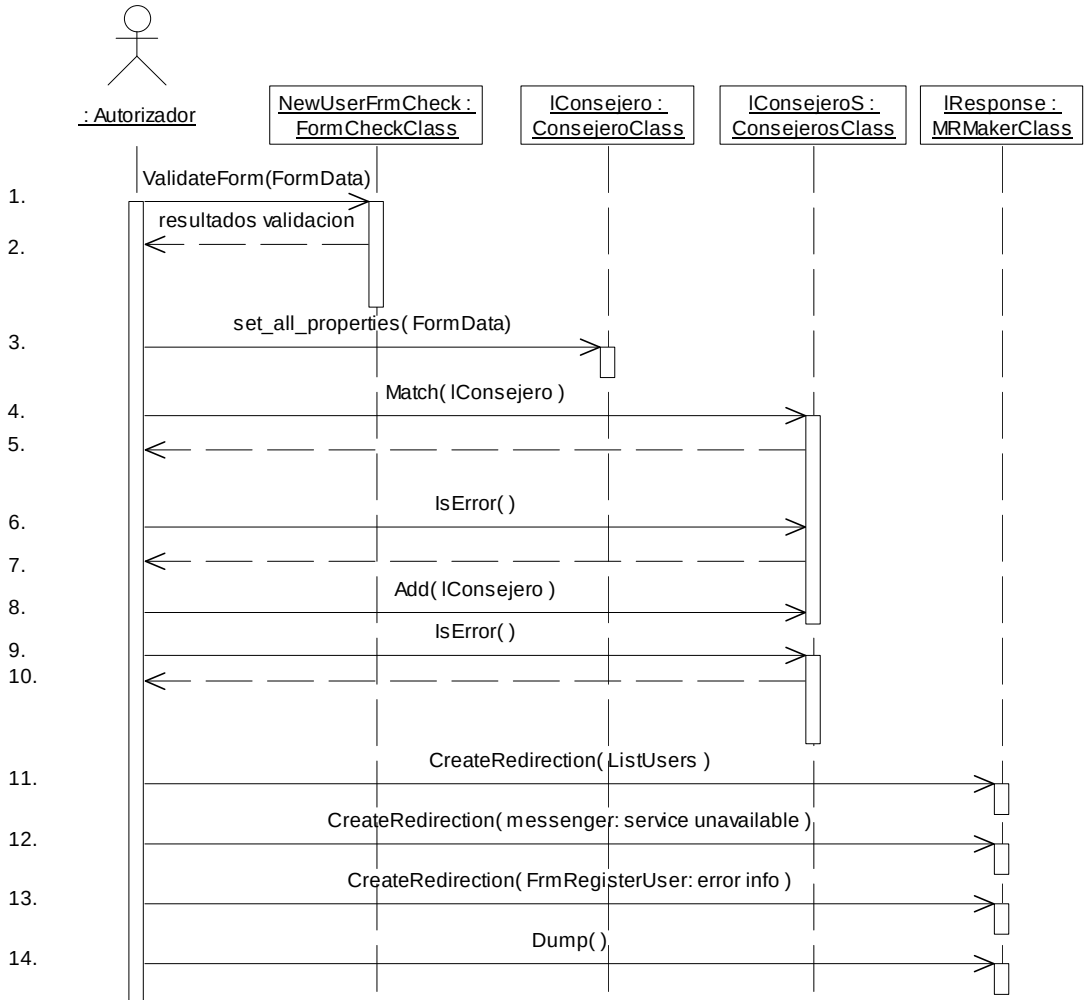

Fig. 2.5.2-10: Secuencia del módulo RegisterUser *(Registrar Usuario)*

Explicación de la secuencia:

1. Se validan los datos del formulario (*FormData*) mediante el objeto *NewUserFrmCheck*.
2. En caso de la captura sea incorrecta se genera un informe de qué datos estaban mal capturados y se va al paso 13.
3. Se crea el objeto *IConsejero* y se le asignan todas sus propiedades.
4. Se verifica que no exista otro consejero con sus mismas propiedades en la colección (se requiere que la propiedad *login* sea única).
5. En caso de que sea un consejero ya existente (propiedad *login*) se procede de manera análoga al paso 2.
6. Se verifican posibles errores de base de datos.

7. En caso de error de base de datos se va al paso 12.
8. Se agrega el consejero (*IConsejero*) a la colección.
9. Se verifican posibles errores en la base de datos.
10. En caso de error de base de datos se va al paso 12,
11. Se crea la respuesta tipo *redirección al módulo 'ListUsers'* con el objetivo de ver los resultados. Se va al paso 14.
12. Se crea la respuesta tipo *redirección al módulo 'messenger'* con el mensaje *servicio no disponible* o alguno similar con el objetivo de informar que hay problemas técnicos en el módulo. Se va al paso 14.
13. Se crea la respuesta tipo *redirección al módulo 'FrmRegisterUser'* con los parámetros de error del formulario con el objetivo de que se vuelva a capturar la información. Se va al paso 14.
14. El módulo envía su respuesta al resto del sistema.

2.5.2.5.2. Caso de uso Crear Tema

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *CREAR TEMA*

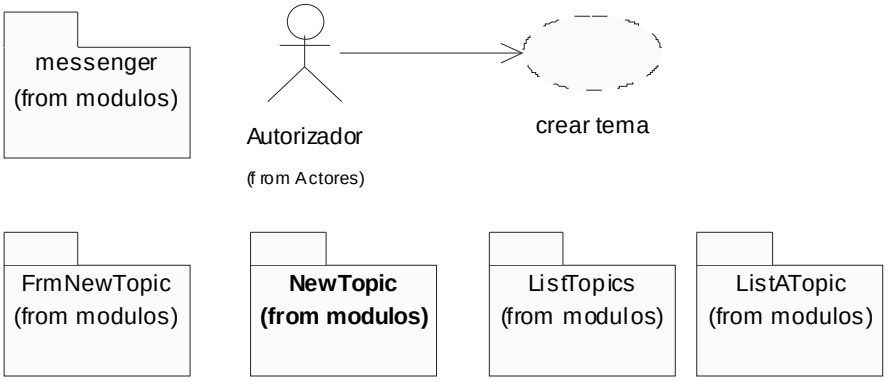

Fig. 2.5.2-11: Módulos del caso de uso crear tema

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **FrmNewTopic:** Módulo que permite capturar los datos relacionados con el tema nuevo.
- **ListTopics:** Módulo que lista los temas ya existentes en el sistema.
- **ListATopic:** Módulo que muestra los datos referentes a un tema.
- **NewTopic:** módulo que realiza el caso de uso, es decir, crea un tema nuevo.

Descripción del módulo *NewTopic* (Crear Tema)

El módulo *NewTopic* (Crear Tema) se realiza mediante la siguiente secuencia:

SECUENCIA DEL MÓDULO *NEWTOPIC* (CREAR TEMA)

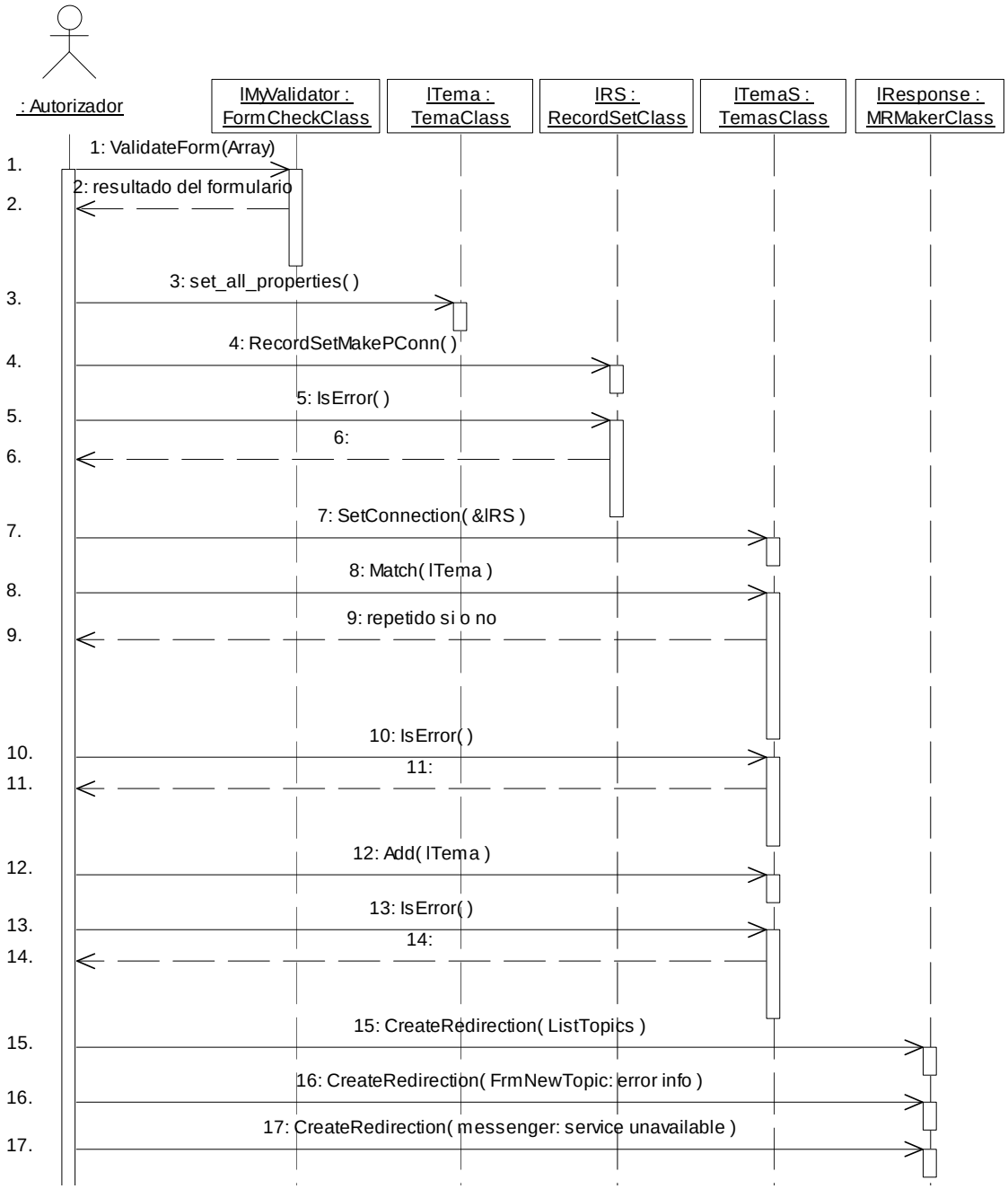

Fig. 2.5.2-12: Secuencia del módulo NewTopic *(Crear Tema)*

Explicación de la secuencia:

1. Se validan los datos del formulario (*FormData*) mediante el objeto *NewTopicFrmCheck*.

2. En caso de la captura sea incorrecta se genera un informe de qué datos estaban mal capturados y se va al paso 16
3. Se crea el objeto *ITema* y se le asignan todas sus propiedades.
4. Crea una conexión con la base de datos.
5. Verifica que no haya errores de base de datos.
6. Si hay error va al paso 17
7. Establece una relación entre la colección de temas y la conexión de base de datos.
8. Se verifica que no exista otro tema con sus mismas propiedades en la colección (se requiere que la propiedad *keycode* sea única).
9. En caso de que sea un tema ya existente (propiedad *keycode*) se procede de manera análoga al paso 2.
10. Se verifican posibles errores de base de datos.
11. En caso de error de base de datos se va al paso 17.
12. Se agrega el tema (*ITema*) a la colección.
13. Se verifican posibles errores en la base de datos.
14. En caso de error de base de datos se va al paso 17,
15. Se crea la respuesta tipo *redirección al módulo 'ListTopics'* con el objetivo de ver los resultados.
16. Se crea la respuesta tipo *redirección al módulo 'FrmRegisterUser'* con los parámetros de error del formulario con el objetivo de que se vuelva a capturar la información
17. Se crea la respuesta tipo *redirección al módulo 'messenger'* con el mensaje *servicio no disponible* o alguno similar con el objetivo de informar que hay problemas técnicos en el módulo.

Finalmente el módulo envía su respuesta al resto del sistema.

2.5.2.5.3. Caso de uso Iniciar sesión

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *INICIAR SESIÓN*

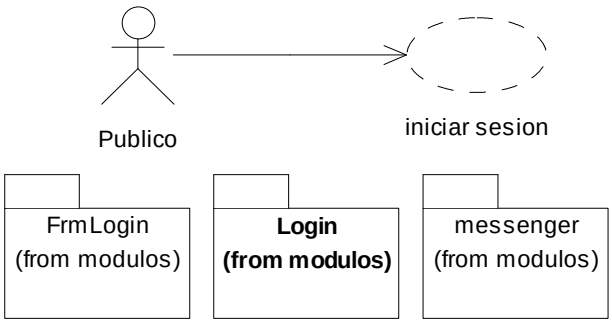

Fig. 2.5.2-13: Módulos del caso de uso iniciar sesión

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **FrmLogin:** módulo que muestra el formulario de inicio de sesión, también muestra los errores en caso de que los datos de sesión sean incorrectos.
- **Login:** módulo que realiza el caso de uso, es decir, inicia la sesión de usuario.

Descripción del módulo *Login* (Iniciar Sesión)

El módulo *Login* (Iniciar Sesión) se realiza mediante la siguiente secuencia:

SECUENCIA DEL MÓDULO *LOGIN* (INICIAR SESIÓN)

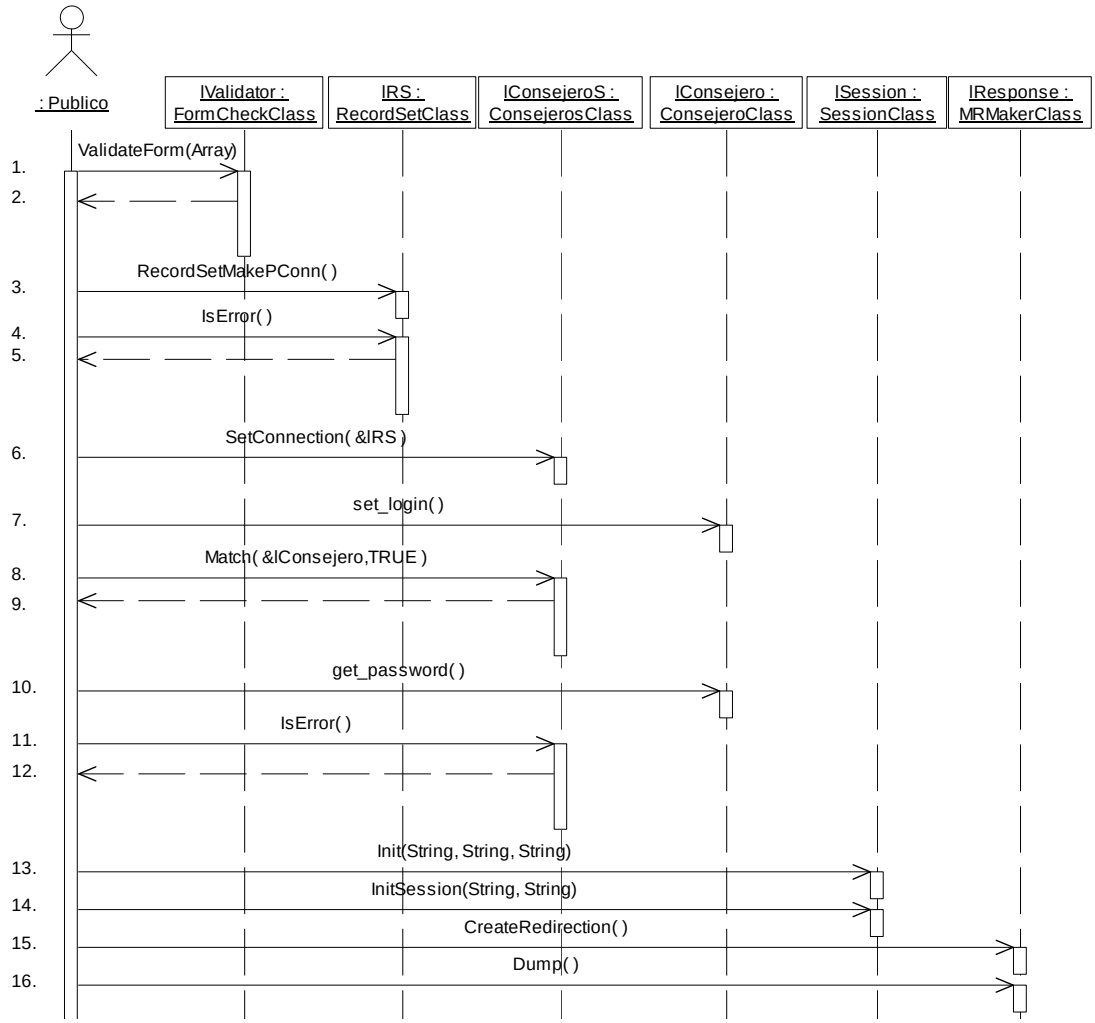

Fig. 14: Secuencia del módulo Login *(Iniciar Sesión)*

Explicación de la secuencia:

1. Se validan los datos del formulario (*Array* con los campos *login* y *password*).
2. Si los datos son incorrectos se genera un informe de error y se va al paso 15 con el estado '*frm error'.*
3. Se crea la conexión a la base de datos (*IRS*).
4. Se verifica el error de la base de datos.
5. Si hay error en la base de datos se genera el mensaje de error y se va al paso 15 con el estado *'db error'*.
6. Se asigna la conexión de base de datos (*IRS*) a la colección de consejeros (*IConsejeroS*).
7. Se crea un consejero (*IConsejero*) al que se le asigna la propiedad *login*.

8. Se busca en la colección un consejero con la misma propiedad *login* que *IConsejero* con la opción de completar el resto de las propiedades de *IConsejero* (parámetro *TRUE*)
9. Se obtiene el resultado si se encontró o no. Si no se encontró (salvo error de base de datos) se procede de manera análoga al paso 2.
10. Si se encontró se obtiene el *password*; si no coinciden el *password* del formulario y el *password* del objeto en la colección se procede de forma análoga al paso 2.
11. Se verifica si hubo error de base de datos.
12. Si hubo error de base de datos se procede de forma análoga al paso 5.
13. Se crea el objeto *ISession* para manejo de sesiones y se configura (*Init*).
14. Se inicia la sesión mediante *InitSession(login,password)*.
15. En caso de que todo funcione correctamente se crea la redirección al tema donde se encontraba anteriormente (al inicio de sesión). En caso de error de formulario (estado *'frm error'*) se crea la redirección al módulo FrmLogin informando que hubo error en los datos de usuario. En caso de error de base de datos (estado *'db error'*) se crea una redirección al módulo *messenger* informando que el servicio no está disponible debido a problemas técnicos.
16. El módulo devuelve el resultado al resto del sistema.

2.5.2.5.4. Caso de uso Cerrar sesión

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *CERRAR SESIÓN*

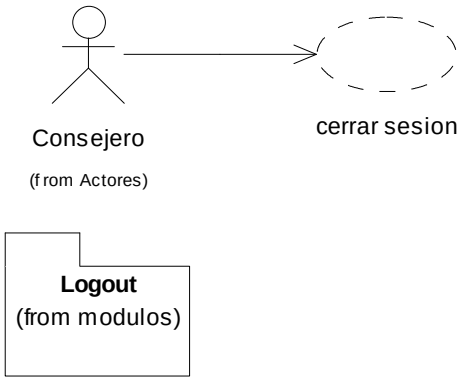

Fig. 2.5.2-15: Módulos del caso de uso cerrar sesión

Descripción de los módulos:

- **Logout:** módulo que realiza el caso de uso, es decir, termina la sesión de usuario.

Descripción del módulo *Logout* (Cerrar Sesión)

El módulo *Logout* (Cerrar Sesión) se realiza mediante la siguiente secuencia:

SECUENCIA DEL MÓDULO *LOGOUT* (CERRAR SESIÓN)

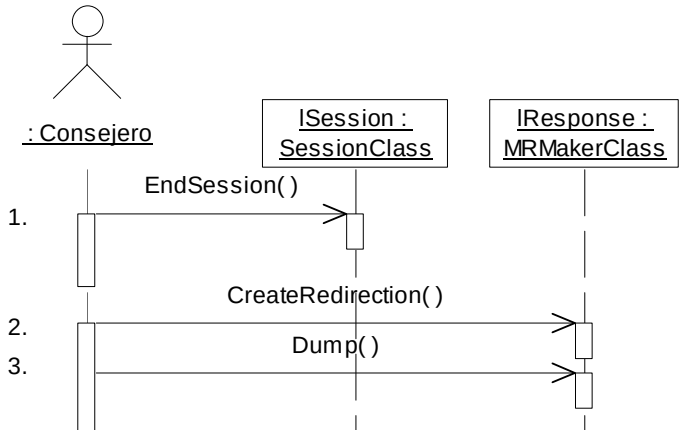

Fig. 2.5.2-16: Secuencia del módulo Logout *(Cerrar Sesión)*

Explicación de la secuencia:

1. Se eliminan las variables de sesión (se termina la sesión).
2. Se crea la redirección al tema donde se encontraba el usuario antes de cerrar la sesión.
3. El módulo regresa la información al resto del sistema.

2.5.2.5.5. Caso de uso Publicar

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *PUBLICAR*

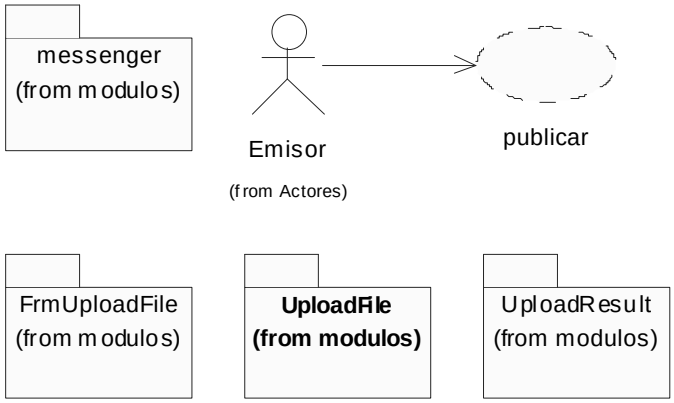

Fig. 2.5.2-17: Módulos del caso de uso publicar

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **FrmUploadFile:** módulo que muestra el formulario de captura de los datos del documento a publicar, también muestra los errores en caso de una captura incorrecta.

- **Uploadfile:** módulo que realiza el caso de uso, es decir, registra el documento a publicar en el sistema y coloca el archivo en el servidor.
- **UploadResult:** módulo que muestra el resultado de la operación.

