UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

REINGENIERÍA DEL SITIO WEB DEL CEAQ DE LA FACULTAD DE MEDICINA

T E S I S

PARA OBTENER EL GRADO DE
INGENIERO EN COMPUTACIÓN

P R E S E N T A N

FERNANDO ALFREDO GIRÓN JIMÉNEZ
IVÁN DE LA CRUZ SÁNCHEZ SÁNCHEZ

DIRECTORA: ING. GABRIELA BETZABÉ LIZÁRRAGA RAMÍREZ

MÉXICO, DISTRITO FEDERAL 2010

RECONOCIMIENTO

A la muy estimada Ing. Gabriela Betzabé Lizárraga Ramírez, directora de nuestro trabajo de tesis, que, de manera altruista, nos proporcionó su apoyo y nos regaló su valioso tiempo durante el desarrollo y ejecución de este proyecto. Gracias por ser parte importante en este trabajo y ser el cimiento en esta etapa de nuestros estudios profesionales.

RECONOCIMIENTO

A la Universidad Nacional Autónoma de México, nuestra alma mater, a ti nos debemos, y con orgullo te llevamos en la sangre, en todo lo que hagamos en nuestra vida siempre vivirás en nuestros corazones. Gracias por tu apoyo incondicional. Por mi raza hablará el espíritu.

RECONOCIMIENTO

A la Facultad de Ingeniería, por abrigarnos en su cobijo, ofrecernos sin interés sus instalaciones y su gente, para la realización de nuestros estudios profesionales y el desarrollo de este trabajo de tesis. No puede existir mejor escuela que tú, por ti somos y con orgullo llevaremos tu nombre por la vida, pues eres la más grande de todas. Gracias.

RECONOCIMIENTO

A nuestros maestros, que nos forjaron, nos acercaron al conocimiento y nos hicieron desarrollar nuestro libre pensamiento, nos enseñaron a caer y a levantarnos, ellos fueron los eslabones más fuertes en nuestra vida, y gracias a ustedes hoy estamos aquí, presentando un proyecto de tesis como resultado de sus horas de dedicación a nosotros.

Reconocemos de manera especial a nuestros sinodales, que nos brindaron su valioso tiempo y sus conocimientos en este proyecto final de tesis, reconocemos a:

- Ing. José Enrique Larios Canale (Presidente)
- Ing. Gabriela Betzabé Lizárraga Ramírez (Vocal)
- Dra. María Cristina León González (Secretario)
- Ing. Marian Aburto Estébanez (Primer Suplente)
- Ing. Salvador Pérez Carcaño (Segundo Suplente).

AGRADECIMIENTO

A mis padres y a Dios quienes me dieron la vida, y que con su amor me acompañan el día de hoy a presentar este proyecto de tesis, por ustedes soy el hombre que soy ahora, y espero siempre ser un orgullo para ustedes.

A mi hermano Carlitos que desde un lugar muy especial siempre nos ha cuidado y lo seguirá haciendo hasta el día que nos encontremos.

A mi familia, que siempre ha estado en mi vida para darme su apoyo y cariño incondicional. A estas dos grandes familias, que han sido inspiración (por sus grados académicos) y motivación (con sus grandes consejos y cariño).

Honor a quien honor merece, y lo merecen ustedes, quienes salvaron mi vida, las personas más maravillosas que cuidan de nosotros, y aunque uno de ellos ya no esté con nosotros físicamente, sabemos que su espíritu vive entre nosotros como el ángel que era en vida. Honor y gratitud a ustedes y a sus hijos, por ustedes estamos aquí y no existen palabras para agradecerles todo lo que han hecho por nosotros.

Te agradezco a ti, con quien aprendí a leer y escribir, con quien crecí, y con quien me acerque a los sistemas por primera vez. Por los años que hemos vivido y los que nos faltan por vivir, a ti y tu familia.

Te agradezco a ti, que me ofreciste la primera oportunidad de acercarme a una computadora y a ti, que apoyaste a mis padres en la parte académica para que ellos me lo transmitieran a mí, gracias a ustedes decidí tomar este camino que hoy finaliza.