Descripción del módulo *UploadFile* (Publicar)

Debido a su complejidad, el módulo *UploadFile* (Publicar) se describe mediante el siguiente diagrama de actividad:

DIAGRAMA DE ACTIVIDAD DEL MÓDULO *UPLOADFILE* (PUBLICAR)

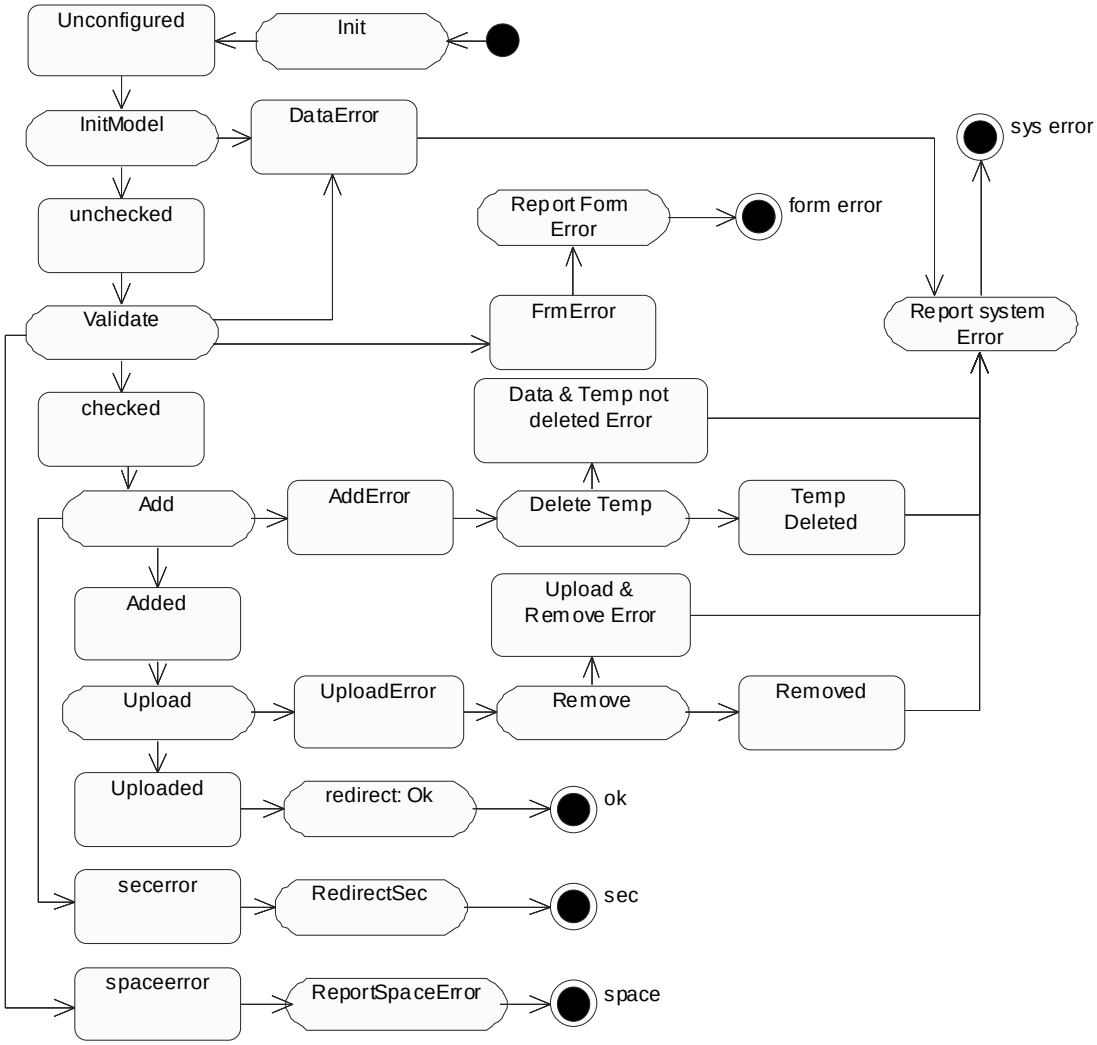

Fig. 2.5.2-18: Diagrama de actividad del comportamiento general del módulo UploadFile *(Publicar)*

Explicación del diagrama:

- **Estados:**
  - **Unconfigured:** indica que no han sido configurados los objetos que realizan las tarea del módulo.
  - **unchecked:** aun no se han validado las condiciones para la publicación.
  - **checked:** se han validado las condiciones para la publicación: Espacio en el servidor, el formulario correctamente llenado y el archivo válido.
  - **Added:** indica que se ha registrado el documento al sistema.

- **Uploaded:** indica que se ha transmitido el archivo al sistema.
- **DataError:** indica que hay un error con la base de datos.
- **FrmError:** indica que hubo un error en el llenado del formulario.
- **spaceerror:** espacio insuficiente en el sistema, se requiere *rotación automática*.
- **secerror:** El usuario no esta autorizado para usar este modulo.
- **AddError:** indica que no se pudo registrar el documento en el sistema.
- **TempDeleted:** Indica que se ha borrado el archivo temporal que se pretendía publicar.
- **Data & Temp not deleted Error:** indica que el documento temporal no pudo ser eliminado.
- **UploadError:** Error al procesar el archivo en el servidor.
- **Removed:** indica que se ha eliminado el registro del documento en el sistema.
- **Upload & Remove error:** No se pudo eliminar el registro del documento en el sistema.

- **Actividades:**
  - **Init:** crea los objetos que se usarán en el módulo.
  - **InitModel:** configura las colecciones.
  - **Validate:** valida el formulario, el archivo y el espacio en el servidor.
  - **Add:** Registra el documento en el sistema.
  - **Upload:** Mueve el archivo a la carpeta donde será almacenado.
  - **RedirectOk:** Genera una redirección al módulo *UploadResult* y se utiliza en caso de que la ejecución del módulo haya sido exitosa.
  - **RedirectSec:** Genera una redirección a la *URL Base* y se utiliza cuando el usuario no está autorizado a usar este módulo.
  - **ReportSpaceError:** Genera una redirección al módulo *messenger* informando de la carencia de espacio en el servidor.
  - **ReportFormError:** Genera una redirección al módulo *FrmUploadFile* configurándolo para que muestre los errores de formulario.
  - **DeleteTemp:** Elimina el archivo temporal.
  - **Remove:** Elimina el registro del documento del sistema.
  - **ReportSystemError:** Genera una redirección al módulo *messenger* y se utiliza cuando hay error de sistema.

2.5.2.5.6. Caso de uso Autorizar Documento

Definimos *estado del documento* como aquel que está caracterizado por las siguientes propiedades: *privado, permanente, aprobado*; las cuales pueden tomar de forma independiente el valor de verdadero o falso.

El proceso de *autorizar documento* es la actividad mediante la cual, el *autorizador* cambia el valor de dichas propiedades y así modifica el estado de un documento.

También puede *eliminar* el documento del sistema, tal eliminación es independiente de la propiedad *permanente*, la cual sólo aplica a la *eliminación automática* que se realiza en la *rotación automática*.

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *AUTORIZAR DOCUMENTO*

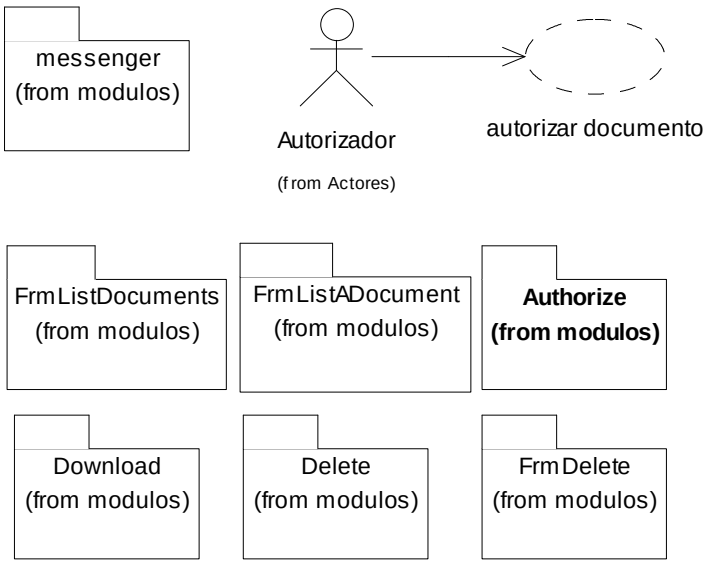

Fig. 2.5.2-19: Módulos del caso de uso autorizar documento

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **FrmListDocuments:** módulo que lista los documentos del más reciente al más antiguo indicando el estado del documento y permitiendo su descarga o eliminación así como pasar a una vista detallada (*FrmListADocument*).
- **FrmListADocument:** módulo que muestra un documento con toda su información y permite modificar su estado: público/privado, permanente/no permantente, aprobado/pendiente; también permite descargarlo o eliminarlo.
- **Authorize:** módulo que realiza el caso de uso de '*autorizar documento*' (cambiar el estado de un documento) y realiza (en caso de ser necesario) la *rotación automática*.
- **Download:** módulo que descarga el documento en el equipo del cliente.
- **Delete:** módulo que elimina el documento del sistema.
- **FrmDelete:** módulo que muestra un formulario para confirmar si se desea eliminar un documento del sistema.

Descripción del módulo *Authorize* (Autorizar Documento)

Debido a su complejidad, el módulo *Authorize* (Autorizar Documento) se describe mediante el siguiente diagrama de actividad:

DIAGRAMA DE ACTIVIDAD DEL MÓDULO *AUTHORIZE* (AUTORIZAR DOCUMENTO)

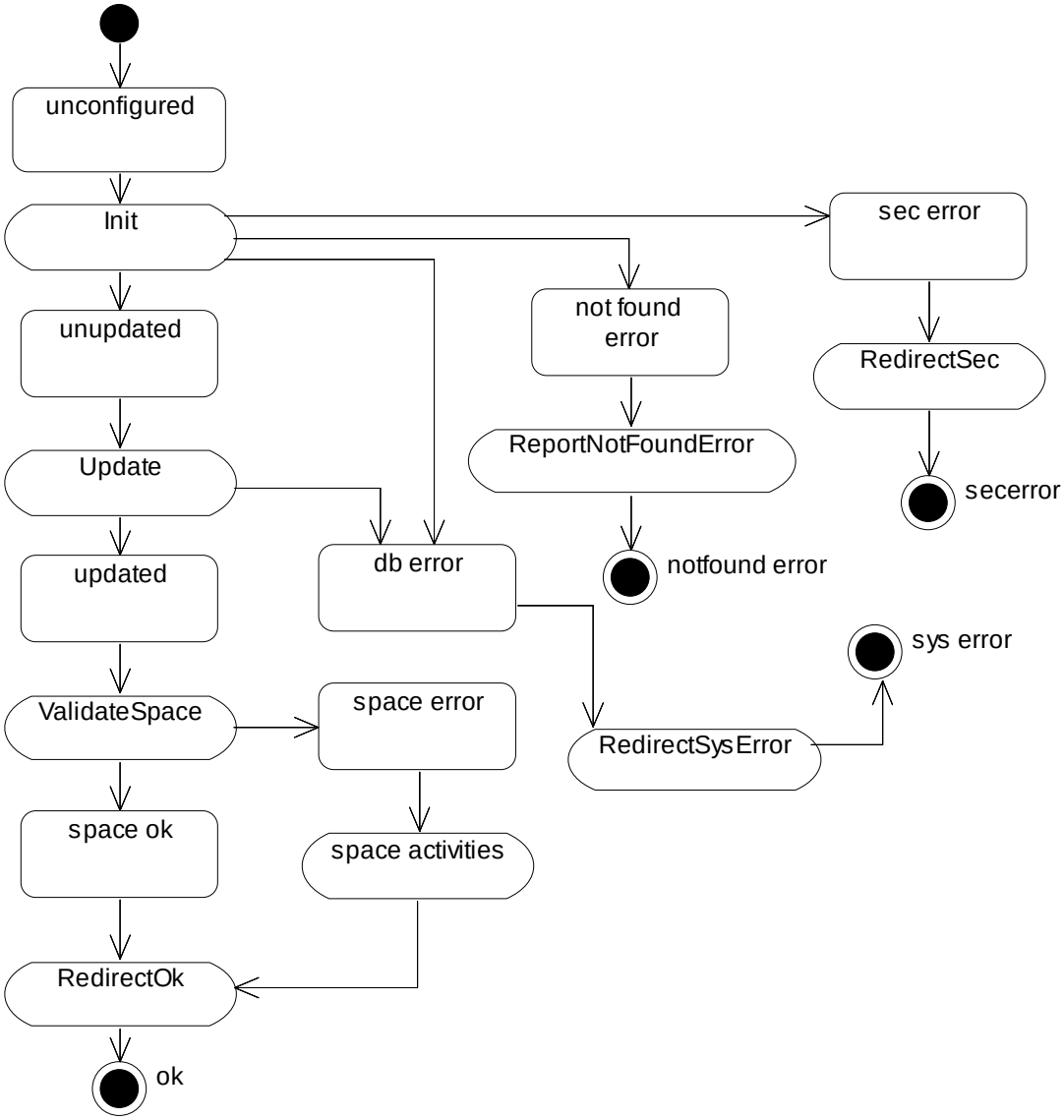

Fig. 2.5.2-20: Diagrama de actividad del módulo Authorize *(Autorizar Documento)*

Explicación del diagrama:

- **Estados:**
  - **unconfigured:** indica que no han sido configurados los objetos que realizan las tarea del módulo.
  - **unupdated:** indica que el documento no ha sido actualizado al nuevo estado.
  - **updated:** indica que el documento ya fue actualizado al nuevo estado.
  - **space ok:** indica que el espacio en el servidor es suficiente y no es necesaria ninguna acción de *rotación automática*.

- **sec error:** error de seguridad: usuario no autorizado para este módulo.
- **not found error:** error de documento no encontrado.
- **db error:** error de base de datos.
- **space error:** espacio insuficiente en el sistema, se requiere *rotación automática*.

- **Actividades:**
  - **Init:** crea y configura los objetos que se usarán en el módulo.
  - **Update:** actualiza el estado del documento en el sistema.
  - **ValidateSpace:** valida el espacio disponible en el servidor.
  - **RedirectOk:** Genera una redirección al módulo *FrmListDocuments* y se utiliza en caso de que la ejecución del módulo haya sido exitosa.
  - **RedirectSec:** Genera una redirección a la *URL Base* y se utiliza cuando el usuario no está autorizado a usar este módulo.
  - **ReportNotFoundError:** Genera un mensaje de documento no encontrado y lo muestra al usuario.
  - **RedirectSysError:** genera un mensaje de error de sistema y lo envía como parámetro a *messenger*, hace la redirección al módulo *messenger* con ese parámetro.
  - **space activities:** actividades que realizan la *rotación automática*.

Las actividades y los estados de este módulo se realizan mediante la clase *AuthorizerClass*.

2.5.2.5.7. Caso de uso Navegar

Muestra los documentos que hay en el tema y que han sido aprobados, aplica filtros de seguridad de tal forma que si el usuario es público solo se muestran los documentos públicos y si es consejero pueda ver los documentos privados.

Este caso de uso está formado por los módulos que se muestran en el diagrama:

MÓDULOS DEL CASO DE USO *NAVEGAR*

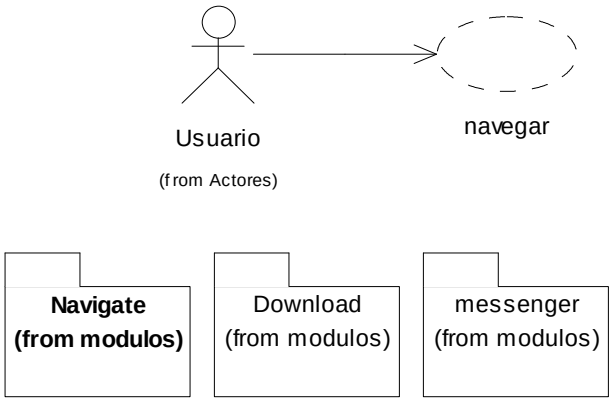

Fig. 2.5.2-21: Módulos del caso de uso Navegar

Descripción de los módulos:

- **messenger:** módulo que muestra mensajes del sistema, usado para informar cuando ocurrió un error con la base de datos.
- **Download:** módulo descarga un documento en el equipo del cliente considerando la seguridad.

- **Navigate:** módulo que muestra los documentos disponibles en un determinado tema colocando el más reciente hasta arriba y mostrando la información relativa a cada documento.

Descripción del módulo *Navigate* (Navegar)

El módulo *Navigate* (Navegar) queda descrito mediante la secuencia siguiente, la cual por motivo de su extensión queda dividida en dos partes:

SECUENCIA DEL MÓDULO *NAVIGATE* (NAVEGAR - I)

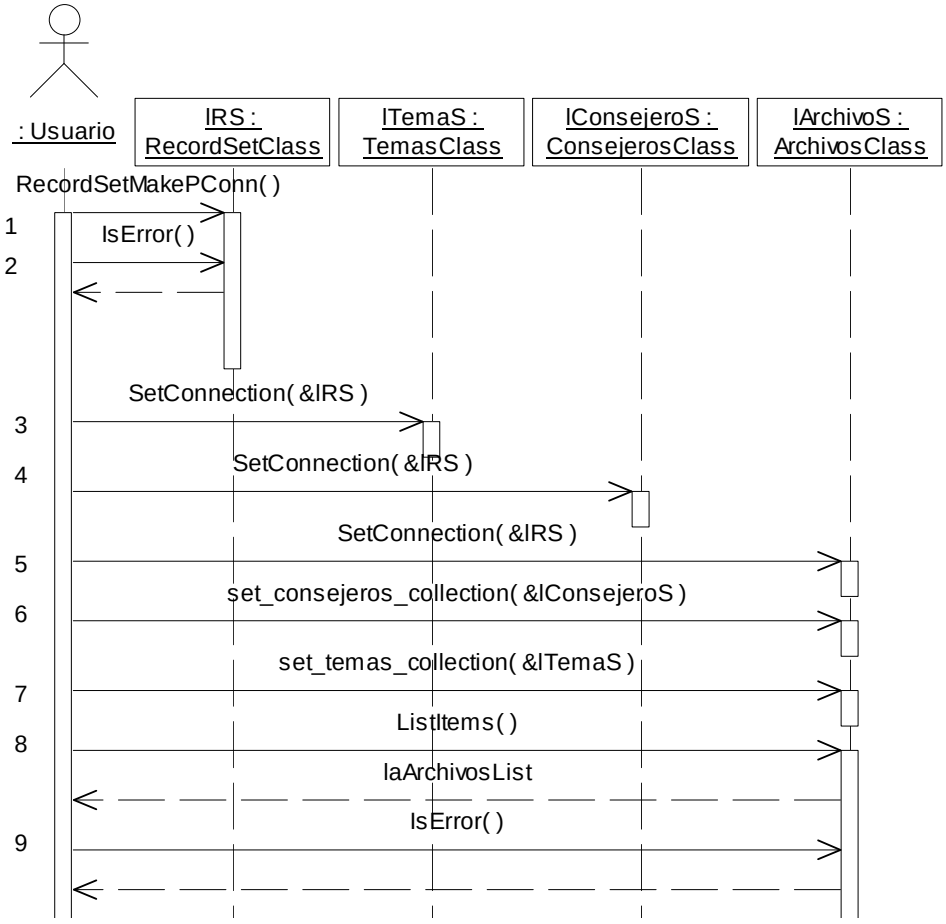

Fig. 2.5.2-22: Secuencia del módulo Navigate *(Navegar - primera parte)*

Explicación de la secuencia (I):

1. Se establece la conexión de la base de datos.
2. Verifica si hay error.
3. Asigna la conexión de la base de datos a la colección de temas.
4. Asigna la conexión de la base de datos a la colección de consejeros.
5. Asigna la conexión de la base de datos a la colección de archivos.
6. Crea una relación entre la colección de consejeros y la colección de archivos.
7. Crea una relación entre la colección de temas y la colección de archivos.
8. Obtiene la lista de los archivos que se deberán mostrar, tomando en cuenta las opciones de privacidad y el tema en que se encuentra el usuario.
9. Verifica si hay error.

SECUENCIA DEL MÓDULO *NAVIGATE* (NAVEGAR - II)

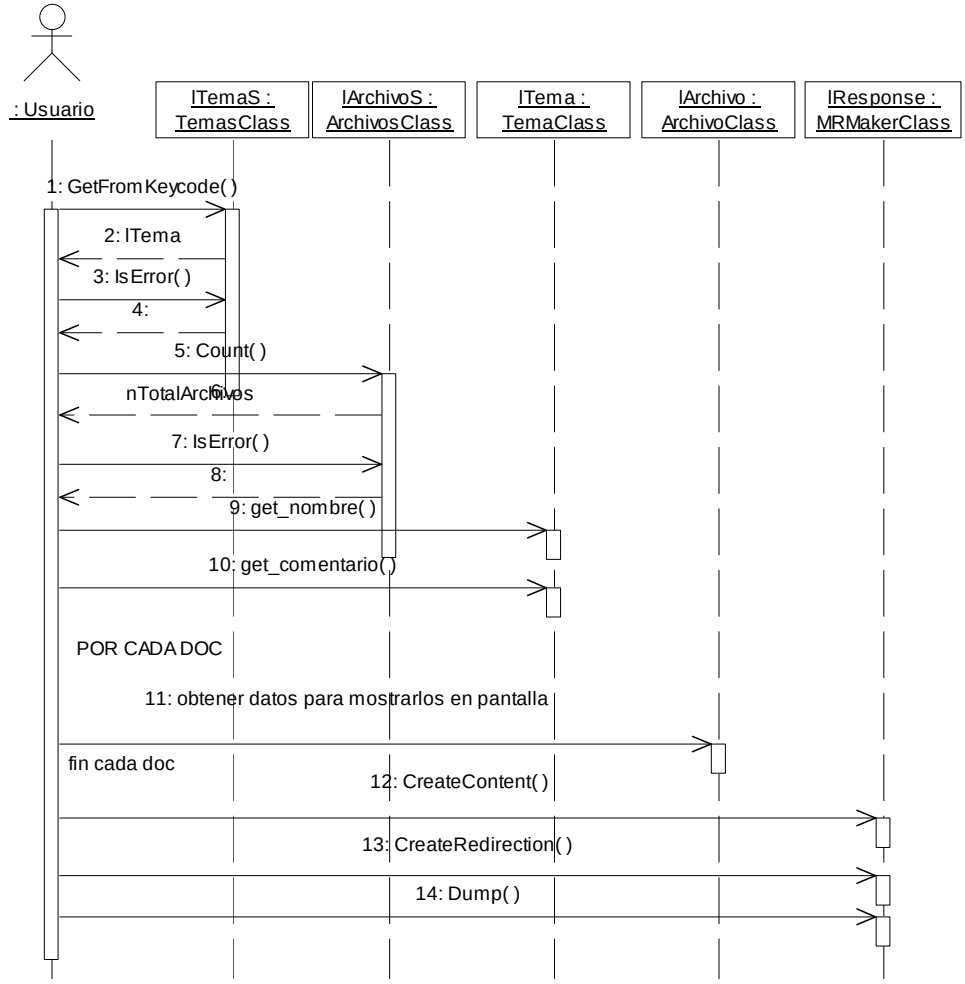

Fig. 2.5.2-23: Secuencia del módulo Navigate *(Navegar - segunda parte)*

Explicación de la secuencia (II):

1. Se solicita el tema.
2. Regresa el tema
3. Verifica si hay error.
4. Devuelve el estado de error.
5. Solicita el número de documentos en el tema
6. Devuelve el número de documentos en el tema
7. Verifica el estado de error
8. Devuelve el estado de error.
9. Solicita el nombre del tema
10. Solicita el comentario sobre el tema
11. Solicita datos sobre el documento
12. Genera los datos para visualización
13. Genera la URL para enviar los resultados
14. Envía los resultados a otro módulo para continuar con el proceso.

2.5.3. Segunda Iteración

En la primera iteración se detectaron riesgos a lo largo del proceso. La segunda iteración consistió en la solución de dichos riesgos

Se solucionaron los riesgos uno a uno, tomando como prioritarios los que tenían un impacto y probabilidad mayores. Al resolver los riesgos se mejoro el sistema. En esta iteración se lograron resolver todos los riesgos detectados.

La probabilidad aparece en cero debido a que ya están resueltos lo cual los hace improbables.

TABLA 2.5.3-1: Lista de Riesgos de la fase de construcción al principio de la segunda iteración

Clave	Nombre	Precisión	Impacto	Probabilidad	Causa	Consecuencia	Estado	Fecha
4	Conexiones y desconexiones de base de datos	5	0.9	0.6	Las conexiones de la arquitectura principal y de los módulos son independientes	Conexiones inutilizadas abiertas consumiendo recursos en el servidor	Pendiente	07/02/2005
19	Redundancia en la inclusión del archivo config.php	5	0.6	1	Se decidió así para probar los componentes mientras se hacía la línea base de la arquitectura	Dificultad en la portabilidad del sitio a otros servidores	Pendiente	07/02/2005
5	Cierre de sesión mientras se sube un archivo	5	0.2	0.2	Tiempo de sesión expirado	Confusión del usuario, operación cancelada	Pendiente	04/01/2005
10	El sistema es vulnerable a ataques a la autenticidad	5	0.3	0.3	Los módulos señalados utilizan mensajes HTML transmitidos mediante la URL	Ataques a la autenticidad del sitio haciendo que muestre contenidos falsos de todo tipo	Pendiente	08/02/2005
15	Alto acoplamiento en el nombre a usar en el tema raíz	5	0.5	1	La clase QueryManagerClass genera el nombre internamente	Mala imagen pues el nombre no se ajusta a lo que requiere el usuario	Pendiente	03/02/2005
3	Problema con los módulos por defecto (omisión)	5	0.4	0.5	El análisis no contemplo los casos	El sistema no funciona con temas vacíos/sin temas	Pendiente	05/01/2005
6	Interrupción del modulo de descarga de documento	5	0.2	1	Ttl de un programa php = 30 segs (por default)	Archivo dañado	Pendiente	08/02/2005
8	No se gestiona seguridad en los módulos	5	0.9	0.7	Simplicidad en la agenda	Sistema inseguro	Pendiente	07/02/2005

Clave	Nombre	Precisión	Impacto	Probabilidad	Causa	Consecuencia	Estado	Fecha
12	Discordancia en el orden de los documentos entre FrmListDocuments y FrmListADocument	5	0.2	0.8	Mostrarlos por fecha descendiente	Confusión en el usuario que lo lleva a cometer errores	Pendiente	10/01/2005
9	Falta una barra de profundidad de temas	5	0.4	1	No se ha contemplado en diseño	El usuario se confunde al no saber en que parte del sitio esta	Pendiente	05/01/2005

Descripción detallada de los riesgos al final de la segunda iteración.

Clave: 4
Nombre: Conexiones y desconexiones de base de datos
Estado: Resuelto
Precisión: 5 Impacto: 0.9 Probabilidad: 0
Fecha de registro del riesgo: 09/11/2004 19:20
Descripción:

- La arquitectura tiene su conexión a base de datos que desconecta y reconecta, y los módulos implementan su propia conexión. Además no todos los módulos se desconectan de la base de datos.

Causas:

- Las conexiones de la arquitectura principal y de los módulos son independientes.

Consecuencias:

- Quedan conexiones abiertas inutilizadas consumiendo recursos en el servidor.

Acciones a considerar:

- [20050207] Se sugiere pasar en los argumentos el objeto de conexión a la base de datos, para así realizar las conexiones y desconexiones de manera única en cada ejecución y por fuera de los módulos.

Contingencia:

- Solicitar asesoría para la optimización de los recursos.

Categoría:

- arquitectura, análisis y diseño

Cambios:

- 09/02/2005 08:00:00 p.m. Se realizaron las acciones y se resolvió.
- Probabilidad de 0.6->0.0

Clave: 19
Nombre: Redundancia en la inclusión del archivo config.php
Estado: Resuelto
Precisión: 5Impacto: 0.6 Probabilidad: 0

Fecha de registro del riesgo: 07/02/2005 18:59

Descripción:

- En todos los scripts se incluye el archivo config.php (de hecho una copia en la carpeta del script)

Causas:

- Se decidió así para probar los componentes mientras se hacía la línea base de la arquitectura

Consecuencias:

- Dificultad en la portabilidad del sitio a otros servidores

Acciones a considerar:

- [20050207] Se sugiere eliminar la inclusión de ese archivo de todas las carpetas excepto el script principal, y en cambio usar 'global' para declarar globales las variables que se requieran de este script en cada caso.

Contingencia:

- Desarrollar un script que actualice todas las copias.

Categoría:

- Arquitectura, análisis y diseño.

Cambios:

- 09-Feb-05 18:59 Se realizaron las acciones y las pruebas fueron satisfactorias.
- Probabilidad de 1->0.0

Clave: 5
Nombre: Cierre de sesión mientras se sube un archivo
Estado: Resuelto
Precisión: 5 Impacto: 0.2 Probabilidad: 0
Fecha de registro del riesgo: 04/01/2005 11:55
Descripción:

- Si un archivo muy grande tarda mucho en enviarse al servidor existe la posibilidad de que se cierre la sesión del usuario por tiempo.

Causas:

- El tiempo establecido para expirar la sesión es limitado.

Consecuencias:

- El usuario puede confundirse, la operación será cancelada sin que el usuario pueda hacer nada para subir su archivo.

Acciones a considerar:

- [20050208] sugiere colocar la instrucción set_time_limit(0) que cambia el tiempo limite a infinito.

Contingencia:

- Buscar asesoría para permitir la operación al usuario.

Categoría:

- Arquitectura, análisis y diseño

Cambios:

- 10/02/2005 05:15:00 p.m.Se realizaron las acciones y las pruebas fueron satisfactorias.
- Probabilidad 0.2 ->0

Clave: 10
Nombre: El sistema es vulnerable a ataques a la autenticidad
Estado: Resuelto
Precisión: 5 Impacto: 0.3 Probabilidad: 0
Fecha de registro del riesgo: 05/01/2005 12:47
Descripción:

- El modulo messenger y los formularios que reportan errores de llenado de datos son vulnerables a un ataque a la autenticidad de la información modificando la URL para introducir contenidos HTML maliciosos, falsos, difamatorios, comerciales o de cualquier otro tipo inauténtico.

Causas:

- Los módulos señalados utilizan mensajes HTML transmitidos mediante la URL

Consecuencias:

- Ataques a la autenticidad del sitio haciendo que muestre contenidos falsos de todo tipo

Acciones a considerar:

- [20050208] Se sugiere codificar los mensajes tanto en messenger como en los formularios

Contingencia:

- Buscar asesoría para solucionar el riesgo.

Categoría:

- arquitectura, análisis y diseño

Cambios:

- 08/02/2005 19:33 Se realizaron las acciones y el resultado fue satisfactorio.
- Probabilidad 0.3 ->0

Clave: 15
Nombre: Alto acoplamiento en el nombre a usar en el tema raíz
Estado: Resuelto
Precisión: 5 Impacto: 0.5 Probabilidad: 0
Fecha de registro del riesgo: 01/02/2005 09:44
Descripción:

- El tema raíz tiene un nombre por defecto que esta altamente acoplado con la clase que gestiona los menús y no es configurable

Causas:

- La clase QueryManagerClass genera el nombre internamente

Consecuencias:

- Acciones a considerar: "[20050201]: se propone agregar un parámetro <RootName> en ContentManagerClass.xml que indique el nombre del tema raíz y pasarlo como parámetro a QueryManagerClass (que lo tendrá como propiedad)

Contingencia:

- Permitir que el sistema sea altamente acoplado

Categoría: arquitectura, análisis y diseño

Cambios:

- 10/02/2005 05:09:00 p.m.se realizaron las acciones y resulto satisfactorio
- probabilidad.: 1.0->0.0"

Clave: 3
Nombre: problema con los módulos por defecto (omisión)
Estado: resuelto
Precisión: 5 Impacto: 0.4 Probabilidad: 0
Fecha de registro del riesgo: 09/11/2004 19:20
Descripción:

- No se ha especificado a nivel de implementación la forma de procesar módulos por defecto cuando no están en los temas, o no hay temas, digamos no está en el XML de configuración del subsistema ni está implementado

Causas:

- El análisis no contemplo los casos

Consecuencias:

- El sistema no funciona con temas vacíos/sin temas

Acciones a considerar:

- Modificado el archivo ContentManagerClass.xml para que incluya un modulo por omisión cuando se esta en algún tema y un modulo por omisión cuando se esta en el tema raíz (ningún tema).

Contingencia:

- Agregar en el manual de usuario las instrucciones para el caso que no haya contenido en el tema.

Categoría:

- arquitectura, análisis y diseño

Cambios:

- 05/01/2005 11:38:00 a.m. Se realiarón las acciones y se resolvió el riesgo.

Clave: 6
Nombre: Interrupción del modulo de descarga de documento
Estado: resuelto
Precisión: 5 Impacto: 0.2 Probabilidad: 0
Fecha de registro del riesgo: 04/01/2005 12:02
Descripción:

- Cuando se descarga un documento grande si se tarda más de 30 segundos se interrumpe la ejecución del php, entonces se corrompe el archivo,

Causas: Ttl de un programa php = 30 segs (por default)
Consecuencias: archivo dañado
Acciones a considerar:

- "[20050208]: se colocó la instrucción set_time_limit(0) en el método dlFile de la clase FileDownloadClass"

Contingencia:

- **Recurrir a ayuda para solucionar el reisgo.**

Categoría:

- **arquitectura, análisis y diseño**

Cambios:

- 08/02/2005 02:11:00 p.m. Se realizaron las acciones y se resolvió el riesgo
- Probabilidad 1 -> 0

Clave: 8
Nombre: No se gestiona seguridad en los módulos
Estado: Resuelto
Precisión: 5 Impacto: 0.9 Probabilidad: 0
Fecha de registro del riesgo: 04/01/2005 12:17
Descripción:

- En los módulos no se verifica que el usuario tenga los privilegios para utilizar el módulo en cuestión

Causas:

- Simplicidad en la agenda

Consecuencias:

- Sistema inseguro

Acciones a considerar:

- Se verificó que el usuario de la sesión fuera un consejero o un autorizador según el caso en los módulos en los que se requería gestionar la seguridad

Contingencia:

- Negociar con el cliente la función de los casos de uso.

Categoría:

- arquitectura, análisis y diseño

Cambios:

- 07/02/2005 18:36 Se realizaron las acciones y se resolvió el riesgo.

Clave: 12
Nombre: Discordancia en el orden de los documentos entre FrmListDocuments y FrmListADocument
Estado: Resuelto
Precisión: 5 Impacto: 0.2 Probabilidad: 0
Fecha de registro del riesgo: 10/01/2005 19:22
Descripción:

- Debido al orden por fecha descendiente hay una discordancia en el orden en que se muestran los archivos entre estos dos módulos: FrmListDocuments y FrmListADocument

Causas:

- Mostrarlos por fecha descendiente

Consecuencias:

- Confusión en el usuario que lo lleva a cometer errores

Acciones a considerar:

- Ahora la consulta se realiza por la clave, ya que la clave crece también en función del tiempo y es más sencilla de manejar.

Contingencia:

- Describir en el manual de usuario con detalle el orden de los documentos.

Categoría:

- arquitectura, análisis y diseño

Cambios:

- 10/01/2005 19:22 Se realizaron las acciones y se resolvió el problema.
- Probabilidad 08->0

Clave: 9
Nombre: Falta una barra de profundidad de temas
Estado: resuelto
Precisión: 5 Impacto: 0.4 Probabilidad: 0
Fecha de registro del riesgo: 05/01/2005 11:39
Descripción:

- Falta una barra o lista o algo que le indique al usuario en que parte del sitio se encuentra navegando

Causas:

- No se ha contemplado en diseño

Consecuencias:

- El usuario se confunde al no saber en que parte del sitio esta

Acciones a considerar:

- "[20050201]: se sugiere generar el parámetro breadcrump desde QueryManagerClass provisionalmente como una etiqueta (HTML) para lo cual se requiere que la clase QueryManagerClass almacene la URL Base mediante un parámetro

Contingencia:

- Asumir el riesgo y dejar el sistema sin la barra de profundidad.

Categoría:

- Requisitos

Cambios:

- 03/02/2005 re realizaron las acciones
  prob. 1.0->0.0"

2.6. FASE DE TRANSICIÓN

2.6.1. Introducción

Los objetivos básicos de esta fase son:[1]

1. Cumplir los requisitos establecidos en las fases anteriores hasta la *satisfacción* de todos los usuarios.
2. Gestionar todos los aspectos relativos a la operación en el entorno del usuario, incluyendo la corrección de los defectos remitidos por los usuarios o los encargados de las pruebas de aceptación.

Las pruebas de aceptación pueden ser realizadas por el cliente o por una organización contratada para este fin.

Por tanto, el proyecto, en esta fase, está centrado en recibir información de los usuarios para:[2]

1. Determinar si el sistema hace lo que demandan sus usuarios y el negocio.
2. Descubrir riesgos inesperados o no detectados en las fases anteriores.
3. Anotar problemas no resueltos.
4. Encontrar fallos.
5. Eliminar ambigüedades y lagunas en la documentación de usuario.
6. Centrarse en áreas en las que los usuarios muestren deficiencias y necesiten información o formación.

Con esta información se modifica el sistema o la documentación. Es importante distinguir aquello que es posible colocar en el ciclo de desarrollo actual del proyecto y aquello que debería ser parte de una versión nueva del proyecto.

La distinción está en función de qué tan profundo es el cambio que se debe de realizar ya que en esta fase de desarrollo los cambios en el sistema no son profundos sino correcciones de *pequeñas deficiencias*[3] que permitan adaptar el sistema al entorno del cliente.

La fase de transición no es *predictiva*,[4] esta característica es común a todo el Proceso Unificado, pero en el caso de la fase de transición se acentúa aún más, pues las acciones a realizarse dependen en gran medida de la respuesta del usuario frente al sistema, lo cual es altamente impredecible.

Aún así, existen determinadas actividades que sí son predictivas en esta fase, como la elaboración del manual de usuario y la instalación del sistema.

Para las actividades no predictivas debe existir un plan de gestión de todas ellas. Finalmente debe existir un criterio para determinar cuándo se termina la fase de transición; éste criterio es: *cuando el cliente queda "satisfecho"*.[5]

---

[1] Ivar Jacobson, Grady Booch, James Rumbaugh; *El Proceso Unificado de Desarrollo de Software*; Addison Wesley; 1ª ed. Española; Madrid 2000; p. 381.
[2] *Ibid.* p. 382
[3] *Ibid.*
[4] Véase: Craig Larman, Philippe Kruchten, Kurt Bittner; *How to Fail with the Rational Unified Process*; Ratonal Software 2001.
[5] Ivar Jacobson; *op cit.*; p. 390

2.6.2. El plan de la Fase de Transición de este Proyecto

Para controlar las diferentes actividades y lograr los objetivos de la fase de transición en este proyecto, se propone el siguiente plan.

Las actividades se agruparán en dos categorías:

1. Actividades relacionadas con el manual de usuario.
2. Actividades relacionadas con la instalación, prueba y correcciones del sistema.

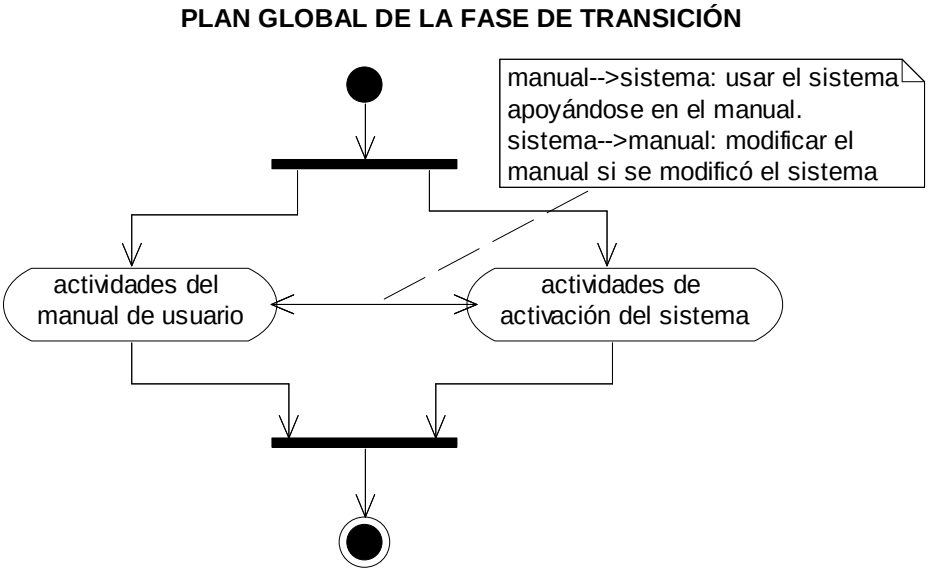

Fig. 2.6.2-1: Plan global de la fase de transición

Las actividades relacionadas con el manual de usuario, parten de la elaboración del manual de usuario y continúan por un proceso de pruebas del mismo manual para determinar si es lo suficientemente claro como para explicar el uso del sistema.

El manual de usuario se elabora en dos presentaciones: una impresa y una electrónica, la edición electrónica está disponible en el sitio Web para que cualquier usuario pueda consultarla.

Las actividades relacionadas con la instalación, prueba y correcciones del sistema comienzan con la instalación del sistema en el servidor donde se hospeda el sitio (*Web Hosting*),[6] posteriormente se realizan las pruebas necesarias para verificar tanto un correcto funcionamiento del sistema como una interfaz y presentación adecuada de la información para que el usuario pueda aprovechar fácilmente los servicios que ofrece el sistema.

Entre las correcciones o mejoras sugeridas por el usuario se deberá discriminar cuáles se realizarán en la presente versión del sistema (en esta fase) y cuáles se anotarán para futuras versiones del mismo.

Debe de existir una sincronización entre estas dos categorías o bloques de actividades, pues un cambio en el manual requiere el uso del sistema para evaluar el manual, y un cambio en el sistema podría requerir modificaciones al manual para que corresponda con el sistema.

El proceso se da por concluido cuando el cliente está satisfecho tanto con el manual como con el sistema.

[6] Véase el subcapítulo II.B. *Solución al problema de Web Hosting*.

2.6.2.1. Plan de documentación (actividades del manual de usuario)

El plan de documentación consta de las siguientes actividades:

1. **Elaboración del manual de usuario:** se elabora el manual tanto en su versión impresa como en su versión electrónica, buscando que cumpla el objetivo de orientar al usuario en el uso del sistema y resolver sus dudas.
2. **Evaluación y uso del manual:** el cliente evalúa el manual y lo utiliza, probando el sistema para verificar si el manual le orienta para usar el sistema adecuadamente o no, si es claro y fácil de usar o no.
3. **Modificación del manual:** si es necesario se realizan cambios en el manual, estos cambios pueden ser el resultado de la evaluación por parte del cliente o bien porque el sistema fue modificado y requiere modificarse el manual.

Las decisiones a tomar son:

1. ¿El cliente está satisfecho con el manual?
2. ¿El manual debe modificarse debido a cambios en el sistema?
3. ¿Terminaron las pruebas de uso del sistema del cliente?

El plan queda descrito en el siguiente diagrama de la figura *2.6.2-2.*

Las actividades del manual de usuario tienen un *punto de espera* que es cuando el cliente está satisfecho con el manual, no se requieren modificaciones por cambios en el sistema y no ha terminado la evaluación del sistema.

Entonces este bloque de actividades pasa a un estado de inactividad, en espera de que sea necesario modificar el manual o de que termine la evaluación de todo el sistema.

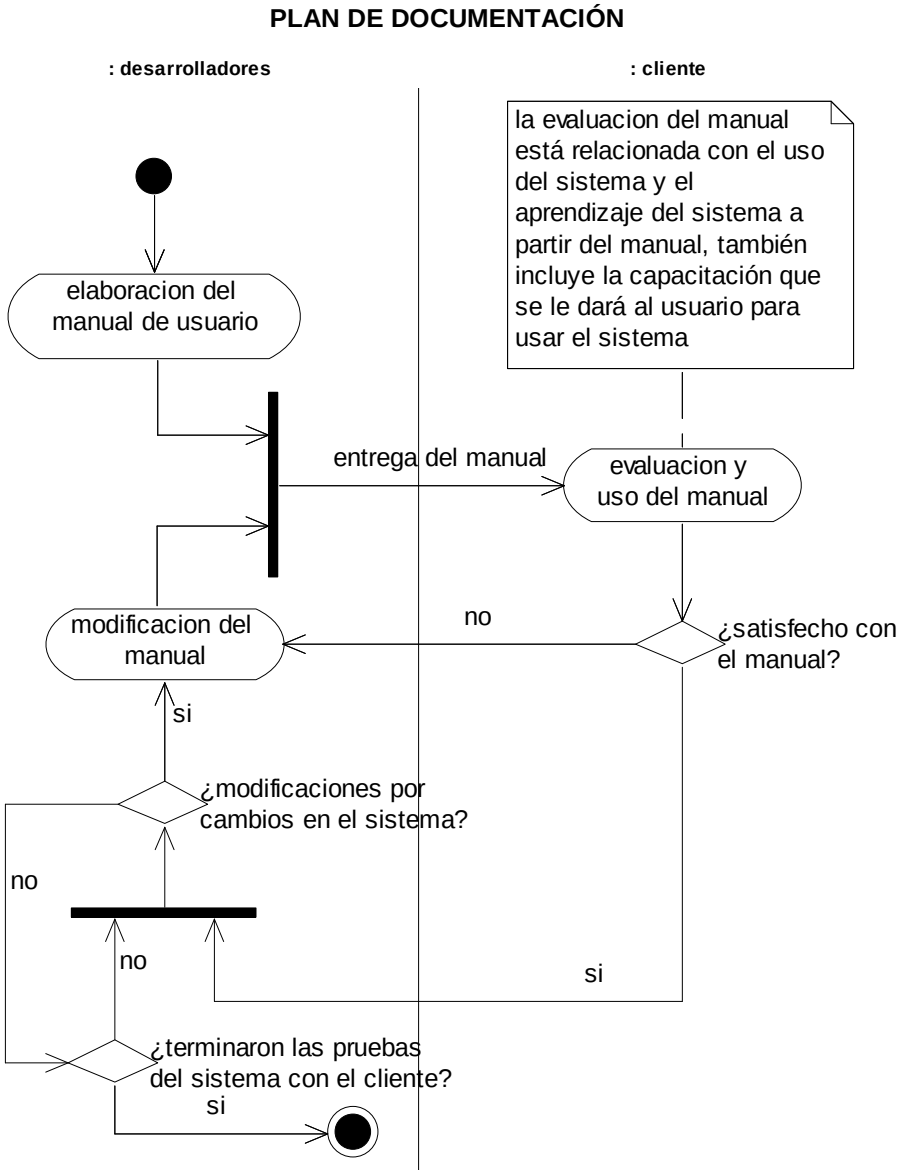

Fig. 2.6.2-2: Plan de documentación

2.6.2.2. Plan de instalación. Prueba y adaptación del sistema

Este bloque de actividades busca mejorar el sistema a partir de la experiencia del usuario con el mismo. Está formado por las siguientes actividades:

1. **Instalación y configuración del sistema:** el sistema se instala y se configura en el servidor y se realizan pruebas de estabilidad y funcionamiento básicas.
2. **Pruebas con los usuarios:** El cliente realiza pruebas del sistema, las pruebas consisten en usar el sistema de acuerdo con los diferentes actores del mismo y sus casos de uso. Las modificaciones requeridas pueden agruparse de la siguiente forma:

   - Fallos en el sistema.
   - Huecos de seguridad.
   - Modificaciones de la presentación de los datos (el usuario se confunde al usar el sistema).

- Riesgos de diverso tipo.
- Mejoras estéticas.
- Solicitud de nuevos servicios.
- Rendimiento del sistema.