Para ti, mi hermana incondicional, nos conocimos estudiantes y nos hicimos guerreros, Gracias por cumplir tu obligación de ser mí guía. Hoy me presentó a ti diciéndote: "*Everything that has a beginning has an end*". Llegue al final del camino y como siempre, estas a mi lado.

A ustedes, el diez perfecto, la unidad y a ustedes, que cada vez que nos reunimos anhelo regresar a esos días donde las conocí, con ustedes jugué y me discutí, pero también aprendí a admirarlos/as y quererlos/as, no éramos muchos, pero si los suficientes para sentirme muy querido y afortunado de conocerlos/as.

A ustedes, a quienes considero mi familia, ustedes que me conocieron estudiante, que me vieron caer y que me ayudaron a levantarme. Por su invaluable apoyo es por lo que el día de hoy estoy aquí. Nunca he dudado de su sincera amistad y guardan un lugar muy especial en este trabajo y en mi corazón. Uno de ustedes me dijo en su mejor momento, "Te consideramos nuestro hermano". Yo les puedo expresar que de esa misma forma los considero yo, una extensión de mí, y hoy me presentó aquí, esperando ser un orgullo para ustedes.

A ti, que afortunadamente te cruzaste en mi vida cuando inicie el camino que hoy finalizó, fuiste mi hermana y me brindas todo tu cariño sin merecerlo. Me apoyaste en las buenas y en las malas, me sigues enseñando muchas cosas y me ayudas a recordar lo que he aprendido en estos años. Tú me has dado un espacio de ti, eres parte de este

proyecto gracias a tu motivación y cariño, lo cual se puede notar en cada línea de este trabajo. Gracias por estar a mi lado el día de hoy.

A ustedes, con quien pase toda mi carrera. Cada uno estuvo en un momento clave de mi trayectoria, tú me presentaste (con tu carisma) a mi directora de tesis, contigo trabaje durante este proyecto (no puedo tener mejor compañero y amigo que tu), y a ti que fuiste uno de los que revisó (con tu sincera amistad) y dio sus observaciones sobre este trabajo.

A ustedes, que desafortunadamente tuvimos que separarnos de durante los años que estudiamos juntos, pero que siempre seguimos juntos gracias a la amistad que nos une. A ti que me invitaste a tu examen profesional y a ti que sé que pronto me invitarás.

A ustedes y a su familia, las dos más pequeñas, que me muestran su admiración y gratitud sin merecerla. Mi agradecimiento para ustedes esperando pronto ver la conclusión de sus estudios, se han ganado mi orgullo y un lugar especial en mi corazón.

A ti, que me enseñas y me sigues enseñando a ser responsable de mis acciones, que me enseñas a ofrecer compromisos y cumplirlos. A ti, que nos conocimos en una etapa muy importante de nuestras vidas, te agradezco por los buenos consejos que me ofreces día a día, por las buenas que pasamos juntos y por los enfrentamientos que tenemos, siempre aprendo algo nuevo contigo. A ti, que me ofreces tu experiencia y tu buen consejo, agradezco que me ofreces antes que nada tu gran amistad, hoy terminó yo, pero espero pronto estar cuando termines tú.

A ti, que te reencontré y a ustedes que los encontré, mi más grande agradecimiento por su apoyo y amistad, en las buenas y en las malas siempre están ahí. "Que nos queda", diría un de ellos, ahora puedo decirle: "*Seguir disfrutando el tiempo que pasamos juntos para hacer más sólida nuestra amistad*".

Gracias,

Fernando Alfredo Girón Jiménez

AGRADECIMIENTOS

Agradezco a Dios por permitirme vivir en este tiempo y compartirlo con personas maravillosas que me han apoyado y me han permitido llegar a este punto en mi vida personal y profesional.

Agradezco a mis padres por todo su apoyo, su amor y comprensión sin ellos yo no estaría aquí... gracias por todo. Los amo.

Agradezco a mis hermanos por estar conmigo siempre, en especial a mi hermana Yuliana, eres parte muy importante en mi vida. Gracias por estar a mi lado.