3. **Modificación del sistema:** El sistema es modificado en alguna de las categorías mencionadas en el punto anterior.
4. **Modificación del manual:** En algunos casos la modificación del sistema puede provocar que se requiera modificar el manual. La modificación del manual se realiza de acuerdo al *Plan de Documentación* de la presente fase.
5. **Lista de mejoras para la siguiente versión:** Algunas modificaciones solicitadas por el usuario se realizarán en la presente fase, otras se anotarán para la realización de una nueva versión del sistema de acuerdo al presupuesto disponible.

Las decisiones a tomar en este plan son:

1. ¿El sistema es estable en el servidor después de la instalación? (prueba rápida)
2. ¿El cliente está satisfecho con el sistema?
3. ¿La modificación es por un fallo en el sistema o un hueco de seguridad?
4. ¿Conviene realizar la modificación solicitada en esta fase o en una nueva versión?

El plan se muestra en el siguiente diagrama de la figura *2.6.2-3.*

El criterio para determinar si una modificación solicitada se realizará en la presente fase o se anotará para la siguiente versión del sistema es:

1. Si se trata de un fallo en el sistema (el sistema no hace lo que debería) se corrige en esta fase.
2. Si se trata de un hueco de seguridad (se ha detectado una forma de violar los servicios de seguridad) y el servicio de seguridad que vulnera se le ofreció al usuario como parte del sistema, se corrige en esta fase de acuerdo a los recursos disponibles.
3. Si se trata de una modificación que no comprometa la arquitectura (sólo cambios menores) se realiza en esta fase.
4. Si es un nuevo servicio cuya realización implica poco desgaste se podrá realizar en esta fase dependiendo de la agenda y de que no requiera alterar la arquitectura.
5. En cuanto al rendimiento del sistema, como implica la revisión y (posible) rediseño de la arquitectura, se buscarán las optimizaciones que no comprometan el diseño.

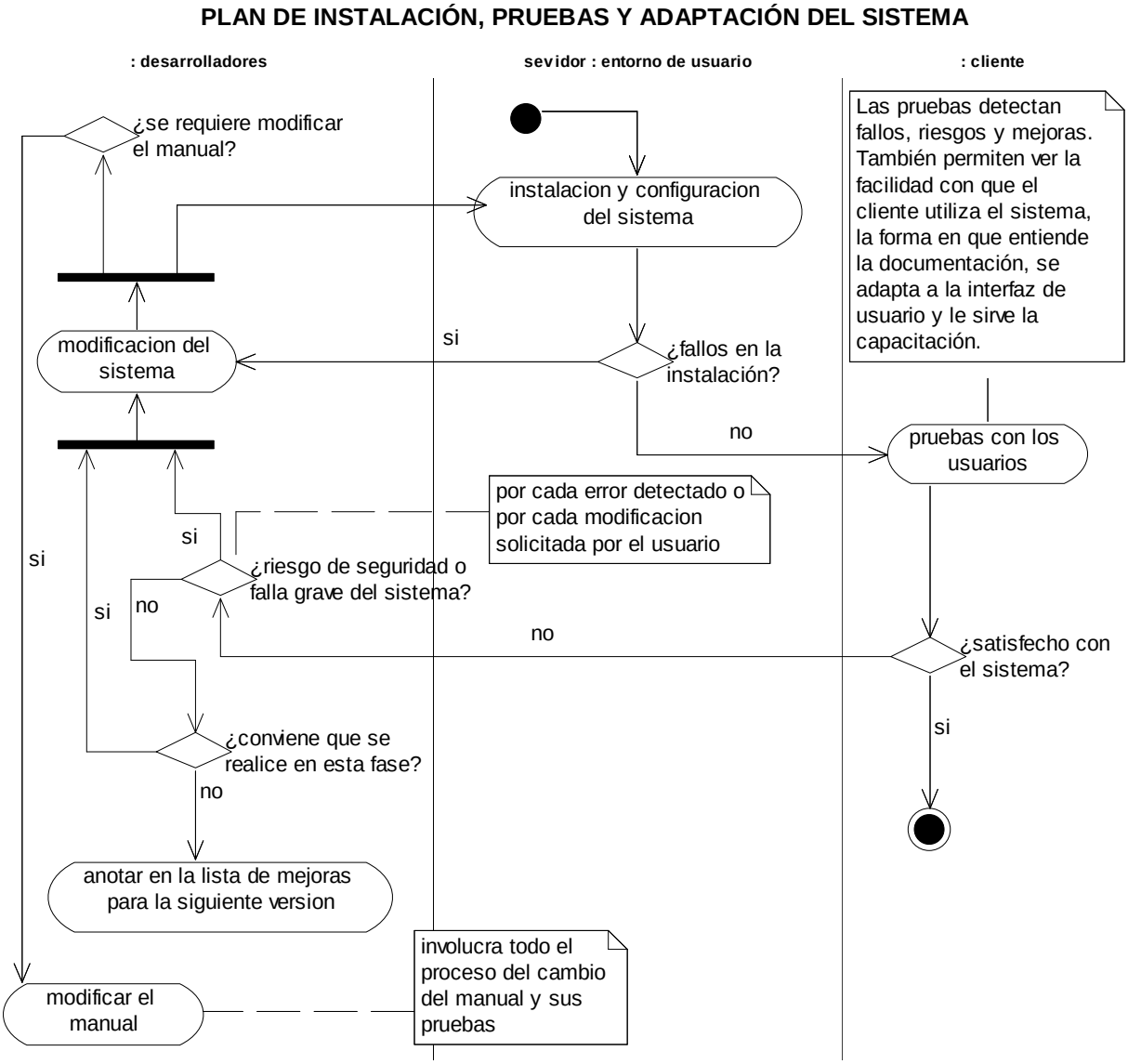

Fig. 2.6.2-3: Plan de instalación, pruebas y adaptación del sistema

2.6.3. Mejoras al Sistema Propuestas para Versiones Posteriores

A continuación se listan las mejoras y nuevos servicios que se han propuesto para futuras versiones; estas mejoras se han ido añadiendo a lo largo de las distintas fases de desarrollo.

TABLA 2.6.3-1: Lista de mejoras y/o servicios adicionales

Mejora o servicio:	Descripción
Nuevos tipos de documentos	Además de los archivos, se tendrán *enlaces, eventos* y *noticias*.[7]
Clasificación de usuarios más robusta.	Para prevenir posibles cambios organizacionales (que posiblemente influirán en los requisitos del sistema) se propone el uso del patrón *Actor-Rol*[8] para modelar la organización y manejar sus cambios y requisitos de seguridad.
Rediseño de la arquitectura	El objetivo es mejorar el rendimiento y hacer el sistema más robusto y ligero. Para esto se propone el uso de patrones de diseño y arquitectura probados y reconocidos, como *MVC*, *Micro-Kernel*, *Decorator, Observer, (Abstract)Factory*, *J2EE Patterns*, etc.[9]
Modificar datos personales.	El consejero podrá modificar sus datos personales.
Eliminación de usuarios.	El autorizador podrá dar de baja usuarios.
Modificación de temas.	El autorizador podrá modificar un tema existente.
Eliminación de temas.	El autorizador podrá eliminar un tema existente.
Búsqueda.	Se podrán realizar búsquedas de documentos y temas en el sitio.
Encuestas.	Creación, aplicación y publicación de sondeos mediante el sitio.
Servicios RSS.[10]	Publicación de una versión RSS de la sección pública del sitio para vinculación con otros sitios de Internet. Vinculación con otras instituciones afines publicando la información de sus respectivos sitios mediante la norma RSS.
Cifrado de documentos e información confidencial.	Los archivos actualmente se encuentran en una carpeta oculta del servidor y sus nombres son números de serie; para descargar un documento el sistema realiza las transformaciones convenientes de forma tal que mantiene oculta la ruta a los archivos. Como un servicio adicional se propone cifrar los archivos y demás tipos de documentos, así como la información de los usuarios. Esto supone un sistema de gestión de contraseñas adecuado.
Compresión de archivos.	Se analizará la posibilidad de comprimir los archivos para ahorrar espacio.
Interfaz de usuario flexible	Que la información se pueda mostrar en distintos formatos como la *versión para imprimir*, *flash*, etc.
Mejoras estéticas	Por ejemplo, el uso de íconos y de imágenes varias, la reubicación de los elementos, aplicación del concepto de *Usability*,[11] etc.
Servicios de correo.	Cuando se actualiza el sitio se envía un correo a los usuarios registrados. Cuando se eliminen documentos del sitio de forma automática se notificará a los autorizadores por correo electrónico con anticipación para realizar respaldos y cuando se realizó la eliminación para notificar resultados.[12]
Servicios de bitácora.	Permitirán detectar fallos en el sistema y ataques a la seguridad.

[7] Sección II.D.1 *Descripción de la Arquitectura;* modelo de análisis, paquete "*modelo de datos*".
[8] Hans-Erik Ericsson, Magnus Pender; *Business Modeling With Uml Business Patterns at Work: Business Patterns at Work*; John Wiley & Sons Inc; 1a. Ed. 2000
[9] Para más información sobre los patrones consulte: http://patternshare.org/ y http://www.cetus-links.org
[10] http://web.resource.org/rss/1.0/spec
[11] Jakob Nielsen; *Designing Web Usability: The Practice of Simplicity*; New Riders Publishing, 7th printing.
[12] Sección II.E.2 *Fase de elaboración*; "Rotación automática".

CONCLUSIONES

Este capítulo resume las conclusiones en los siguientes rubros: a) resultados y satisfacción del cliente. b) evaluación del método utilizado. c) evaluación de la inclusión del Diagrama Entidad Relación. d) conclusiones acerca del uso de XML. e) alcance del *patrón genérico*

3. CONCLUSIONES

3.1. SATISFACCIÓN DEL CLIENTE

Este proyecto de tesis cumple el objetivo de controlar el *flujo de documentos* mediante un sistema en un sitio Web, satisface el objetivo de brindar una imagen al público, esa información está disponible a más de diez millones de usuarios de Internet en México. También se cubrió el objetivo de llevar un control de los consejeros y que los consejeros puedan publicar sus opiniones y resultados de su trabajo en dos niveles, público y privado; además los temas se gestionan de manera automática

Los requisitos de la primera entrega están documentados en la descripción de la arquitectura. Todos se cubrieron satisfactoriamente. La versión mínima fue mostrada al cliente y éste quedó satisfecho. Los requisitos faltantes están documentados en la fase de transición.

La arquitectura ofrece la cualidad de la robustez, es decir, es resistente a los cambios en los requisitos.

POSIBILIDADES DE LA ARQUITECTURA

Fig. 3-1: Modelo en capas mostrando las posibilidades de la arquitectura

La figura *3-1* ilustra la flexibilidad del sitio en donde la apariencia es independiente de los datos a presentar y los servicios son independientes de la forma en que se almacena la información.

3.2. EL MÉTODO RUP FUNCIONA ADECUADAMENTE

Los fundamentos de RUP se confirmaron. Se comprobó la validez de los principios de RUP, así como la conveniencia de utilizar este método de desarrollo de software. El hecho de que la arquitectura sea *multicapa* permite que en futuros desarrollos se pueda mejorar cada aspecto del sistema sin que afecte a las demás partes.

3.2.1. Para que RUP funcione...

Se debe desarrollar iterativamente. En este caso concreto se buscó que la carga de requisitos, análisis y diseño, implementación y pruebas estuvieran balanceadas de acuerdo con las fases de RUP y el propósito que se buscaba en las mismas fases. Esto permitió resolver a tiempo conflictos importantes y realizar los cambios que evitaran el colapso del sistema.

Se deben utilizar tecnologías y metodologías orientadas a objetos. Se planteó desde el principio y la puesta en práctica fue progresiva a medida que se mejoraban las habilidades para utilizar el enfoque orientado a objetos. Asimismo se generaron diversos componentes (clases) que pueden ser utilizados en otros proyectos.

El desarrollo del sistema está centrado en los riesgos. La gestión de riesgos es una forma de trabajar *proactivamente* ante posibles amenazas y problemas que puedan aparecer en el

desarrollo del sistema. Prevenir y solucionar éstos problemas evita que, en el peor de los casos, se colapse el proyecto. Al gestionar los riesgos durante el desarrollo, se generó un sistema de alta calidad y gran flexibilidad.

3.3. DIAGRAMA ENTIDAD RELACIÓN (DER)

En la fase de inicio se planteó la hipótesis siguiente:

Aunque el DER no forma parte de RUP ni de UML, resulta conveniente incluirlo.

Esta hipótesis se sustentaba en otras tres:

1. *Conviene usar el DER en una etapa intermedia entre el análisis y el diseño.*
2. *El DER (entre el análisis y el diseño) permite una comprensión más sencilla de la realidad que la alcanzada utilizando exclusivamente UML.*
3. *Permite pasar del análisis al diseño con mayor facilidad que UML exclusivamente.*

Del desarrollo de este proyecto concluimos que:

1. Un diagrama de clases permite comprender mejor la realidad que un DER.
2. Pasar del análisis al diseño directamente de UML es más sencillo que mediante un DER.
3. Un DER puede ser realizado mediante UML (utilizando un diagrama de clases) por lo que resulta redundante la utilización de DER's en otras notaciones.
4. La práctica ha mostrado que es mucho más difícil mantener un sistema donde el modelo de objetos (UML) esta dirigido por el modelo de datos (DER) que uno donde el modelo de datos es *mapeado* del modelo de objetos (técnica OR-Mapping)
5. Por lo tanto resulta inconveniente el DER entre el análisis y el diseño.
6. Por tanto las tres hipótesis en que se apoya la hipótesis principal resultan falsas.
7. Sin embargo, conviene utilizar el DER para la generación automática de código SQL

3.4. XML

Escogimos XML como estándar para la comunicación entre subsistemas por las siguientes ventajas:

1. XML es independiente de la plataforma (*cross platform*)
2. XML es independiente de la aplicación
3. XML está basado en estándares
4. XML tiene una amplia aceptación en la industria (lo que significa que muchas personas y empresas desarrollan para y basados en XML)
5. Los documentos XML son legibles por humanos
6. XML separa el contenido de la presentación

Del desarrollo del sistema concluimos que:

1. Trabajar con documentos XML es una tarea que consume muchos recursos (memoria, código, tiempo de desarrollo, etc).
2. Conviene diseñar el documento XML con una estructura simple y flexible.
3. Conviene usar clases y objetos para representar los datos que se comunican los subsistemas (*command pattern*) y utilizar XML como una forma de implementación concreta de esos objetos (*bridge pattern*).

3.5. “PATRÓN GENÉRICO”

Christopher Alexander afirma: «*Cada patrón describe un problema que se repite una y otra vez en nuestro entorno, y entonces describe el núcleo de la solución de ese mismo problema, en una forma en que uno puede utilizar esa solución más de cien mil veces, sin recorrer nunca el mismo camino dos veces*».

Este patrón es de comportamiento y no de diseño Es decir, no hay un *mapeo* directo entre las clases del *patrón genérico* y las que conviene utilizar en la implementación concreta.

Sirvió para diseñar la arquitectura; esto se logró cuando se adaptó el *patrón genérico* para la *navegación*, entonces permitió identificar los diferentes subsistemas y sus características y sentó las bases de lo que sería la arquitectura que está actualmente en uso.

ANEXOS

Documentos anexos de esta tesis:
a) manual. b) glosario. c) referencias.

A) MANUAL DE USUARIO

El presente manual explica paso a paso los casos de uso del sistema del portal de la CMDH.

Se utilizaron los navegadores Mozilla 1.7.5 y Firefox 1.0.2 para las imágenes pero no debe haber diferencias en otro navegador como el MSIE. Al ingresar la dirección del portal en el navegador aparece una página de bienvenida.

PANTALLA PRINCIPAL

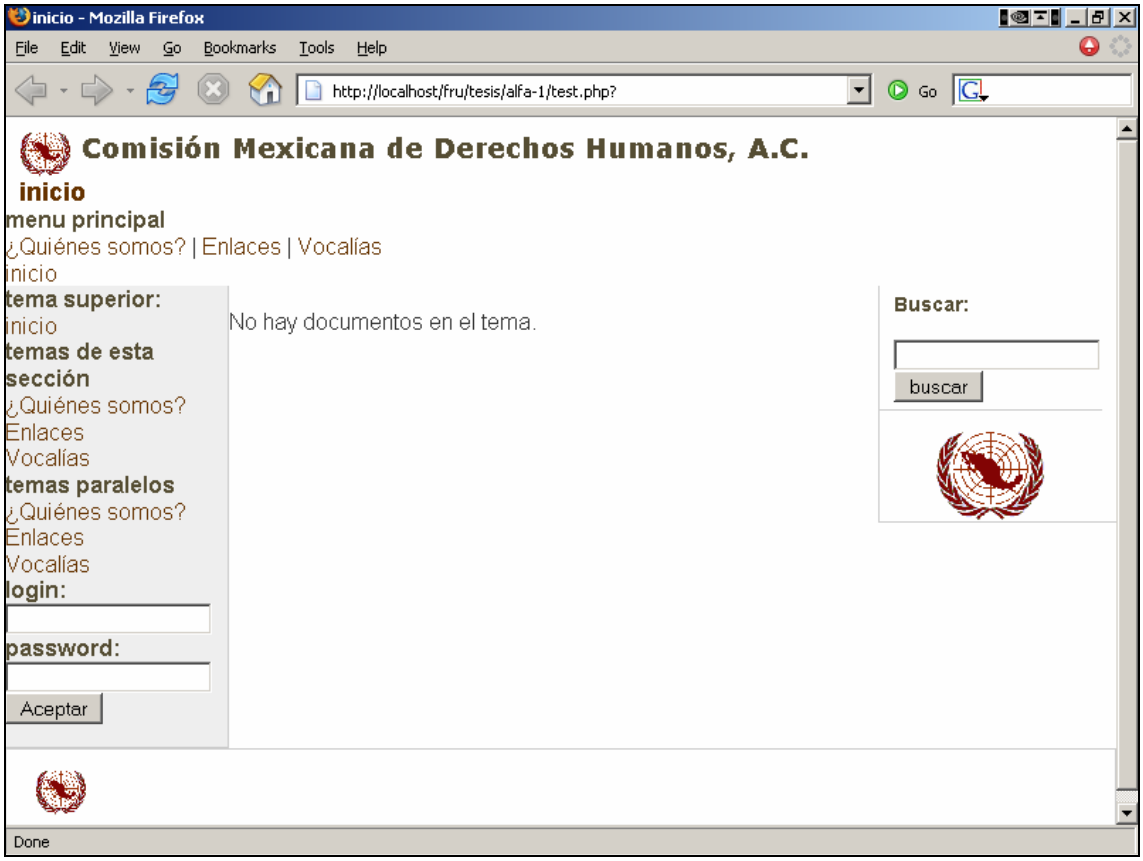

Fig. A-1: Pantalla de bienvenida al portal

En esta página podemos distinguir las principales secciones del portal. En la parte central está el contenido y alrededor están los hipervínculos a otras áreas del portal. El menú del lado izquierdo es el más importante ya que nos permite navegar por el sitio. Si no hemos iniciado sesión solo podremos ver los documentos públicos que están almacenados en el servidor.

Básicamente las operaciones que puede realizar un consejero en el sistema son: **Iniciar sesión**, **cerrar sesión**, **publicar documento** y **navegar** en busca de información almacenada en el sitio

A.1. Iniciar sesión (*consejero*)

Para iniciar una sesión se introduce en la sección del lado izquierdo en las cajas de texto el nombre de usuario (login) y la contraseña (password).

INICIAR SESIÓN

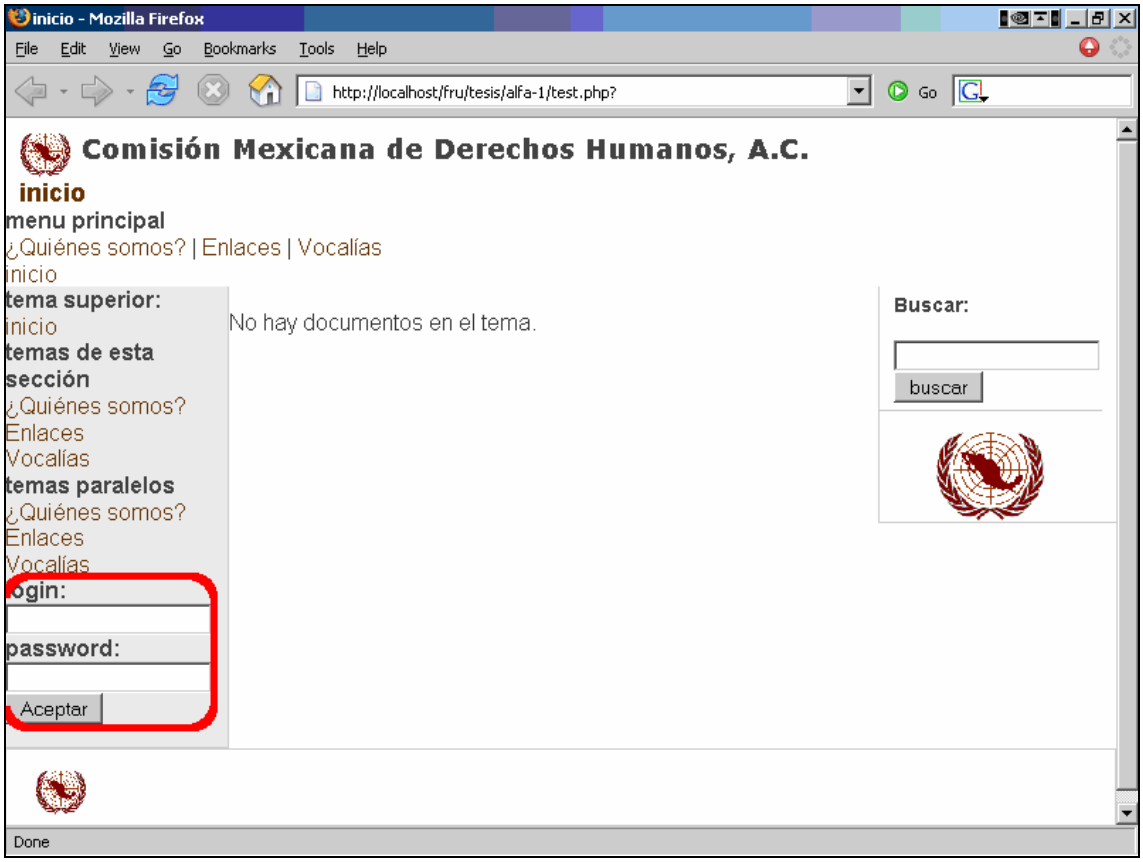

Fig. A-2: Iniciar sesión

Dependiendo de los permisos que tenga el usuario se desplegará una pantalla de sesión iniciada mostrando un menú donde antes pedía login y password.

PANTALLA DE SESIÓN INICIADA

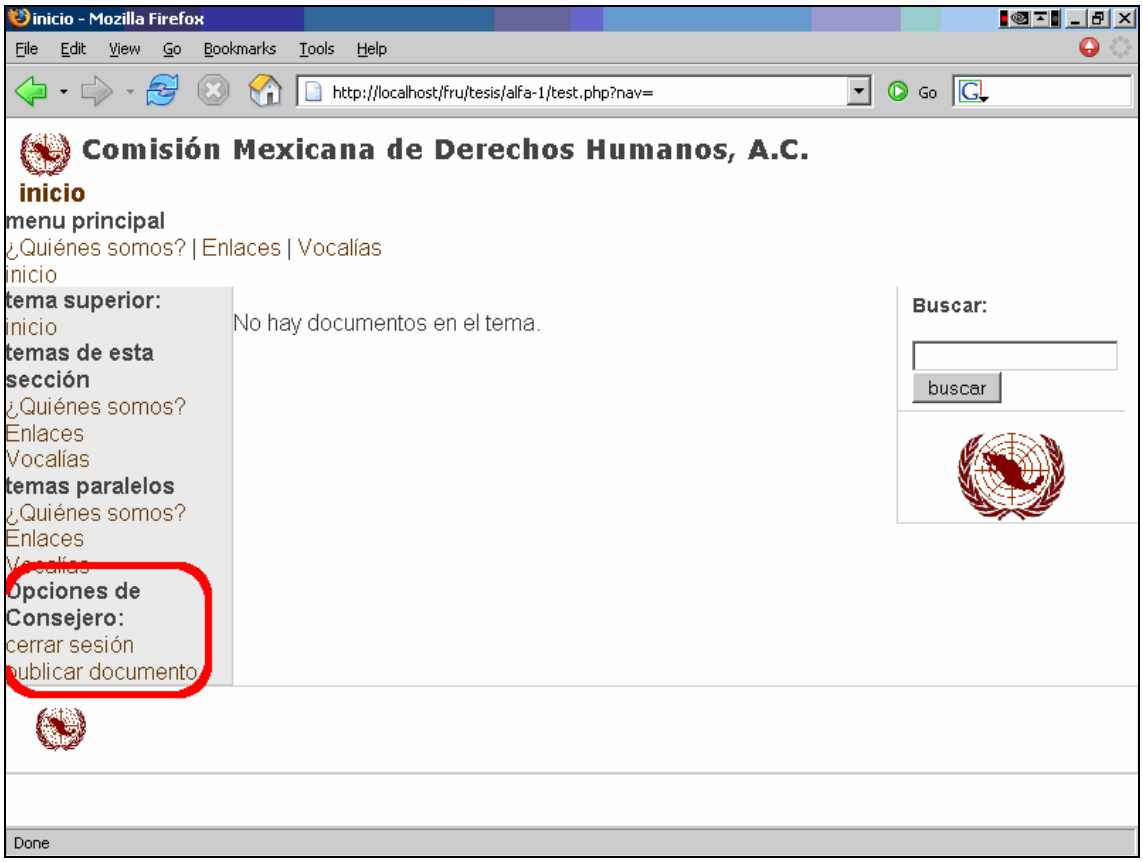

Fig. A-3: Pantalla con sesión iniciada

En esta pantalla se muestran las opciones del usuario consejero.

- **Cerrar sesión**: Termina la sesión iniciada y regresa a la pantalla principal pero ahora como un usuario sin permisos que solo podrá ver el material que es público.
- **Publicar documento**: Permite al usuario enviar un documento para publicarlo en este sitio previa autorización de un comité evaluador.

A.2 Publicar Documento

MENÚ PUBLICAR DOCUMENTO

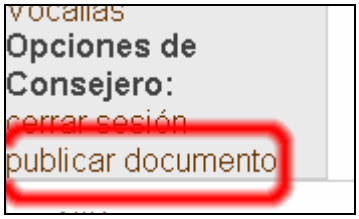

Fig. A-4: Menú Publicar Documento

Envío de documentos al sistema para ser publicados después de ser evaluados por los autorizadores.

PANTALLA DE PUBLICAR DOCUMENTO

Fig. A-5: Publicar Documento

Al seleccionar publicar documento en la pantalla principal el sistema nos presenta este formulario para ingresar un documento al sistema.

- **Título:** el titulo del documento a publicar.
- **Archivo:** se debe ingresar la ruta y nombre del archivo a publicar.
- **Descripción del archivo:** Descripción del contenido del archivo a publicar.
- **Palabras clave de búsqueda:** las palabras mas importantes que relacionan el contenido del documento para futuras búsquedas de información.
- **Tema:** el tema al que pertenece el archivo. Los temas son creados por los autorizadores.

Una vez lleno el formulario se pulsa el botón Aceptar. El sistema nos presenta la operación realizada.

ENVIAR EL ARCHIVO

Fig. A-6: El archivo enviado

Sólo resta esperar a que el documento sea autorizado para que este disponible y ser visto por lo usuarios.

A.3 Navegar

Navegar en el sistema sirve para revisar los documentos publicados en el sitio, en busca de información de interés. Los documentos deben ser descargados para poder ser leídos.

PANTALLA NAVEGANDO SIN SESIÓN

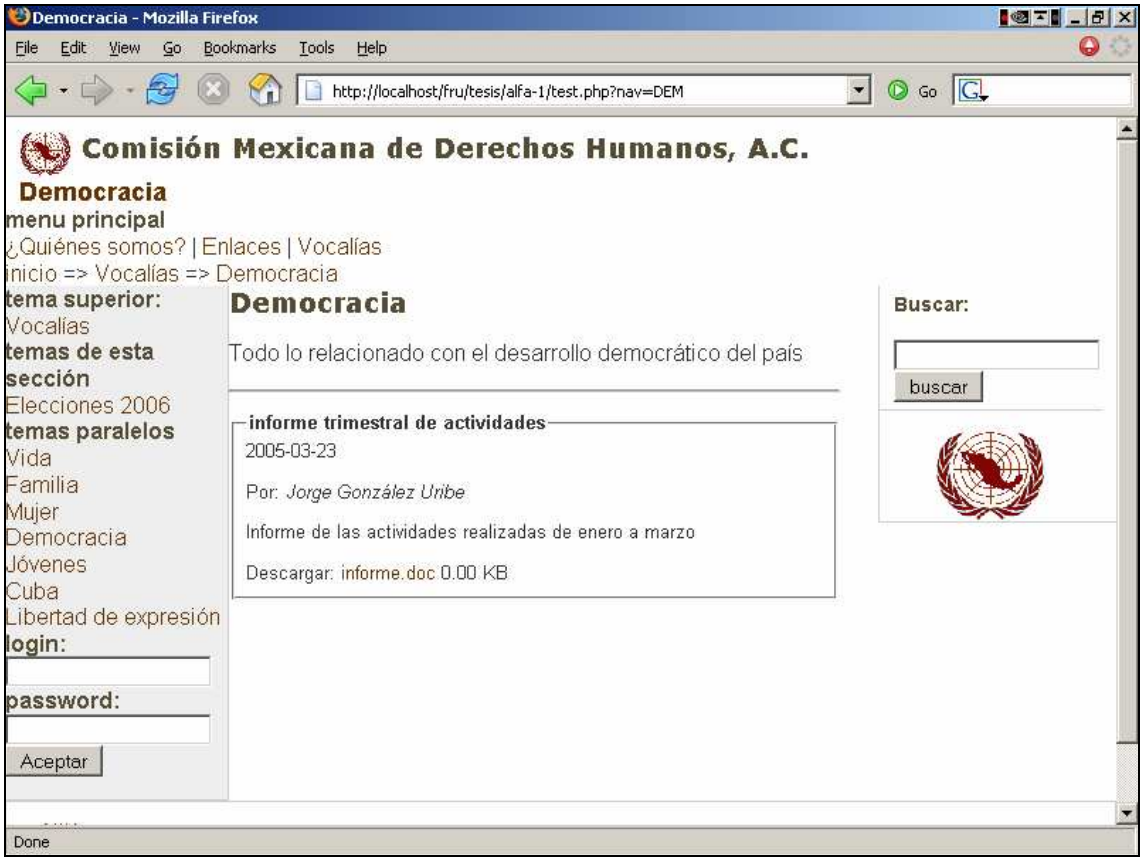

Fig. A-7: Navegando por los temas sin sesión iniciada

NAVEGANDO CON SESIÓN

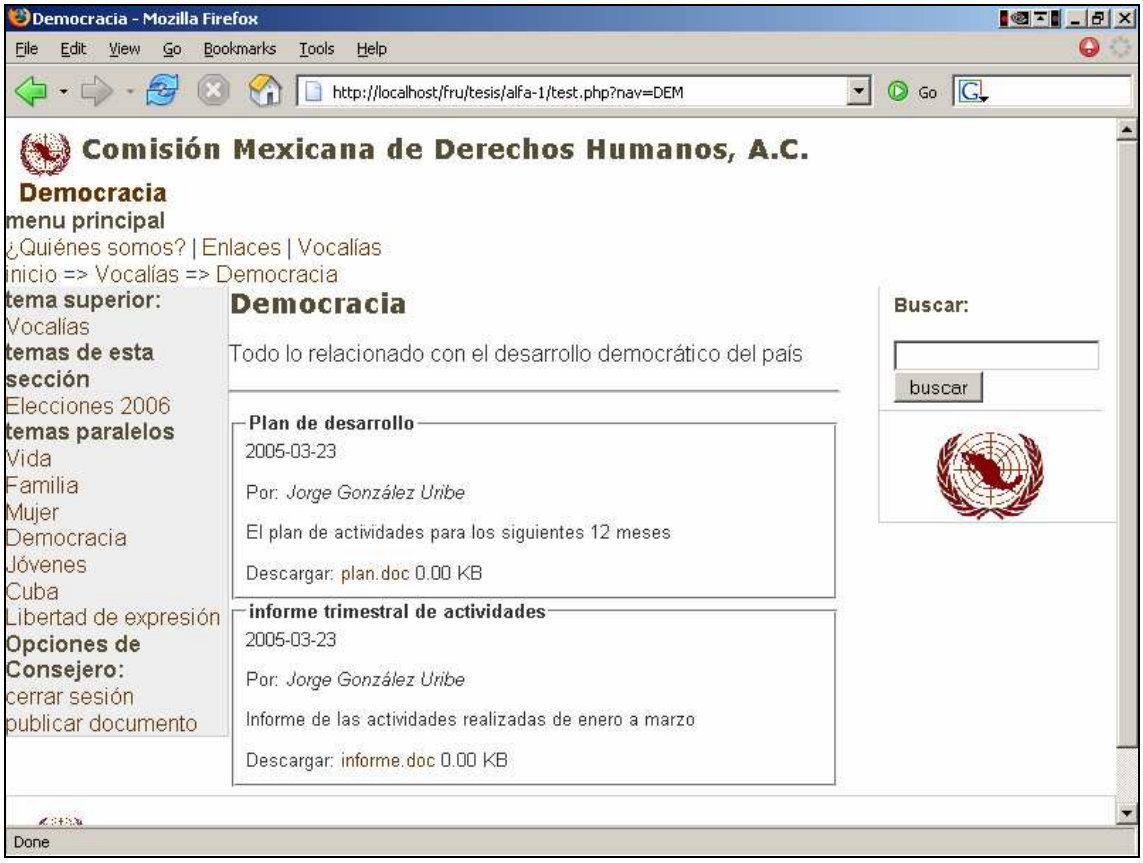

Fig. A-8: Navegando por los temas con sesión iniciada

Retomando navegación de usurario no registrado. La pantalla muestra diferentes secciones: menú de temas, barra de profundidad, tema superior, temas de la sección, subtemas (temas al mismo nivel) Estas son las posibilidades para moverse en los temas.

DESCARGAR

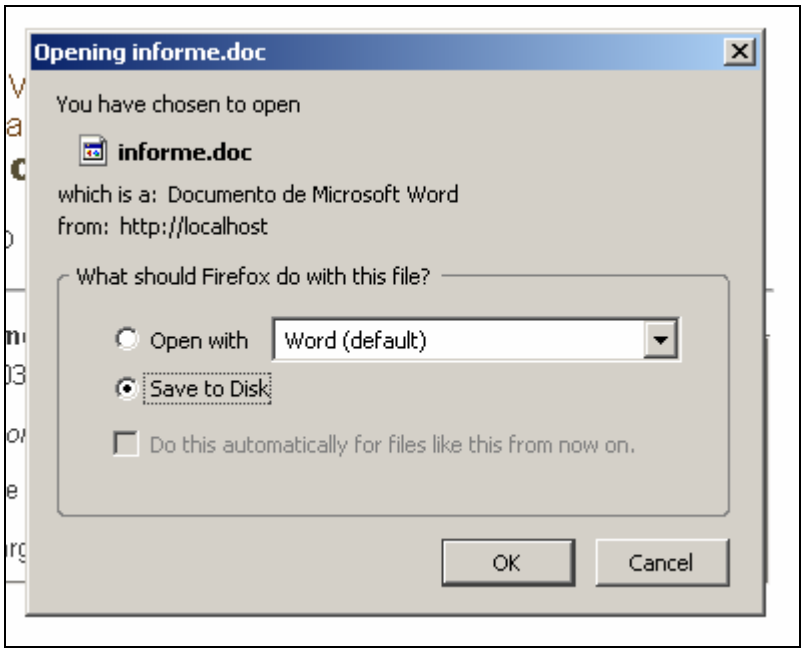

Fig. A-9: Descargando un documento

SELECCIONANDO UN SUBTEMA

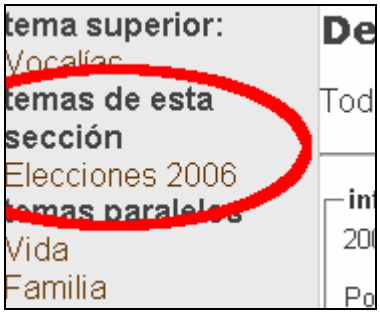

Fig. A-10: Seleccionando un subtema

Si seleccionamos el link del subtema Elecciones 2006

PANTALLA DE UN SUBTEMA

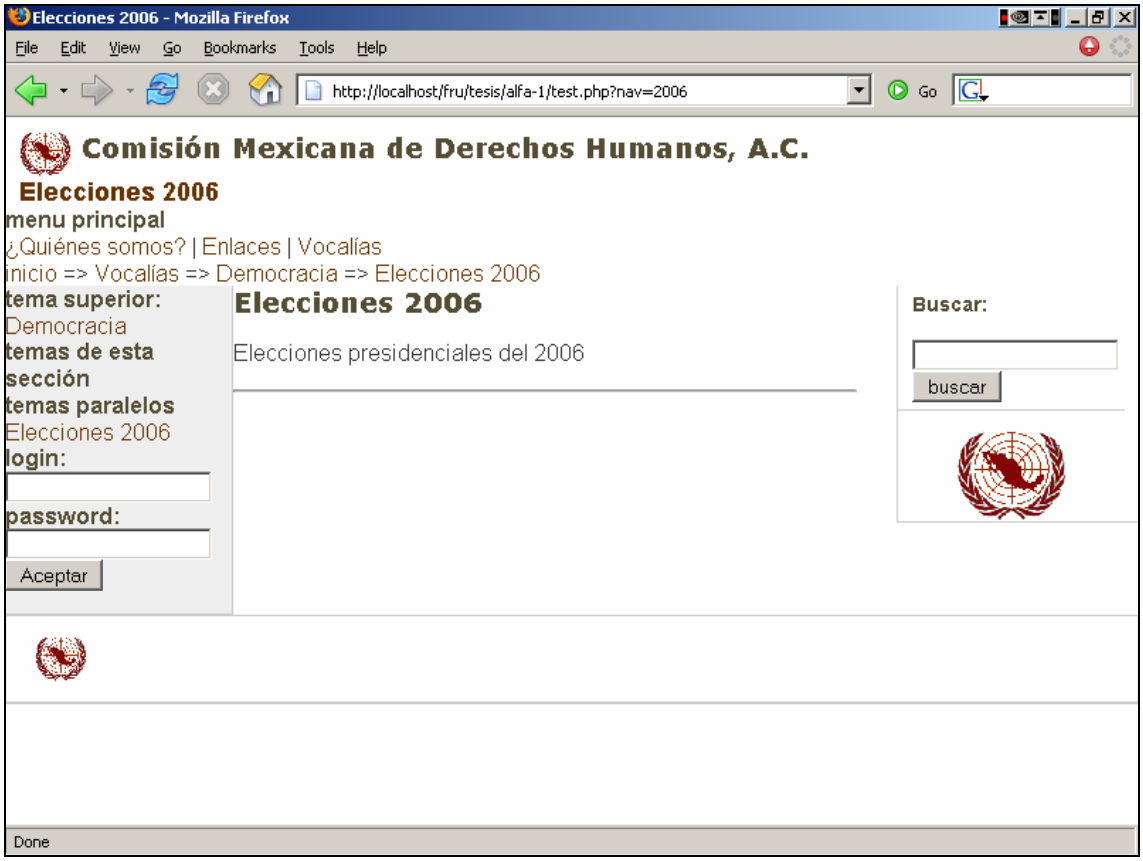

Fig. A-11: Pantalla de un subtema

En la pantalla se pueden apreciar cambios en tema superior y en los subtemas.

NAVEGANDO EN LOS TEMAS

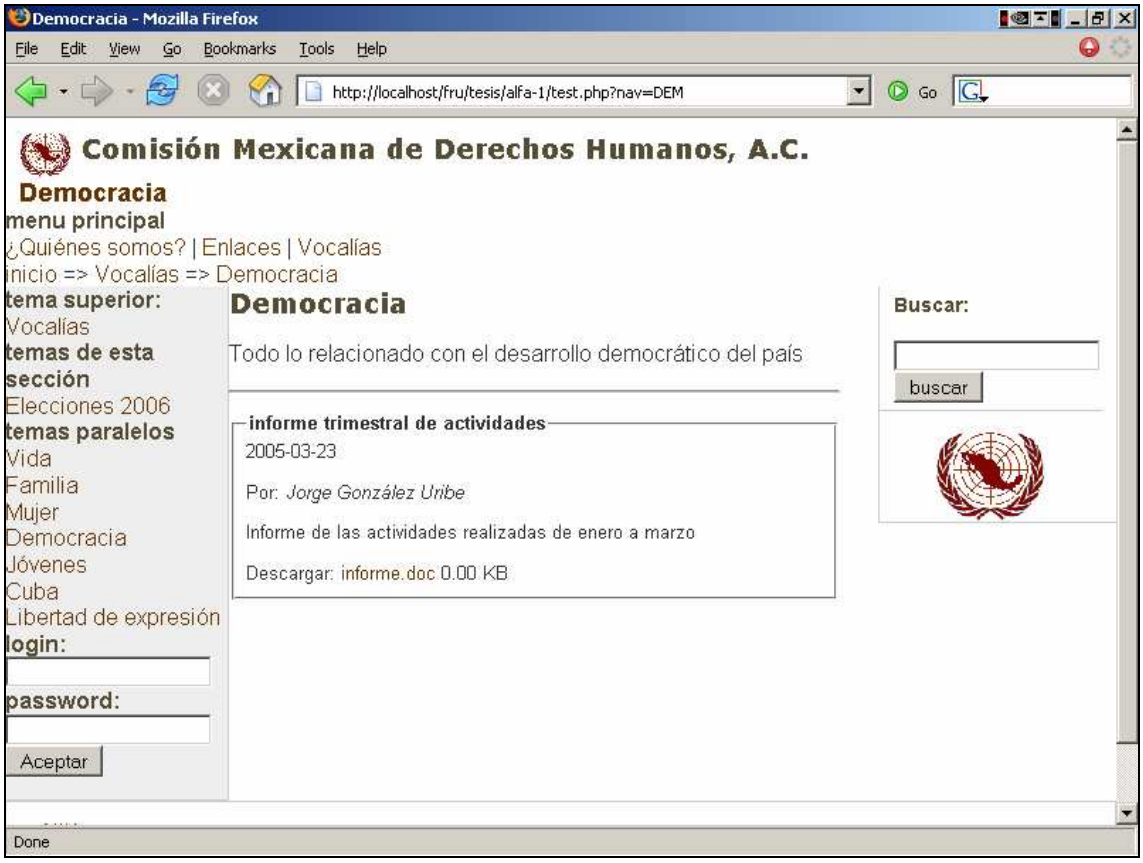

Fig. A-12: Navegando en los temas

Un usuario con permisos de Autorizador podrá realizar más operaciones dentro del sistema además de las mencionadas.

SESIÓN AUTORIZADOR

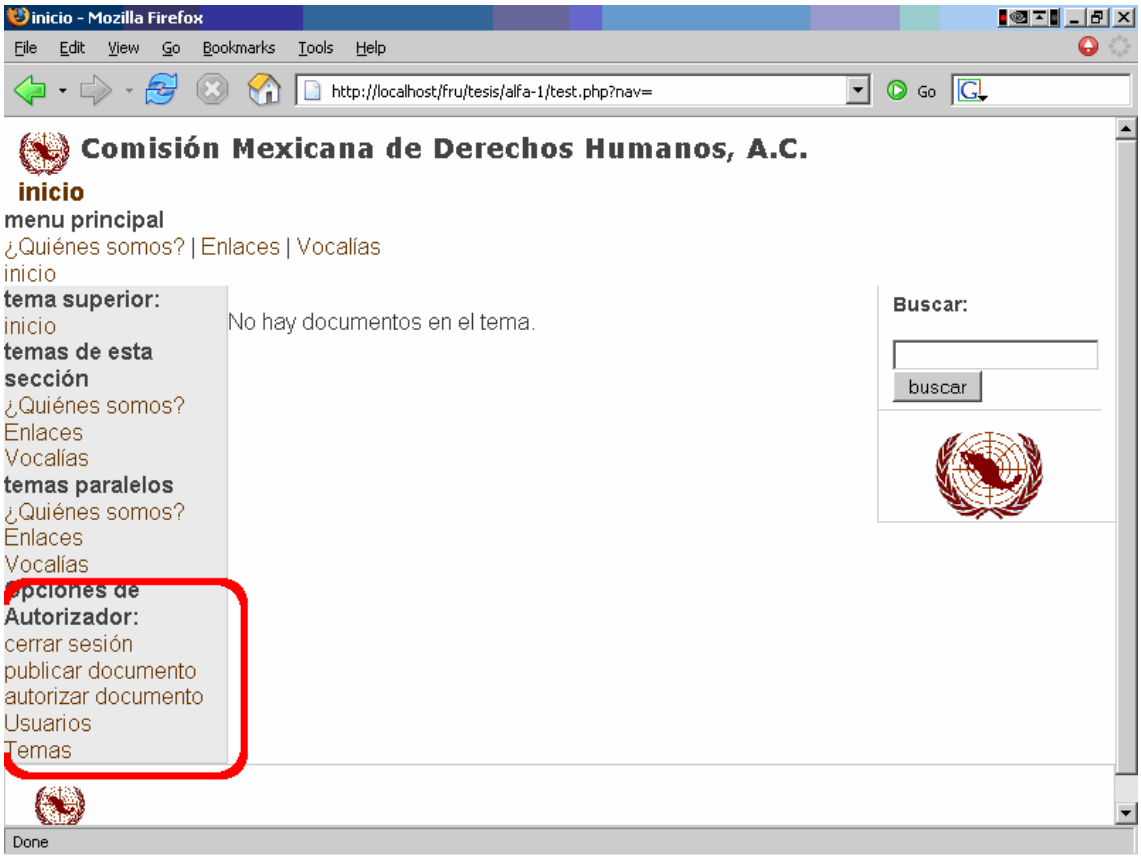

Fig. A-13: Iniciando sesión como Autorizador

MENU AUTORIZADOR

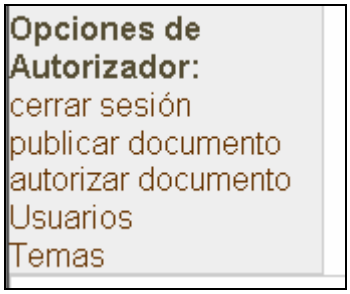

Fig.A-14 : Opciones del menú autorizador

Se han agregado tres opciones: **Autorizar documento**, **usuarios** y **temas**

A.5 Autorizar documento

El consejero autorizador es parte del comité que se encarga de evaluar los documentos enviados al sistema para ser publicados. Para autorizar un documento se siguen unos sencillos pasos:

PANTALLA AUTORIZAR

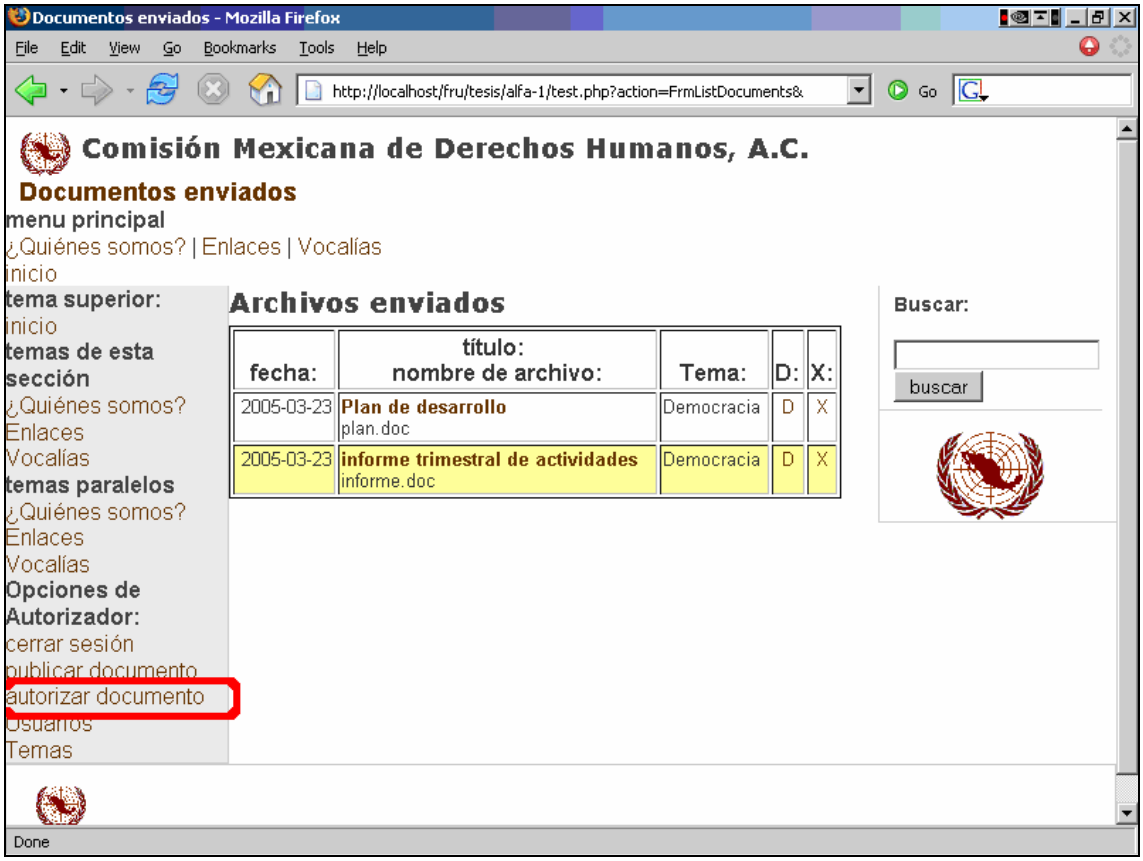

Fig.A-15 : Autorizar documento

Al seleccionar en el menú principal la opcion de autorizar documento. El sistema desplegara la página arriba. En el centro se encuentra una tabla que muestra los documentos enviados para ser publicados con fecha y tema al que pertenecen. En esta página se pueden descargar los archivos o eliminar. Las filas en color representan a los documentos que no han sido autorizados.

ELEMENTOS DE LA TABLA

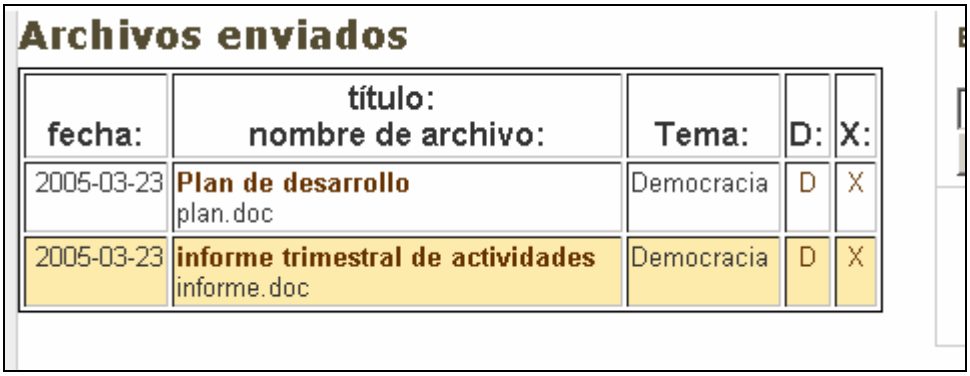

Fig. A-16 : Elementos de la tabla

- **Fecha:** La fecha en que se envió el documento
- **Titulo:** El titulo del documento
- **Nombre del archivo:** el nombre físico del archivo

- **Tema:** El tema a que pertenece.
- **D:** Link para descargar el documento
- **X:** Link para eliminar al documento

PANTALLA AUTORIZAR

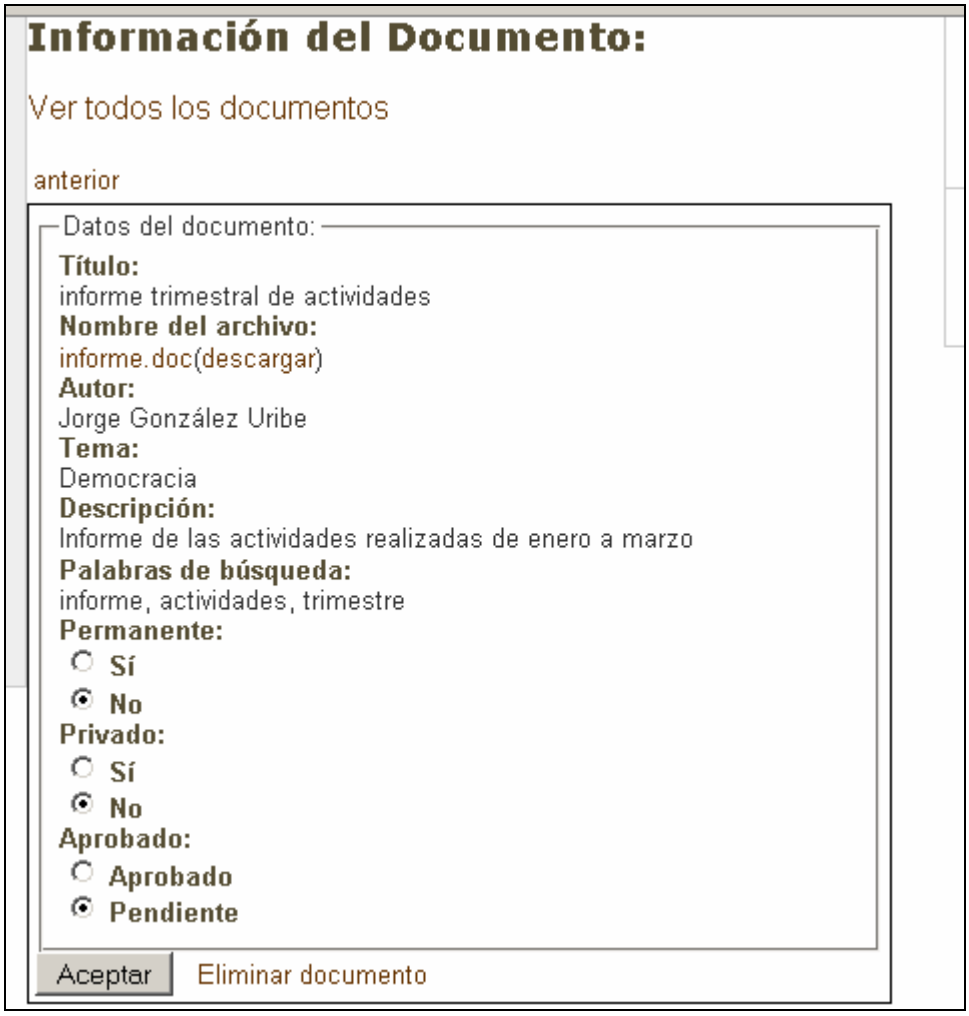

Fig. A-17 : Formulario para autorizar un documento

En el formulario aparecen los datos del documento, se puede descargar el archivo desde aquí. Hay que seleccionar los atributos de documento y oprimir aceptar.

- **Permanente:** El archivo no será eliminado automáticamente al rotar el contenido.
- **Privado:** Sólo podrá ser descargado y visto por usuarios consejeros.
- **Aprobado:** El documento queda aprobado y puede ser descargado y visto por los usuarios del sistema. Esta acción es irreversible.

Si el documento no es aprobado se queda pendiente; además se puede eliminar desde aquí seleccionando el hipervínculo Eliminar documento.

ELIMINAR UN DOCUMENTO

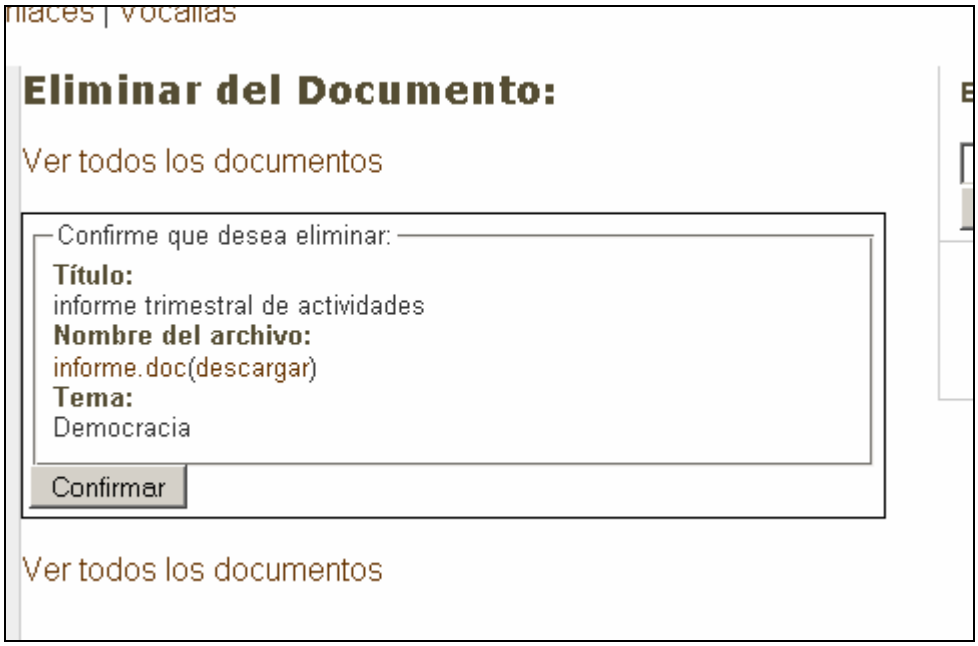

Fig. A-18: Eliminar un documento no autorizado

A.6. Usuarios

El consejero autorizador puede dar de alta nuevos usuarios en el sistema. Al seleccionar la opción Usuarios del menú principal se muestra la siguiente pantalla.

PANTALLA USUARIOS

Fig. A-19: Usuarios registrados en el sistema

Para dar de alta un nuevo usuario se selecciona el hipervínculo Agregar Consejero de esta página.

PANTALLA AGREGAR CONSEJERO

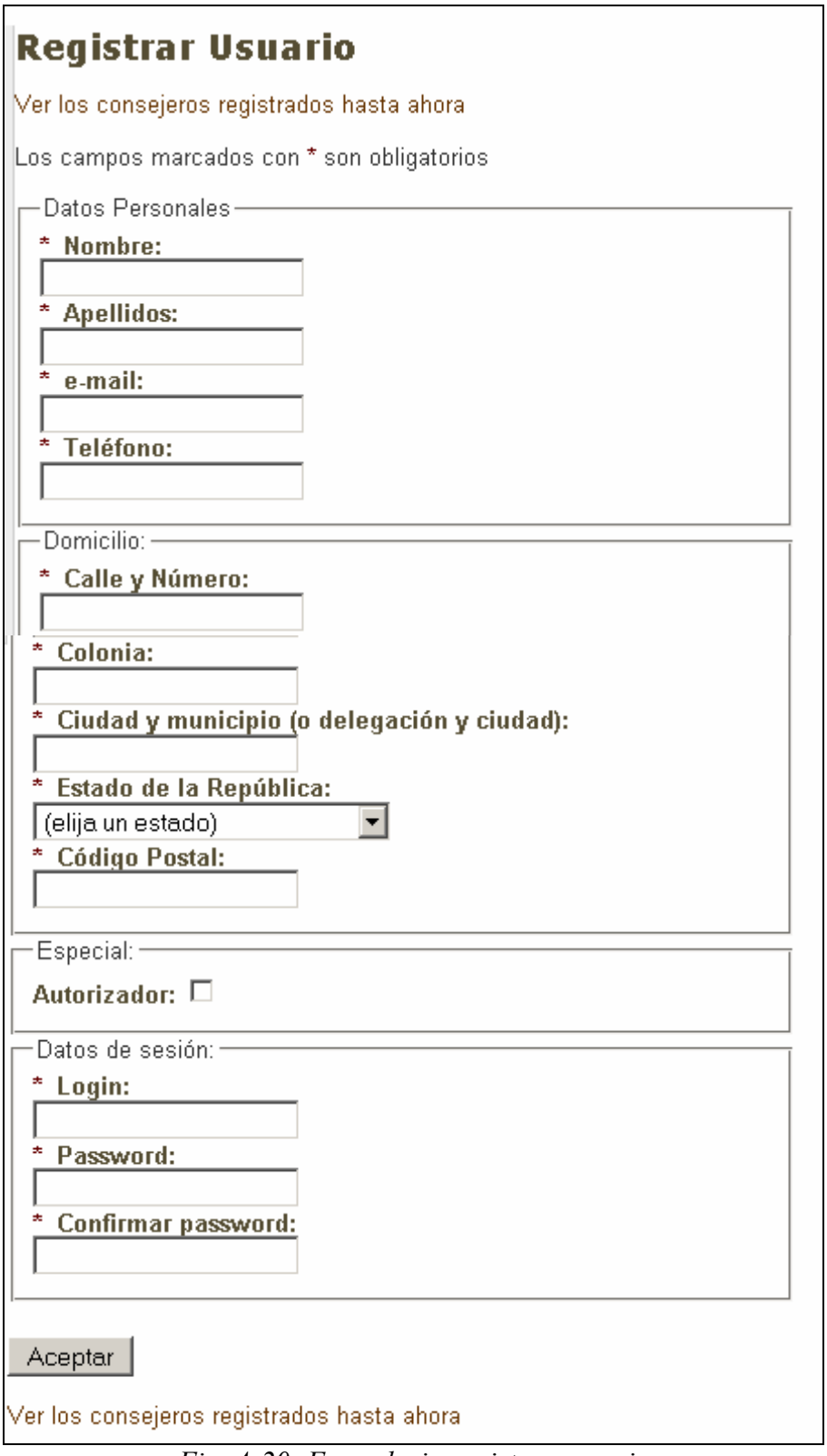

Fig. A-20: Formulario registrar usuario

Se deben capturar los datos personales del nuevo usuario, domicilio y datos de sesión que incluye asignarle un login y un password. En la sección especial hay una caja de verificación para darle permisos de autorizador (Control total sobre el sistema) al usuario

que se esta registrando, solo debe marcarse para los usuarios a quines se les da ese atributo.

DETALLES DEL USUARIO

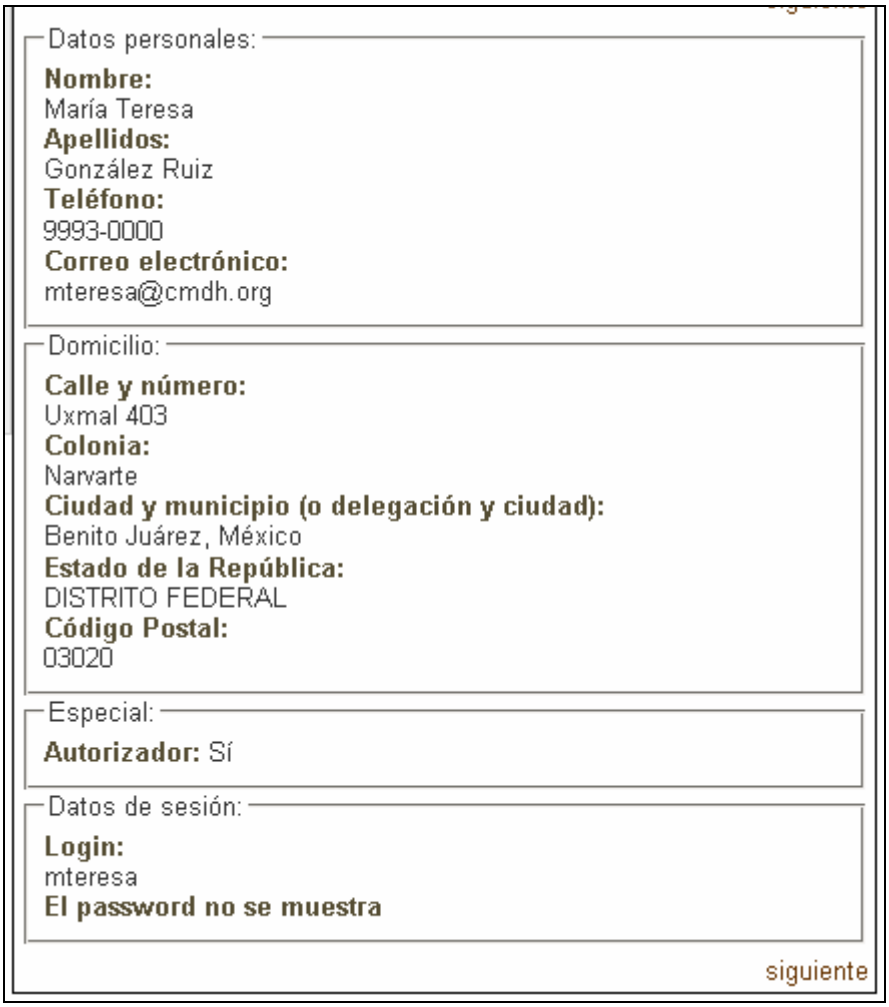

Fig. A-21: Detalles del usuario

El sistema permite ver los detalles de los usuarios registrados

A.7. Temas

El consejero administrador puede registrar nuevos temas en el sistema. Seleccionando la opción Temas en el menú principal aparece la siguiente pantalla.

PANTALLA DE TEMAS

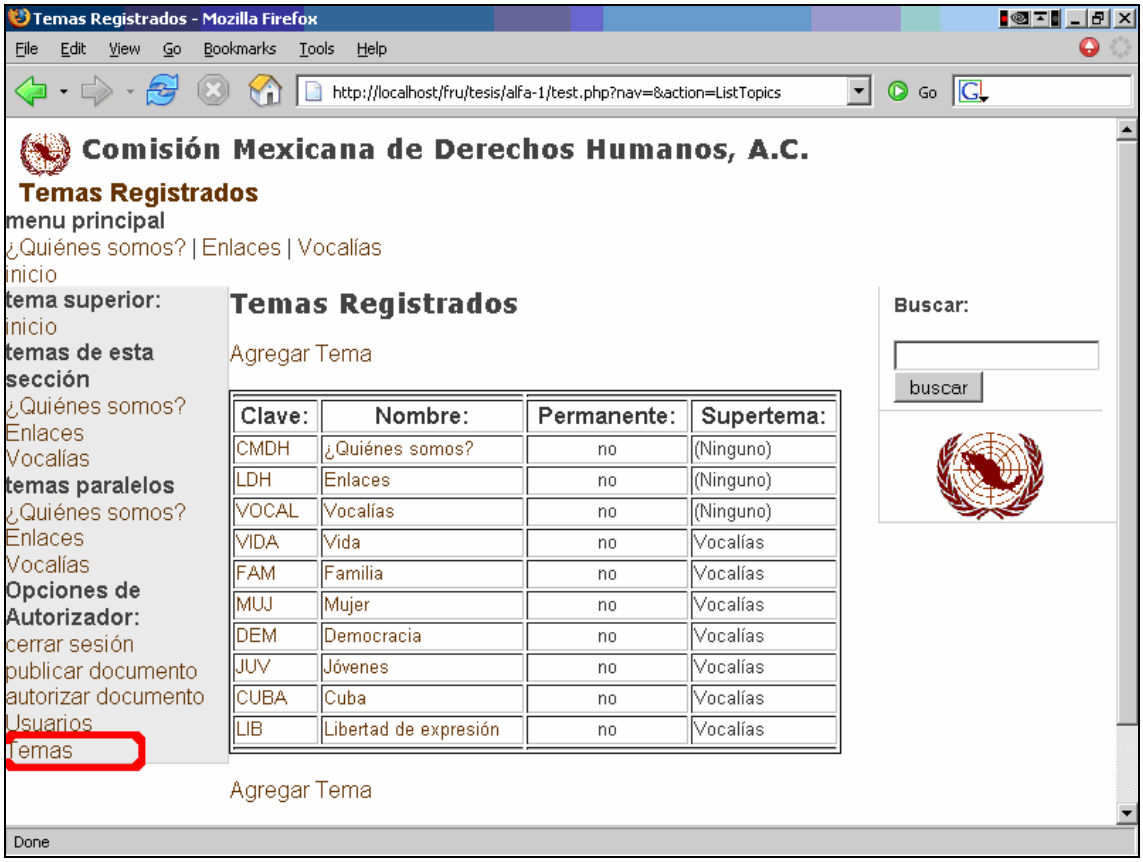

Fig. A-22: Pagina de temas

En esta página se pueden ver los detalles de los temas ya registrados. Para registrar un tema nuevo seleccionar el hipervínculo Agregar Tema.

PANTALLA AGREGAR TEMA

Fig. A-24: Formulario para registrar temas en el sistema

Se deben capturar los datos en el formulario para concluir el registro.

- **Nombre clave** : Clave del tema. Usar únicamente el alfabeto americano (no acentos, ni ñ). Se permiten números, guiones y subrayas.
- **Nombre del tema**: Nombre que se le asigna al tema nuevo.
- **Descipción del tema**: Resumen del tema
- **Palabras clave de búsqueda**: Palabras que se relacionan con el tema
- **Permanente**: no será eliminado del sistema
- **Tema superior**: El tema superior en caso de tener uno.

Una vez lleno el formulario seleccionar aceptar.