Doy gracias a la vida por permitirme tener estos maravillosos amigos ellos han sido alegría en mi vida, apoyo y motivación a seguir adelante, gracias amigos por todo. Gracias Fer por toda tu ayuda a lo largo de la carrera y quien mejor que tú para ser mi compañero en este trabajo.

Agradezco mucho a aquellos que me acompañaron al inicio de este camino y que ya no se encuentran conmigo... siempre los tengo presentes y los llevo en mis pensamientos.

Y claro como no agradecer a mi querida UNAM en ti he aprendido tantas cosas y parte de los mejores recuerdos de mi vida hasta el momento los he pasado en ti. Gracias por todos esos buenos momentos.

Hay tantos a los cuales dar gracias por haberme acompañado en lo que llevo de vida que me faltan palabras y la memoria no es lo suficientemente buena para nombrarlos a todos... pero gracias a todos ellos por haber formado parte de mi vida.

Gracias,

Iván de la Cruz Sánchez Sánchez.

Índice

Capítulo I. Introducción

Justificación

La reingeniería del sitio web y el desarrollo de un sistema para la distribución de material multimedia y noticias para el Centro de Enseñanza y Adiestramiento Quirúrgico (CEAQ) de la Facultad de Medicina tienen como fin, aportar nuestros conocimientos para diseñar procesos que acerquen a la comunidad de esta facultad a las investigaciones y publicaciones de este centro y así aportar nuevas herramientas de estudio utilizando la infraestructura tecnológica con la que cuenta nuestra universidad.

Adicionalmente de proporcionar un nuevo servicio a este centro de investigación, dentro del proceso que se utilizará y de acuerdo al tamaño del sistema que se está diseñado, se propondrá un nuevo modelo para el ciclo de vida del software, utilizando los beneficios de 2 metodologías ya existentes de forma combinada (Proceso Unificado de Rational – RUP y Programación Extrema –XP) que se detallan más adelante.

La problemática actual del centro es que genera una gran cantidad de información, pero ésta no puede ser distribuida de forma correcta a su comunidad. La idea es proporcionar herramientas y canales de distribución a esta información de tal forma que se acorten distancias entre el centro y la comunidad de estudiantes y a su vez generar intereses hacia los trabajos que se están realizando.

Se ha elegido como canal de distribución Internet debido a su gran crecimiento en el país, tan solo basta con ver el comparativo de 2007 y 2008 publicadas en el sitio web de la **AMIPCI (Asociación Mexicana de Internet)**[1], de donde podemos destacar los siguientes puntos:

[1] Hábitos de los usuarios de Internet en 2007 y Hábitos de los usuarios de Internet en 2009

- En México, existían 23.7 millones de internautas en el 2007, en 2008 existen 27.6 millones, teniendo un crecimiento de 16.4% en un año.
- La tasa de penetración nacional de Internet en personas mayores a los 6 años aumentó del 25.6% al 29.7%
- En un año, se incrementó el número de computadoras con acceso a Internet, de 8.7 millones a 11.3 millones, esto representa el 62% de equipos de cómputo que existen en nuestro país.
- Existen 3.5 computadoras por cada 10 hogares.
- En 2007 el 78% de los equipos conectados a internet utilizan banda ancha, en 2008 los usuarios de banda ancha representan 93% de los usuarios conectados.
- Actualmente el 63% de la población entre los 12 y 19 años son usuarios de Internet. Esto nos indica que en los próximos años, las aplicaciones que desarrollemos para los centros de investigación de la UNAM, llegarán a esta población.
- El hogar y el lugar de estudio representan el 59% de puntos de acceso a Internet.

Dentro del análisis, se obtuvieron datos sobre los hábitos de las personas que utilizan internet en México, dentro de esos podemos resaltar los siguientes:

- El tiempo promedio de conexión al día en nuestro país es de 2 horas y 54 minutos, posicionándose como el tercer medio en tiempo de exposición, tan solo detrás de la televisión y el radio.
- Los usuarios de internet se enfocan en medios de entretenimiento multimedia (música, videos y juegos online).
- El medio se democratiza: el mayor incremento en la penetración de usuarios de internet se ha dado en el nivel socioeconómico D+[2] en el último año.