PANTALLA DETALLES DEL TEMA

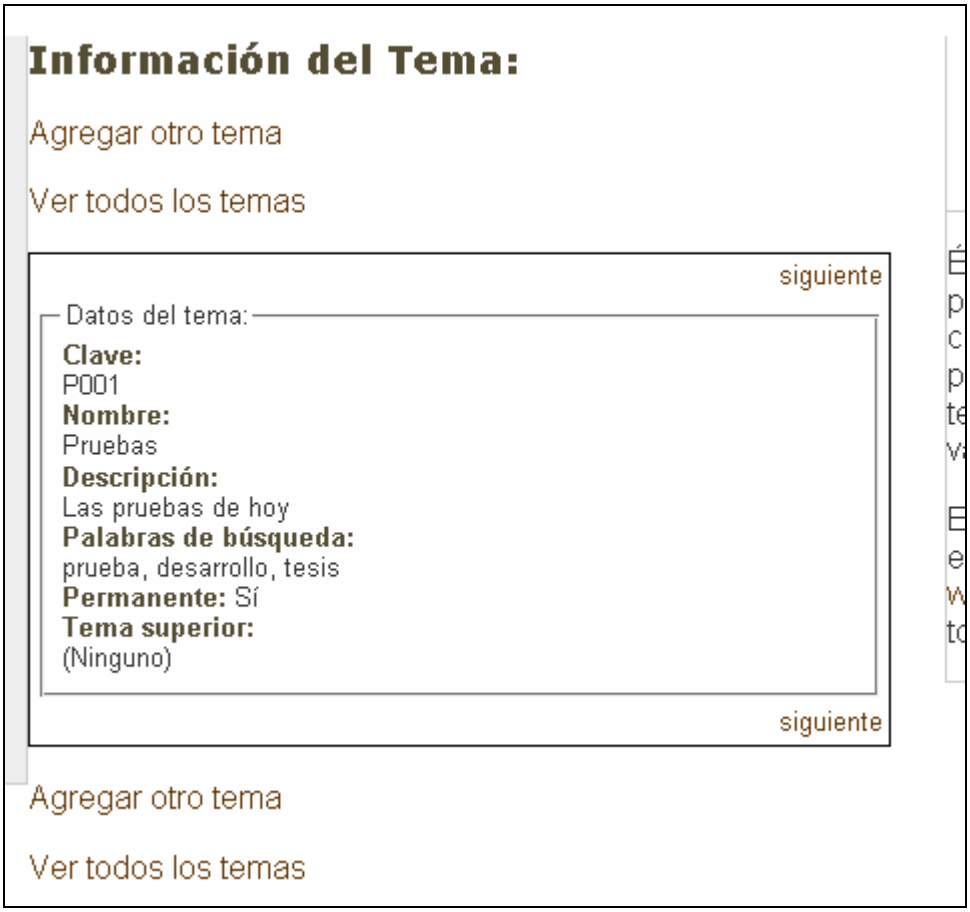

Fig. A-25: Los detalles del tema

El sistema permite ver los detalles de los temas registrados. Y se pueden agregar nuevos temas desde aquí.

A.8. Intalación

Para instalar el sistema por ahora se necesita que los desarrolladores instalen las bases de datos posteriormente se creará un script que lo haga automáticamente.

Recién instalado el sistema la primera vez que se ingresa las bases de datos están vacías. Veremos una pantalla como la siguiente

PANTALLA DEL SISTEMA VACÍO

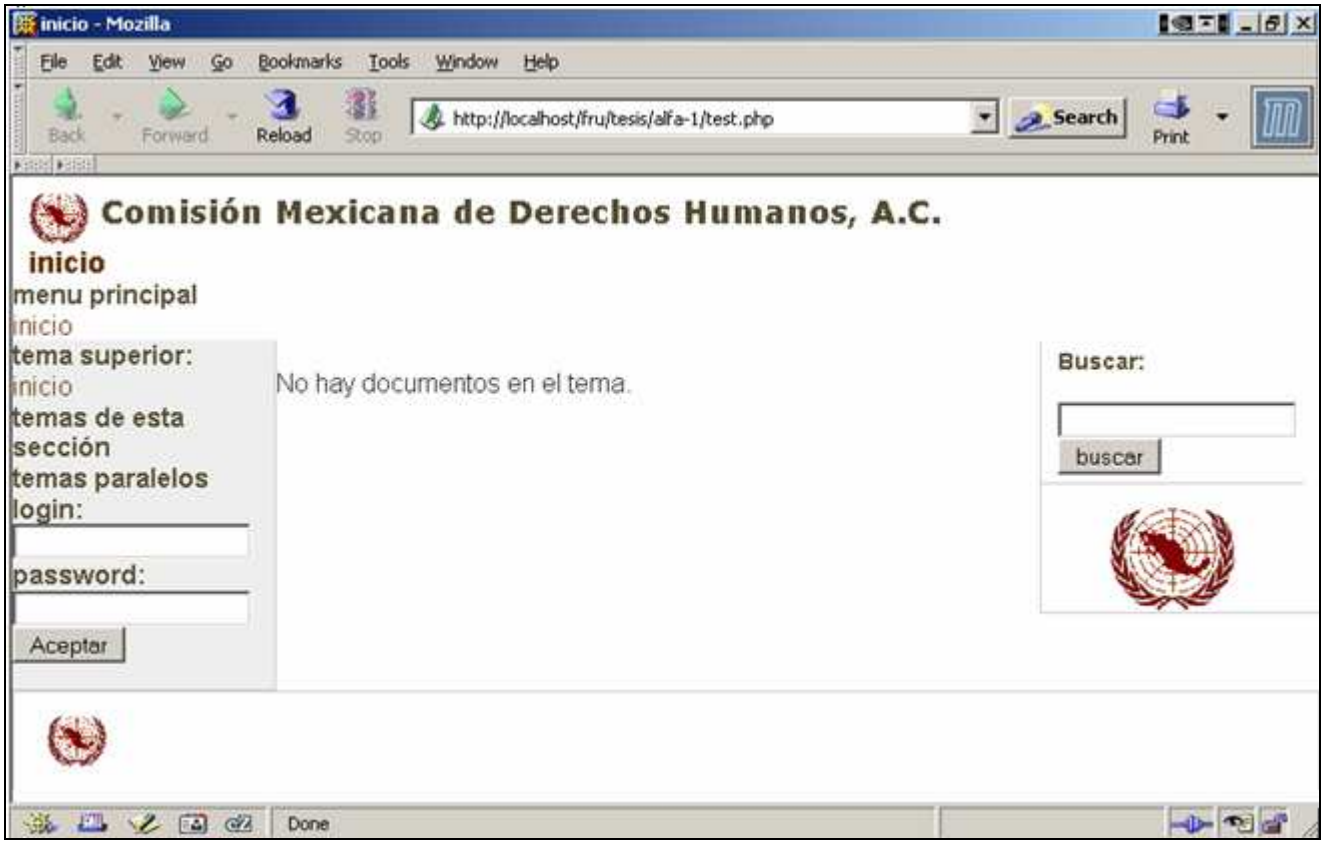

Fig. A-26: Pantalla del sistema vacío

Para iniciar la instalación se teclea la siguiente dirección:

http://www.cmdh.org/?action=FrmRegisterUser

Esta instrucción le pide al sistema que permita registra a un usuario.

PANTALLA DE REGISTRO DEL PRIMER USUARIO

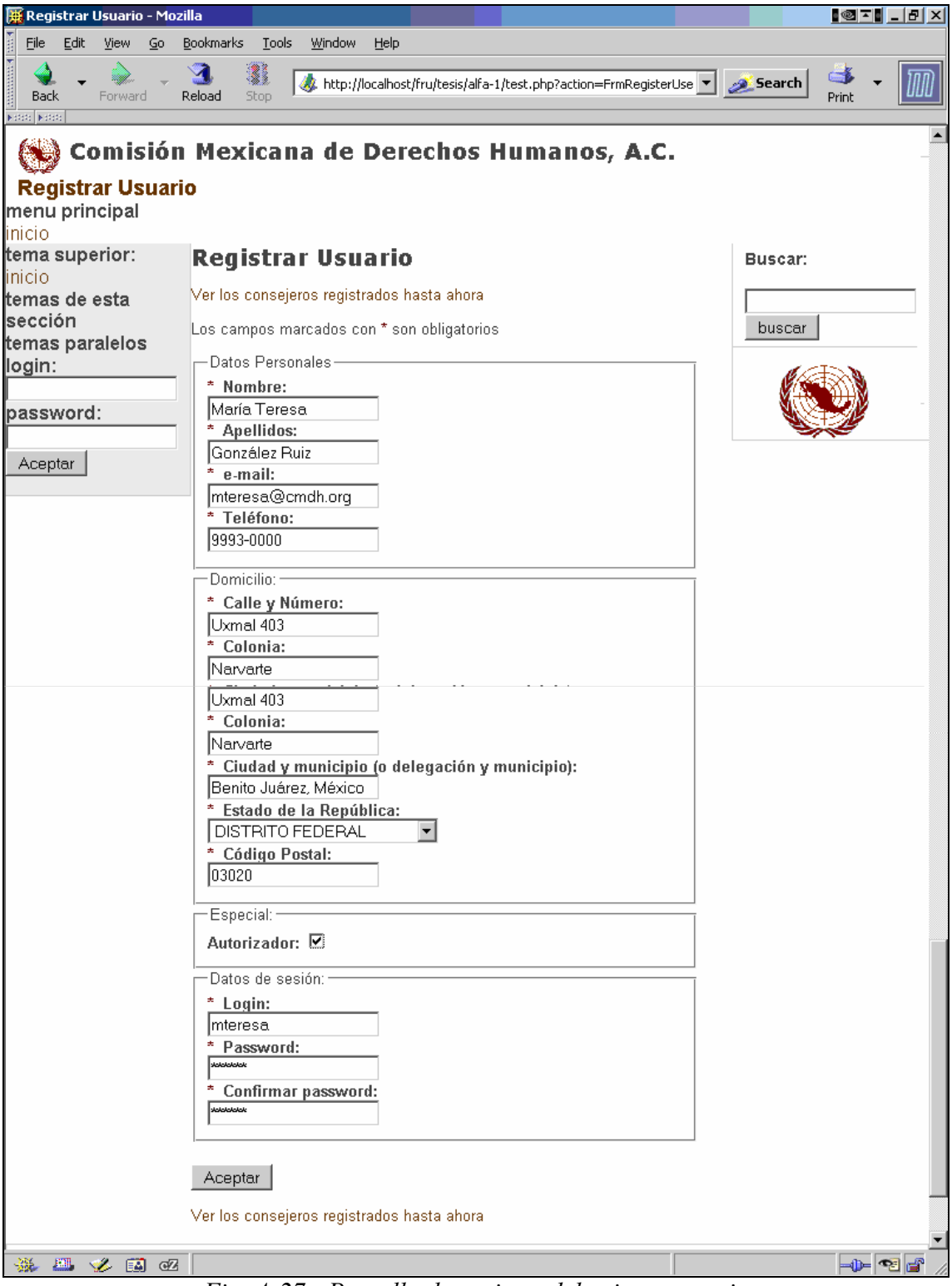

Fig. A-27 : Pantalla de registro del primer usuario

Se llena el formulario; es sumamente importante anotar el password de este primer usuario, ya que es el único que lo puede administrar por el momento, si no se han registrado otros usuarios y se pierde este password también se pierde el control de la administración del sistema. Una vez lleno el formulario se oprime el botón de Aceptar

PANTALLA DESPUÉS DE REGISTRAR EL PRIMER USUARIO

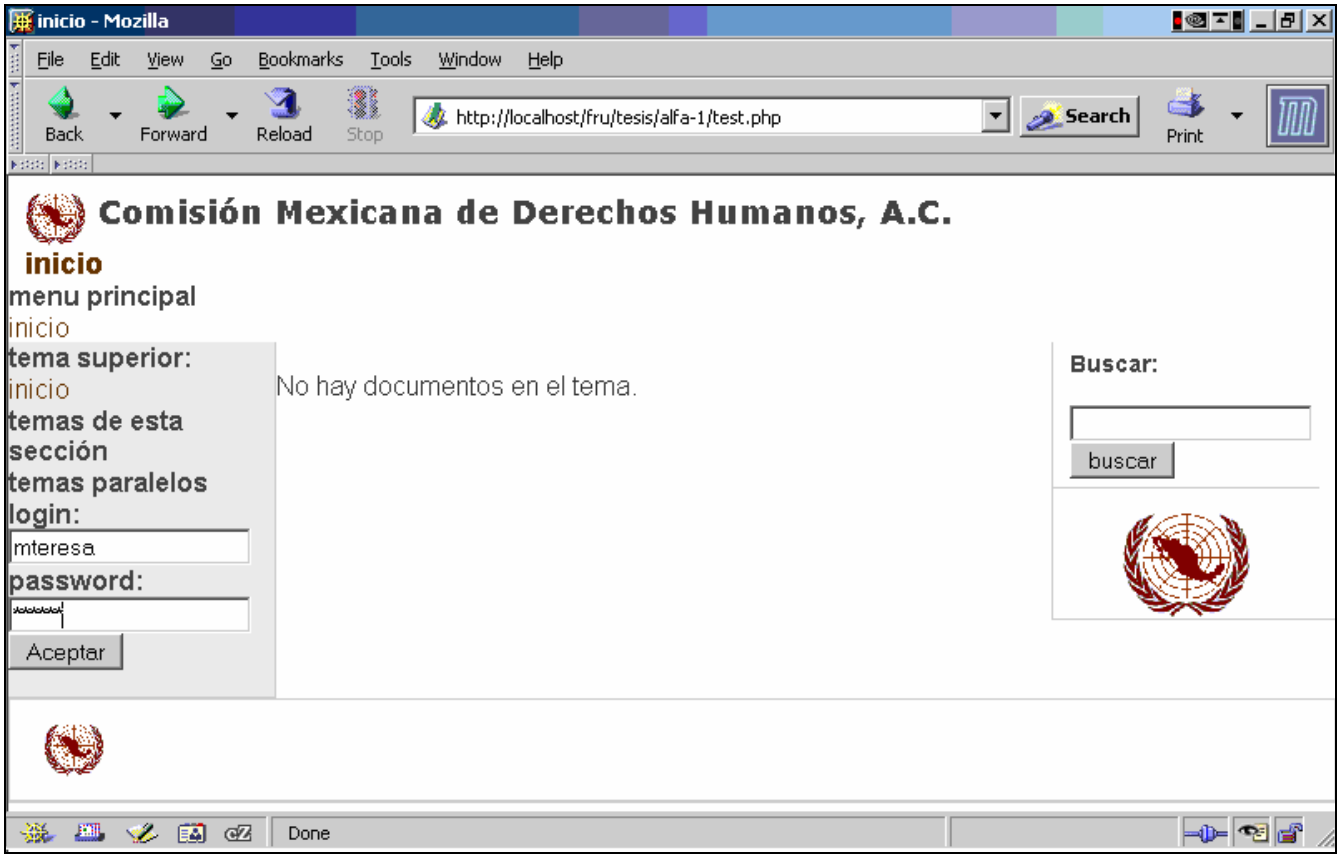

Fig. A-28: Pantalla después de registrar el primer usuario

Al ingresar al sistema con el login y el password del primer usuario debe aparecer la pantalla anterior si todo funciono adecuadamente. Si hay algún problema habrá que contactar con nosotros.

ESTA ES LA PANTALLA QUE DEBE SALIR SI TODO ESTA CORRECTO

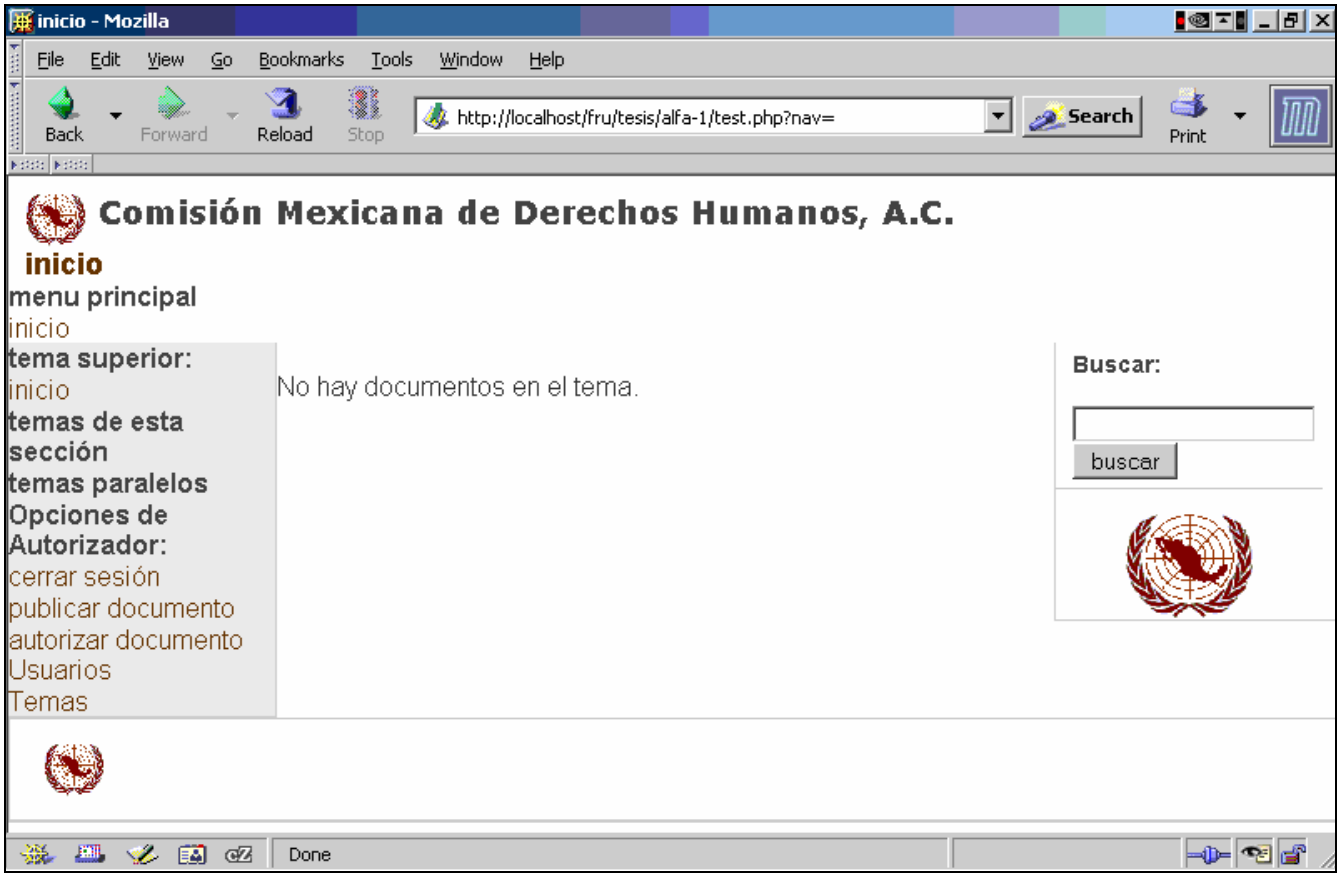

Fig. A-29: *Pantalla que debe salir si todo esta correcto*

En esta pantalla en la esquina inferior izquierda esta el menú. A continuación se selecciona esa opción

PANTALLA DE USUARIOS

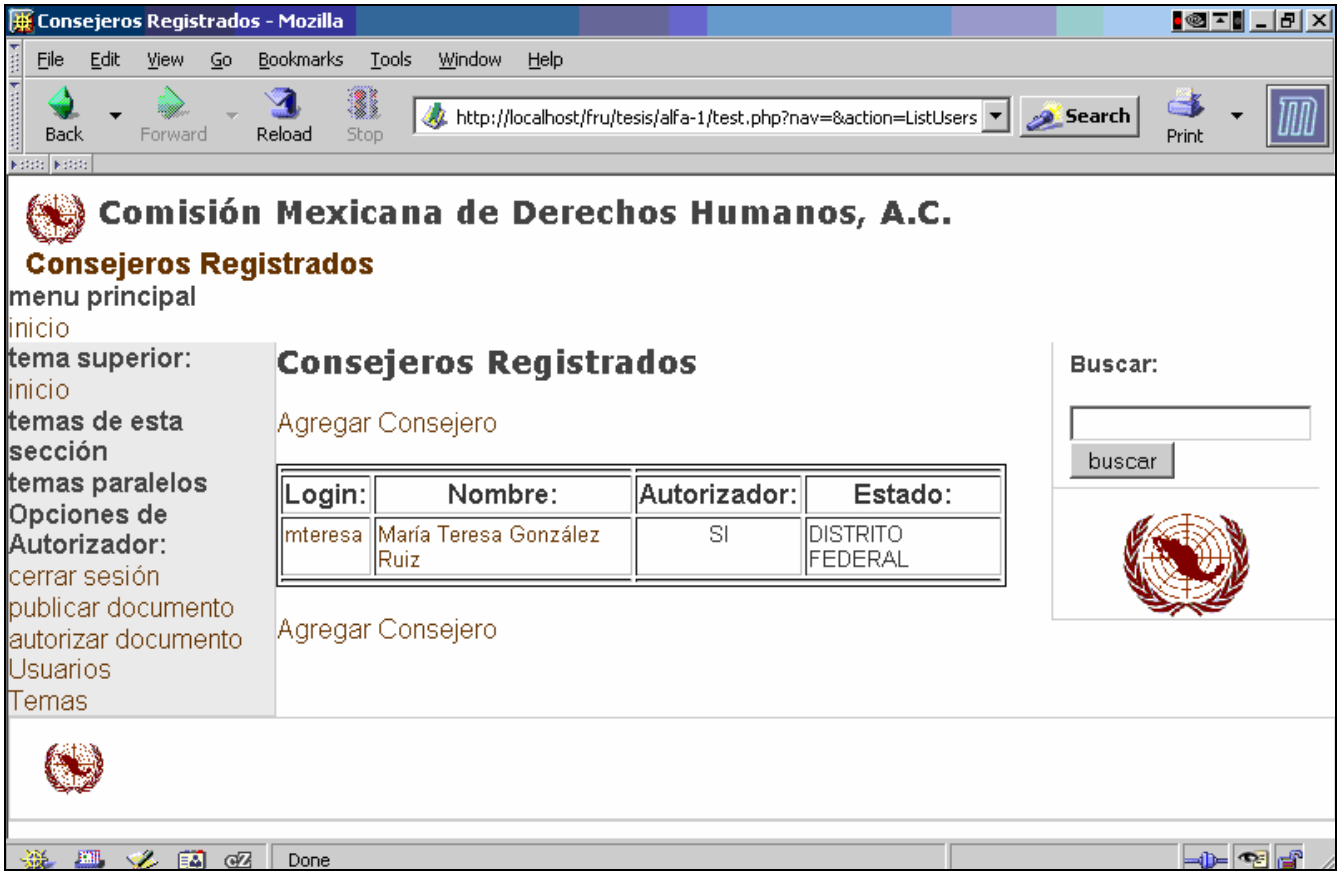

Fig. A-30: pantalla de usuarios

En esta pantalla aparece el consejero que se dio de alta. Se puede empezar a utilizar el sistema. La instalación ha finalizado.

B) GLOSARIO

En este glosario se reunieron términos importantes. Para mejor identificación se dividieron en tres categorías: Técnicos, De Negocio y Teóricos.

B.1 Técnicos

ArchivoClass
Representa un archivo que es una especie de documento

ArchivosClass
Colecciona objetos clase ArchivosClass

Authorize
Módulo que realiza el caso de uso de ‘autorizar documento’ (cambiar el estado de un documento) y realiza (en caso de ser necesario) la rotación automática.

CollectionClass
Sirve para manejar colecciones mediante arreglo de datos de cualquier tipo y de objetos

ConsejeroClass
Representa a un consejero(o a un autorizador)en el sistema

ConsejerosClass
Colección de objetos ConsejeroClass con persistencia en base de datos

ContentManagerClass
Clase que reune el contenido para visualizar y procesa el resultado del QueryManagerClass y da formato

ContentManagerIF
Interfaz para controlar el sistema externamente, es el sistema de navegación

DbCollectionClass
Esta clase es una colección de objetos con persistencia en base de datos

Delete
Módulo que elimina el documento del sistema

Determinador
Clase que determina que menus y que contenido se va a mostrar además del módulo a utilizarse.(análisis)

DisplayerClass
Genera y controla los objetos visuales de la página

DocumentoClass
Representa un documento en el sistema

DocumentosClass
Colección de DocumentoClass

Download
Módulo descarga un documento en el equipo del cliente considerando la seguridad

FileDownloadClass
Sirve para que el usuario descarque vía web archivos.