[2] Clase media baja

De esto podemos extraer datos muy interesantes, el primero es el rápido crecimiento que está teniendo internet en la población mexicana y el alto porcentaje de usuarios entre los 12 y 34 años; siendo éste nuestro objetivo principal debido a que por sus edades, una parte de esta población está realizando estudios de licenciatura o de posgrado, o bien, en los próximos años los estarán realizando.

En la parte de los hábitos encontramos que tenemos una población variada en cuanto a sus gustos, sin embargo, podemos identificar que sus principales objetivos son:

1. Leer noticias
2. Jugar en línea
3. Ver contenido multimedia.

Considerando las facilidades de acceso al medio, el rápido incremento de los usuarios y los hábitos dentro de este canal, determinamos que internet era el mejor medio para poder proveer a los alumnos y académicos de los servicios y el material que genera este centro.

Se ha tomado como eje principal la aplicación de nuevas tecnologías hacia la distribución de este material generado y se ha propuesto una nueva infraestructura tecnológica basada en el tipo de servicio que proveerá el centro hacia sus usuarios. Actualmente el centro cuenta con un espacio en el servidor web de la Facultad de Medicina donde se alberga una aplicación desarrollada con el software Macromedia Flash con información, imágenes y sitios de interés.

El proceso aplicado a la solución del problema lo hemos descrito en la figura 1, donde podemos ver los pasos que seguimos para la realización de este sistema.

Figura 1. Proceso de solución

Estos pasos indican el seguimiento de los puntos que se toman para la realización de esta tesis y cada uno de los recuadros indica un proceso, en algunos pasos un proceso cerrado y en otros un proceso abierto. Esto se hace con el fin de poder realizar una planeación sobre cada punto del desarrollo para tener un avance constante y sin desviarnos del objetivo principal, el cual es la elaboración de una solución para este centro de investigación.

El primer paso nos indica el **Análisis de la situación actual,** esto es, saber perfectamente cómo son sus sistemas de distribución actuales y su sitio web.

Actualmente el centro cuenta con un sitio web, el cual fue elaborado con Macromedia Flash en Octubre de 2006; éste cuenta con siete botones alrededor del logo del centro que nos llevan a la misma escena, la cual nos presenta cuatro nuevos botones (siendo un submenú de las características principales). Cada uno de éstos nos manda a otra escena, todo dentro del mismo archivo flash. Es un diseño muy sencillo y el tamaño del archivo es de 8.38 MB, que con una conexión de banda ancha tardó aproximadamente más de 1 minuto en iniciar (a 256 Kbps), lo cual no es muy recomendable considerando el tiempo de los usuarios.

En cuanto a la **Infraestructura Tecnológica**, la Facultad de Medicina proporciona al centro un espacio dentro de su servidor web para poder albergar un sitio, sin embargo, se propuso utilizar servidores independientes debido al contenido multimedia del sistema.

El hecho de ser equipos independientes, permitió definir las tecnologías que se utilizarían para el proceso de desarrollo de la aplicación, logrando así proponer aquellas donde tuviéramos mayor experiencia y mayor control.

El proceso para analizar la situación actual se puede definir gráficamente en la figura 2.

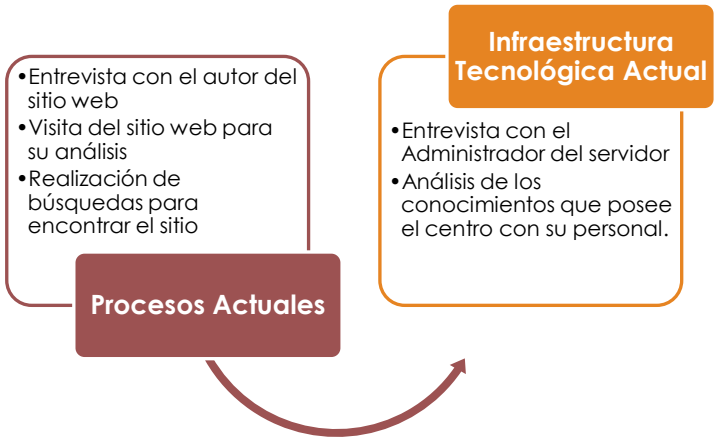

Figura 2. Situación Actual

Aquí podemos ver que el camino constó de 2 módulos; el primer paso fue entrevistarnos con el encargado de generar el material tridimensional, esto con el fin de conocer el trabajo del centro y como utilizan una página web para mostrar información valiosa para la comunidad.

La entrevista con el administrador del servidor fue vía telefónica, con la cuál obtuvimos datos muy precisos sobre la tecnología que utiliza, el límite de almacenamiento, el proceso de publicación y el motor de bases de datos que

nos proporcionan, sin embargo en este punto fue donde se realizó la propuesta sobre nueva infraestructura tecnológica para no interferir con los procesos actuales de la Facultad de Medicina.

Las actualizaciones han sido muy difíciles porque el centro no cuenta con personas que posean conocimientos sólidos de lenguajes de programación y los investigadores no cuenta con interfaces que permitan el ingreso y la actualización constante del material que generan.

Sin embargo, el centro cuenta con personas de servicio social de carreras relacionadas con sistemas para su proceso de creación de contenido multimedia. Por lo que la solución propuesta tiene que contener una interfaz para que los investigadores puedan crear material y el personal de apoyo de sistemas pueda realizar actualizaciones de forma fácil, sin que esto interfiera con la labor principal del centro.

Una vez que conocemos cual es su situación actual, podemos pasar al levantamiento de los requerimientos, siendo esta etapa una de las más importantes debido a que será la que defina el alcance del sistema y la complejidad del mismo.

El proceso del análisis de requerimientos lo hemos definido como un ciclo cerrado que nos permitirá conocer las necesidades de los usuarios, analizar los requerimientos, transformarlos en módulos del sistema, documentar los cambios necesarios y volver al conocer nuevas necesidades o dudas respecto a lo que se pretende con cada nueva funcionalidad.

El proceso se representa gráficamente a continuación:

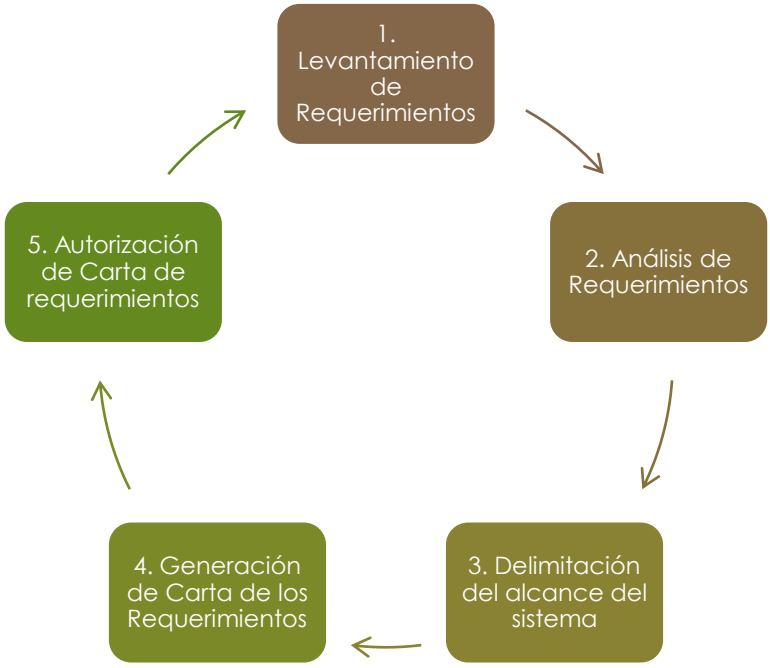

Figura 3. Proceso de levantamiento de requerimientos

Aquí podemos ver que la primer parte será donde nos reunamos con los usuarios, a platicar sobre lo que esperan del sistema, de esas pláticas generamos unos documentos denominados minutas, donde se anota todo lo platicado en esa junta.

De ahí, comenzamos a identificar las necesidades, las analizamos en función de la realidad, y los aterrizamos a verdaderos requerimientos de un sistema. Para esto tuvimos que contemplar su situación actual y el futuro del sistema.