FileUploaderClass
Controla los archivos que se envían al servidor

FormatManagerClass
Clase que implementa la interfaz FormatManagerIF y realiza sus funciones. En particular Analizar los XML del menu y contenido para transformarlos y desplegarlos

FormatManagerIF
Interfaz del subsistema de general de usuario de la capa presentación de la información.

FormCheckClass
Valida formularios web. Esta clase funciona mediante reglas

FrmDelete
Módulo que muestra un formulario para confirmar si se desea eliminar un documento del sistema

FrmListADocument
Módulo que muestra un documento con toda su información y permite modificar su estado: público/privado, permanente/no permanente, aprobado/pendiente; también permite descargarlo o eliminarlo

FrmListDocuments
Módulo que lista los documentos del más reciente al más antiguo indicando el estado del documento y permitiendo su descarga o eliminación así como pasar a una vista detallada (FrmListADocument)

FrmLogin
Módulo que muestra el formulario de inicio de sesión, también muestra los errores en caso de que los datos de sesión sean incorrectos

FrmNewTopic
Módulo que permite capturar los datos relacionados con el tema nuevo.

FrmRegisterUser
Modulo que muestra el formulario de captura de los datos del consejero, también muestra los errores en caso de una captura incorrecta

FrmUploadFile
Módulo que muestra el formulario de captura de los datos del documento a publicar, también muestra los errores en caso de una captura incorrecta

GenericIF
Interfaz genérica abstracta

ListATopic
Módulo que muestra los datos referentes a un tema

ListAUser
Módulo que muestra los datos de un consejero en particular

ListTopics
Módulo que lista los temas ya existentes en el sistema.

ListUsers
Módulo que lista los consejeros registrados en el sistema con algunos de sus datos

Login
Módulo que realiza el caso de uso, es decir, inicia la sesión de usuario

Logout
Módulo que realiza el caso de uso, es decir, termina la sesión de usuario

MesClass
Representa el mes a partir del cual se deberán eliminar los documentos antiguos

messenger
Sirve para enviar mensajes con un título en la navegación.

MessesClass
Es la colección de objetos clase MesClass

MRMakerClass
Sirve para generar las respuestas de los módulos de manera sencilla

Navigate
Módulo que muestra los documentos disponibles en un determinado tema colocando el más reciente hasta arriba y mostrando la información relativa a cada documento

NewTopic
Módulo que realiza el caso de uso, es decir, crea un tema nuevo

NewUserFrmClass
Formulario para poder capturar los datos de un nuevo usuario

NewUserOperatorClass
Clase para operar nuevos usuarios

PluginManagerIF
Interfaz mediante la cual se crea una instancia del subsistema modules. Puede generar redirección, o producir un xml con el contenido.

QueryManagerClass
Realiza consultas a base de datos para obtener menus de acuerdo con el tema

RecordSetClass
Realiza consultas a base de datos

RegisterUser
Módulo que realiza el caso de uso, es decir, registra el consejero en el sistema

Sesiones
Clase que almacena información de las sesiones(análisis)

Session Manager
Clase que controla las sesiones

SessionClass
Controla las sesiones de usuarios

TemaClass
Representa un tema de navegación

TemasClass
Colecciona objetos clase TemaClass con persistencia mediante base de datos

Uploadfile
Módulo que realiza el caso de uso, es decir, registra el documento a publicar en el sistema y coloca el archivo en el servidor

UploadResult
Módulo que muestra el resultado de la operación

WebContentClass
Clase que almacena un contenido Web

WebContentSectionClass
Clase que almacena y despliega los elementos de la sección contenido de la página

WebFootSectionClass
Clase que alamacen ay despliega los elementos de la sección del pie de página

WebFormClass
Clase que almacena un formulario Web y lo despliega

WebFormfieldClass
Clase que almacena un botón de enviar datos (submit) de un formulario

WebFormInputClass
Control de entrada de tipo texto

WebFormSecretClass
Control de entrada de contraseñas

WebFormSelect1ItemClass
Control de entrada de selección de una opción entre varias

WebFormTextareaClass
Control de entrada de tipo activado/desactivado

WebItemClass
Clase abstracata de la que se derivan todas las demás.

WebLabelClass
Contiene una etiqueta de texto

WebMenuClass
Clase abstracta que almacena un menu

WebMenuClass
Clase que despliega un menu en forma de texto sin formato

WebMenuItemClass
Clase que almacena un elemento del menu

WebMenuSectionClass
Clase que almacena y despliega los elementos de la sección del menú

WebModelClass
Clase que almacena y controla el modelo

WebModelFieldClass
Clase que almacena un campo (variable) del modelo

Web-models
Familia de clases relacionadas entre si que permiten la construcción robusta de páginas con menús y formularios.

WebModelSubmissionClass
Clase que almacena un campo con valores de envío (submit) del formulario

WebNewsSeccionClass
Clase que almacena y despliega los elementos de la sección de noticias de la página

WebPageClas
Clase para almacenar controlar y desplegar páginas web

WebPageClass
Colecciona varios modelos y varias secciones

WebSectionClass
Clase abstracta que representa una sección de la página Web

WebSelect1ItemClass
Clase que almacena una opción de selección de un control de selección tipo WebForm

WebTabMenuClass
Clase que despliega un menu en forma vertical con formato

WebTextLabelClass
Agrupa a varias etiquetas de texto aplicando un mismo estilo

WebTopSectionClass
Clase que almacena y despliega los elementos de la sección superior de la página

YAFUClass
Recibe archivo enviados por los usuarios a través de formularios web y los guarda en el servidor.

B.2 De Negocio

archivo
Cualquier archivo (word, pdf, etc) que se quiera publicar

autorizador
Es un consejero con privilegios especiales para administrar la información del portal

autorizar
El autorizador revisa los documentos publicados y autoriza o rechaza los mismos, asignando la categoría de publico o privado.

avisos selectivos
son avisos, informes, invitaciones, etc. Que se hacen llegar a determinados grupos de contactos, que forman la Red de Relaciones Públicas

CMDH
Comisión Mexicana de Derechos Humanos

consejero
Son los consejeros directivos o consultivos, los cuales poseen una cuenta en el sistema

consejo consultivo
Asesora al presidente y al consejo directivo. Esta formada por los ex presidentes y personas designadas por la asamblea

consejo directivo
Realiza las funciones de dirección. Formado por uno o mas vicepresidentes, presidentes estatales un tesorero y un secretario

contenido
Es el resultado de cualquier módulo

controles
Cualquier elemento de un formulario con los que interactúa un usuario

documento
La información que se va a publicar

documento privado
Se publica solo para los consejeros

documento publico
Se publica para cualquier usuario del sistema

emisor
Es el rol que cumple cualquier consejero cuando publica un documento

enlace
Alguna dirección de internet

estilo
Es un formato que se le asigna a un elemento HTML

etiqueta
Cualquier texto al que se le aplica un estilo determinado

evento
Cualquier evento programado por la CMDH para realizar en el futuro

flujo de documentos
Es el proceso de controlar los documentos generados por los consejeros para mejorar la comunicación del consejo directivo

forma
Es un conjunto de controles que se relacionan con variables de un modelo y que permiten desplegar un formulario web.

iniciar sesión
Un usuario inicia su sección mediante un nombre (login) y una contraseña (password)

interfaz abstracta
Una clase abstracta que se usa como modelo para implementar interfaces. Debido a que en php no hay interfaces, se construyeron de este modo

keycode
Campo de la clase temas para hacer referencia al tema de manera más conceptual

línea base
En el otro glosario

lógica del negocio
Es una capa de diseño que contiene las realizaciones de los casos de uso.

menú
Entendido como objeto. Es capaz de almacenar la información menú. Los enlaces a los temas de navegación del sitio.

modelo
Es un conjunto de variables con sus respectivos valores. Una abstracción de un sistema cerrada semánticamente

navegar
Consiste en obtener información del sitio, como los documentos (contenido) y los temas en que se agrupan

OSC
Organización de la sociedad civil

publicar
El emisor envía un documento al sistema, el sistema coloca el documento en la base de datos en estado pendiente esperando ser autorizado.

Red de Relaciones Públicas
es una base de datos de grupos de interés con el fin de contactar a cierto tipo de personas para ciertas cosas

registrar nuevo usuario
Los usuarios son consejeros el autorizador es quien los registra, capturando sus datos en el sistema. Login y password son los datos que se envían al consejero por correo

rotación automática
Es el proceso mediante el cual se desplazan documentos viejos la ingresar documentos nuevos al sistema

sección
Se refiere a una parte visible de una página.

servicio de noticias
Consiste en una síntesis noticiera diaria con lo más destacado de la política nacional, la cual es enviada por correo electrónico a todos los socios

simpatizante
Personas con inclinación afectiva hacia la CMDH

tema
Los documentos estan agrupados en temas, estos temas los crea el autorizador.

B.3 Teóricos

acoplamiento
Agrupar dos piezas o sistemas, de manera que su funcionamiento combinado produzca el resultado conveniente

actor
Un conjunto coherente de roles que los usuarios de los casos de uso desempeñan cuando interaccionan con estos casos de uso

almacenamiento
Registrar información en la memoria, generalmente en archivos

análisis
Estudio, mediante técnicas informáticas, de los límites, características y posibles soluciones de un problema al que se aplica un tratamiento por ordenador.

arquitectura
Conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema.

artefacto
Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades; representa un área de responsabilidad, y es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento

ASP
Abreviatura de Active Server Pages. Páginas Activas de Servidor

atributo
Una propiedad con nombre de un clasificador que describe el rango de valores que las instancias de una propiedad pueden tomar

autenticidad
Certificación con que se testifica la identidad y verdad de algo.

base de datos
Colección de información organizada bajo una estructura en que el programa manejador de base de datos puede rápidamente procesar la información.

calidad
Propiedad o conjunto de propiedades que permite juzgar su valor

campo
Parte de un registro, los registros forman las tablas de las bases de datos

caso de uso
Una descripción de secuencias de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado

clase
Una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica

clasificador
Mecanismo que describe características estructurales y de comportamiento. Los clasificadores incluyen interfaces, clases, tipos de datos, componentes y nodos.

clave primaria
Campo con información corta que debe tener un valor único

cliente
Un clasificador que solicita servicio de otro clasificador. Otra definición en el otro glosario

codificación
Pasar a lenguaje de programación los diseños de la solución de un problema

cohesión
Capacidad de una entidad de mantener juntas sus partes

componente
Una parte física y reemplazable de un sistema que se ajusta a, y proporciona la realización de un conjunto de interfaces

configurar
Establecer los valores iniciales de un programa o sistema de cómputo para una aplicación

consistencia
Duración, estabilidad, solidez

control de acceso
Es un servicio de seguridad que autoriza a una entidad (usuario o sistema) a usar un determinado recurso

cookies
archivo de texto que guarda un mensaje. Utilizado por el navegador cuando recibe un mensaje de un servidor web. El navegador envía de vuelta este Archivo cada vez que se conecta a ese servidor

criptográfico
Arte de escribir con clave secreta o de un modo enigmático

CSS
Abreviatura de Cascading Style Sheets. Hojas de estilo en cascada

defecto
Anomalía del sistema, por ejemplo un síntoma de un error en el software descubierto durante las pruebas, o un problema descubierto durante una reunión de revisión

diagrama de secuencia
Un diagrama de interacción que hace énfasis en la ordenación temporal de los mensajes

Diagrama entidad-relación
Representación gráfica de datos y sus relaciones

diagramas de precedencia
Diagrama usado para identificar causas de un riesgo y para planear

diagramas tipo Ishikawa
Diagrama usado para identificar causas de un riesgo y para planear

diseño
Proyecto, plan

disponibilidad
La cualidad de poder tener el servicio cuando se necesite

distribución
Reparto del producto

documentación
Documento o conjunto de documentos que acreditan algo

Domxml
Abreviatura de Document Object Model XML. API propuesta por la W3C para manipular documentos XML.

DTD
Abreviatura de document type definition. Definición de tipo de documento. Se utiliza para definir la estructura de un archivo en XML.

estado
Una condición o situación durante la vida de un objeto la cual éste satisface alguna condición, lleva a cabo alguna actividad o espera algún evento

estereotiopo
Una extensión del vocabulario de UML, que permite la creación de nuevos tipos de bloques de construcción que se derivan de otros existentes pero que son específicos a un problema particular

flujos de trabajo
Realización de un caso de uso de negocio o parte de él. Puede describirse en términos de diagramas de actividad, que incluyen a los trabajadores participantes, las actividades que realizan y los artefactos que producen

formato de datos
La manera en que esta representada la información.

formulario web
Página web parecida a un cuestionario por medio de la cuál se pueden enviar datos al servidor

FTP
File Transfer Protocol. Protocolo de transferencia de archivos

gestión de reisgos
Definir y mantener los riesgos. A fin de reducirlos y atacarlos

GET
Comando del protocolo HTTP

HTML
Hyper Text Markup Langage. Lenguaje de marcas de hipertexto. Utilizado para la elaboración de páginas web

implementación
Poner en funcionamiento, aplicar métodos, medidas, etc. Para llevar a cabo una aplicación

inception
Primera etapa de RUP. Se ponen estados al proyecto de negocio. Se determina los recursos que se necesitaran para desarrollar y si es posible desarrollarlo.

incremental
Cualidad del proceso de desarrollo de software mediante la cual se agregan al sistema partes pequeñas y manejables.

instancia
Una manifestación concreta de una abstracción; una entidad sobre la que pueden aplicarse un conjunto de operaciones y que tiene un estado que almacena los efectos de las operaciones; un sinónimo de objeto.

interfaz
Una colección de operaciones que son utilizadas para especificar un servicio de una clase o de un componente

integridad
Cualidad por la cual se comprueba que no carece de sus partes

iteración (iterativo)
En el contexto de ciclo de vida del software, proceso que implica la gestión de una serie de versiones ejecutables

JSP
Java Server Pages

lenguajes de programación
Lenguajes utilizados para dar instrucciones a la computadora. Usualmente llamados de alto nivel

login
Nombre de usuario

MD5
Algoritmo para crear firmas electrónicas. Diseñado para usarse con maquinas de 32 bits

método
La implementación de una operación

módulo
Parte de un programa. Puede contener varias rutinas

no repudio
Técnicas para evitar que el usuario rechace o no acepte un trámite realizado en línea

normalización
Proceso para organizar información para minimizar redundancia.

objeto
Véase instancia

objeto persistente
Un objeto que existe después de que el proceso o hilo que lo creó ha dejado de existir

página
Un documento en el world wide web. Cada página web tiene un identificador único URL

paquete
Un mecanismo de propósito general para organizar elementos en grupos

password
Contraseña

patrón de desarrollo
Solución común a un problema común de un determinado contexto

performance
Medida de rendimiento. Velocidad de respuesta del sistema

PHP
Abreviatura de hypertext preprocessor. Preprocedador de hipertexto. Lenguaje de scripts del lado del servidor para crear páginas web dinámicas

plantilla
Modelo a partir del cual se obtiene algo

portal de internet
Sitio web que se ofrece un menú de recursos o servicios

POST
Comando del protocolo HTTP

privacidad
Ámbito que se tiene derecho a proteger de cualquier intromisión

proceso unificado de desarrollo (UML)
Metodología de desarrollo de software basado en UML.

producto
Sistema de software producido

programa fuente
Archivo que contiene el programa escrito en lenguaje de alto nivel. Las instrucciones del programa en su forma original

protocolo
Un formato de transmisión de datos entre dos dispositivos bajo aceptación.

prototipos
Un tipo, forma o instancia original que sirve de base o estándar para etapas posteriores. Un modelo a escala completa y usualmente modelo funcional de un producto nuevo o una nueva versión de un producto que ya existe.

proyecto
Esfuerzo de desarrollo para llevar un sistema lo largo de un ciclo de vida

pruebas
Medio por el cual se pretende mostrar la eficacia del sistema de software

pruebas de caja blanca
Prueba de la interacción interna entre los componentes del sistema

pruebas de caja negra
Prueba del comportamiento observable del sistema

pruebas de integración
Prueba en la que se verifica que los componentes interactúan entre sí.

pruebas de sistema
Prueba para verificar que el sistema funciona correctamente como un todo

pruebas de unidad
Prueba sobre una clase o método

red
Conjunto de dispositivos interconectados para compartir recursos

registro de dominio
Dentro de internet el dominio es definido por la dirección IP y debe ser registrado ante la autoridades de internet

requisito
Condición o capacidad que debe cumplir un sistema

riesgo
Variable de un proyecto que pone en peligro o impide su éxito.

robustez
Capacidad de una entidad para adaptarse al cambio

rol
El comportamiento específico de la semántica de una entidad que participa en un contexto particular

RSS
Abreviatura de Rich Site Summary. Resumen enriquecido del sitio.

script
Lita de comandos que son ejecutados sin la interacción del usuario. Se pueden hacer programas sencillos

Schema
Estructura de una sistema de base de datos. Describe en un lenguaje formal soportado por algún DNMS. Define las tablas, los campos en cada tabla y las relaciones entre campos y tablas.

seguridad
Técnicas para asegurase que la información almacenada en una computadora no puede ser leída por individuos sin autorización

semántica
Significado de las palabras de un lenguaje

servicio
Subsistema independiente que realiza una función específica

servidor
Una computadora o dispositivo en una red que administra recursos de la red

sistema
Una colección de subsistemas organizados para llevar a cabo un propósito específico y descritos por un conjunto de modelos, posiblemente desde distintos puntos de vista

sitio
Una localidad en la web. Contiene páginas web

SQL
Abreviatura de Structured Query Language. Lenguaje de consultas estructurado. Estándar para solicitar información de una base de datos

STUB
Una rutina que no hace otra cosa que la declaración de si misma y los parámetros que acepta. Usada para contener rutinas que necesitan ser desarrolladas

suplantación
Ocupar con malas artes el lugar de alguien

tabla
Arreglo de información en filas y columnas

tablas inteligentes
Es una tabla que ofrece funcionalidad adicional sobre búsqueda, ordenamiento y presentación de la información

UML
Abreviatura de Unified Modelign Language. Lenguaje de modelado unificado. Lenguaje notacional de propósito general para especificar y visualizar software complejo, especialmente grande. Proyectos orientados a objetos.

URI
Abreviatura de Uniform Resource Idenrifier. Identificador de recurso uniforme

URL
Abreviatura de Uniform Resource Locator. Localizador de recurso uniforme. La dirección global de documentos y otros recursos en la web

versión
Conjunto de artefactos relativamente completo y consistente entregado a un usuario interno o externo.

vista de arquitectura
Una proyección de un modelo, la cual es vista desde una perspectiva determinada o punto estratégico y que omite las entidades que no son relevantes para esta perspectiva. Otra definición en el otro glosario

W3C
Abreviatura de World Wide Web Consortium. Consorcio red de área mundial. Consorcio para desarrollar estándares para la web

Web hosting
Almacenamiento de documentos para publicarse en la web

XForms
Tecnología propuesta por W3C basada en xml para hacer formularios web avanzados. Aun no soportada por los navegadores

XHTML
Abreviatura de Extensible Hipertext Markup Language. Extensible lenguaje de marcas de hipertexto. Un híbrido entre HTML y XML específicamente diseñado para dispositivos de despliege en red

XML
Abreviatura de Extensible Markup Language. Lenguaje de marcas extensible. Una especificación desarrollada por el W3C. Para permitir diseñar documentos web con sus propias tags personalizadas.

C. Referencias

Ambler, S (2005), *Advanced XML? No, Just Realistic XML*, http://www.agiledata.org/essays/advancedXML.html

Ambler, S (2005), *Data Modeling*, http://www.agiledata.org/essays/dataModeling101.html

Ambler, S (2005), *Mapping Objects to Relational Databases: O/R Mapping In Detail*, http://www.agiledata.org/essays/mappingObjects.html

Ambler, S (2005), *The Object-Relational Impedance Mismatch*, http://www.agiledata.org/essays/impedanceMismatch.html

Ambler, S (2005), *Why Data Models Shouldn't Drive Object Models (And Vice Versa)*, http://www.agiledata.org/essays/drivingForces.html

Cetus Team (2004), *Databases: Mapping Objects to Relations*, http://www.cetus-links.org/oo_db_systems_3.html

Cetus Team (2004), *Links on Objects & Components*, http://www.cetus-links.org/

Chen, PP (1976) en *Scientific Literature Digital Library* (2005) http://citeseer.csail.mit.edu/context/76/0

Comisión Federal de Telecomunicaciones (2005) http://www.cofetel.gob.mx, para estadísticas sobre usuarios de internet en México

Comisión Mexicana de Derechos Humanos (¿?) *Estatutos Sociales*

Craig, L, Philippe, K, Kurt, B (2001), *How to Fail with the Rational Unified Process*, Ratonal Software

Ericsson, HE, Pender, M (2000), *Business Modeling With Uml Business Patterns at Work: Business Patterns at Work*; John Wiley & Sons Inc

Gamma, E, Helm, R, Johnson, R, Vlissides, J (1994), *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley; 1994

GNU General Public License (1991), http://www.gnu.org/copyleft/gpl.html (2)

Holowczak, R (2002), *Entity Relationship Modeling and Normalization*, Database Management Systems II, Zicklin School of Business, Baruch College City, University of New York http://cisnet.baruch.cuny.edu/holowczak/classes/9440/entityrelationship/

Jacobson, I, Booch, G, Rumbaugh, J (2000), *El Proceso Unificado de Desarrollo de Software*, Addison Wesley, Madrid, 438 pp

Kroll, P, Kruchten, P (2003), *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison-Wesley

Leffingwell, D, Widrig, D (2000), *Managing Software Requirements: A Unified Approach*, Addison-Wesley

Lucena, MC (1999), *Criptografía y Seguridad en Computadores*, Universidad de Jaén (2)

Lynch, P, Horton, S (2002) *Web Style Guide* (2), www.webstyleguide.com

Nielsen, J (1999), *Designing Web Usability: The Practice of Simplicity*, New Riders Publishing (7)

Pressman, RS (2001), *Ingeniería del Software*, Mc Graw Hill (5)

Project Management Institute (2000), *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, Newton Square, Pennsylvania, USA

Rivest, R (1992), *The MD5 Message-Digest Algorithm*, MIT Laboratory for Computer Science and RSA Data Security, Inc., http://rfc.net/rfc1321.html

Singh, I, Stearns, B, Johnson, M (2002), *Designing Enterprise Applications with the J2EE™ Platform,* Addison-Wesley (2)

Software Engineering Institute (2005), *How Do You Define Software Architecture*, Carnegie Mellon University, http://www.sei.cmu.edu/architecture/definitions.html

Swartz, A (2000), *RDF Site Summary (RSS) 1.0*, http://web.resource.org/rss/1.0/spec

Trowbridge, D, Cunningham, W, Evans, W, Brader , L (2005), *PatternShare Community*, Microsoft Corp., http://patternshare.org/

UML Resource Page (2005) http://www.uml.org/

World Wide Web Consortium (2005), *Cascading Style Sheets Home Page*, http://www.w3.org/Style/CSS/

World Wide Web Consortium (2005), *Guide to the W3C XML Specification ("XMLspec") DTD* (2.1), http://www.w3.org/XML/1998/06/xmlspec-report-v21.htm

World Wide Web Consortium (2005), *XForms - The Next Generation of Web Forms*, http://www.w3.org/MarkUp/Forms/

Yourdon, E (1975), *Techiques of program structure and design*, Prentice Hall