Del análisis tuvimos una primera impresión sobre el alcance del sistema y pudimos delimitarlo, con el fin de no realizar módulos que no sean propios del planteamiento original y así no generar atrasos en los tiempos de entrega o un sistema inestable.

De estos pasos generamos una carta de requerimientos donde planteamos las necesidades del sistema tal y como las entendimos, apoyándonos en las minutas y entrevistas que realizamos. El siguiente paso realmente es una condicional para

el ciclo, si la carta es autorizada por el centro, entonces el ciclo termina ahí, en caso contrario, se volverá a iniciar agregando o quitando requerimientos en función al desacuerdo que hubiera con la carta.

Cabe mencionar que si el desacuerdo está en una nueva funcionalidad que no se encontraba contemplada en la primera versión o en la mejora que se está realizando, es descartada para versiones posteriores, con el fin de evitar quedarnos en el proceso de análisis de requerimientos indefinidamente.

Figura 4. Pirámide de Documentación

Para lograr un exitoso segundo paso, administrativamente vamos a definir un modelo guía que nos proporcione un camino de validación correcto y nos permita jerarquizar la importancia de cada una de las tareas, con el fin de aplicar correctamente nuestros esfuerzos. En esta segunda etapa tenemos como objetivo conocer el negocio y generar a partir del proceso de levantamiento de requerimientos un modelo real del negocio, con todos los procesos que intervengan en la distribución de información.

Cabe mencionar que por ser un proceso de reingeniería tenemos que cubrir las nuevas necesidades del centro considerando lo que ya se hace actualmente.

Una vez definidos los procesos de negocio tenemos que generar los documentos que avalen el trabajo e identificar a los actores del sistema, lo obtenido en este paso será lo que defina todo el proyecto y garantice que los resultados obtenidos cumplen con los objetivos principales.

El proceso se ilustra en la figura 4 y tiene la forma de una pirámide, como una forma de representar la importancia de cada etapa del proceso, el ciclo para el levantamiento de requerimientos forma la base de la pirámide debido a que es la tarea más importante y fallar en esta etapa debilitará el resultado final del sistema.

El modelo también muestra cómo, un error al entender el modelo del negocio podría provocar que la reingeniería no cumpla su objetivo principal y deje de ser una solución completa. Esto no quiere decir que la generación de la documentación no sea importante, más bien, nos permite conocer perfectamente la jerarquía del flujo.

Una vez que concluimos y validamos el segundo paso, podemos iniciar un análisis detallado de la siguiente etapa.

Para la **etapa del diseño**, el tercer paso del método de solución, se ha considerado que la forma de realizarlo se puede reflejar en un modelo, donde cada una de las cosas que hagamos, formaran parte de una estructura más compleja.

Lo que son los diagramas y modelos, que son extraídos de la metodología RUP serán la cúspide de esta etapa, pero sus cimientos serán:

- El Modelo de Negocio, donde podremos definir las reglas que debe cumplir el sistema de acuerdo a como lo está pidiendo el centro.
- El Análisis del flujo de la información, donde se determinará la forma en que se obtendrá, almacenará y distribuirá.

- Planteamiento de objetivos, que será la etapa donde tendremos que evaluar si el diseño cumple con los requerimientos generados en la fase anterior y los objetivos de esta etapa.

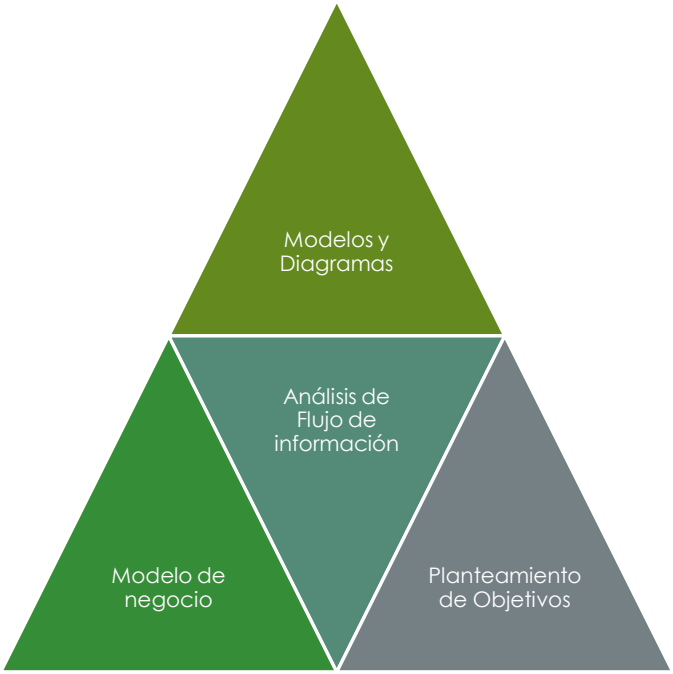

Figura 5. Etapa de diseño de la solución

En esta etapa comenzaremos a ver los primeros diagramas que serán implementados para darle funcionalidad al sistema. Estos son el resultado del entendimiento del modelo de negocio, el flujo de información y planteamiento de objetivos. Serán interpretados y codificados en lenguajes de programación buscando que la información sea confiable, íntegra y se distribuya correctamente por los canales adecuados.

Para la etapa **de implementación**, vamos a enlazar 3 elementos, los diagramas y modelos de la metodología RUP, las buenas prácticas de programación de la metodología XP y un paradigma de programación llamado Programación por capas. La idea es tener una aplicación estable y que cumpla con las necesidades del centro a través de este proceso de reingeniería.

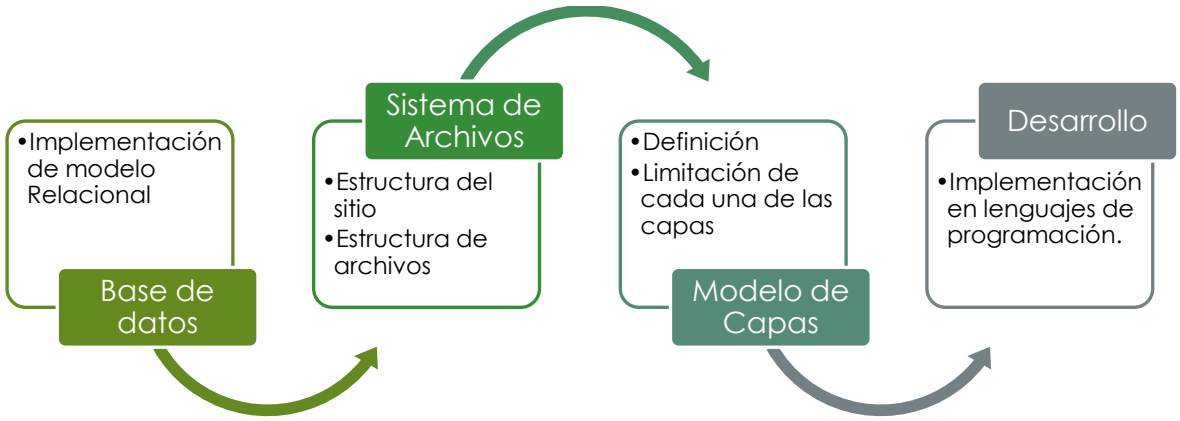

Figura 6. Aplicación de metodologías

El resultado de la idea de unir las metodologías RUP[3] y XP[4] ha sido probado por IBM y se ha descubierto que es ideal cuando el sistema tiene un tamaño pequeño y el grupo de desarrolladores son muy pocos; esto debido a que una de las prácticas dice que la etapa de la programación tiene que hacerse en parejas buscando crear mejores líneas de código.

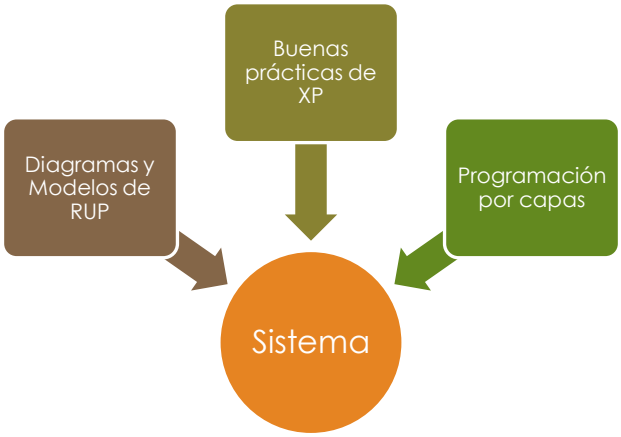

Figura 7. Etapa de Implementación

La implementación de la base de datos y el sistema de archivos constituyen el núcleo de la solución, pues ellas forman parte del Sistema de Información, el cual almacenará y proveerá la información procesada a los usuarios del centro. Un buen modelado en el sistema de información promoverá que la aplicación sea

[3] Rational Unified Process. Metodología de desarrollo de software utilizando lenguaje UML
[4] XP. *eXtreme Programming*. Metodología de desarrollo de software ágil.

actualizable en el futuro con nuevas técnicas de explotación de datos, siendo cada vez, una estructura más compleja para proveer, pero conservando su fácil entendimiento gracias a la documentación.

La etapa de la implementación traerá como resultado la primera versión del sistema que deberá cubrir los objetivos que se plantearon al principio y será el resultado del proceso de reingeniería.

Por último la etapa de evaluación incluye la migración del servidor de pruebas al servidor de producción, sin embargo, existirá un proceso previo donde se creará un ambiente productivo y se pondrá a prueba el sistema completo, para verificar que cumpla con todos los estándares planteados a lo largo del proceso y que cumpla el total de los requerimientos iniciales.

Una vez que se ha superado esta etapa se proseguirá a migrar la aplicación al servidor de producción y será monitoreada para verificar su desempeño en un ambiente real. Una vez que haya sido probada y verificada por los usuarios internos del centro, será liberada a toda la comunidad que así lo requiera y con esto daremos fin al desarrollo de la versión 1.0 del sistema y al proyecto de reingeniería.

El último paso será la puesta en operación del sistema y la evaluación por parte de los administradores sobre la nueva herramienta para distribuir sus investigaciones y trabajo generado.

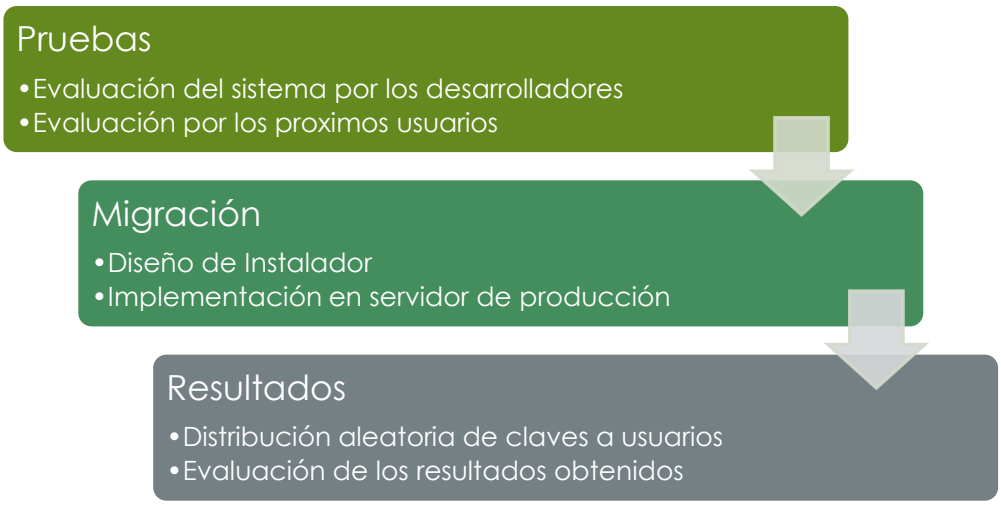

Figura 8. Proceso de Pruebas, Migración y Resultados

Definición y justificación de la metodología

Las metodologías para el desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y ayudas a la documentación para el desarrollo de productos de software.

Ian Sommerville lo define en el libro *Ingeniería de software* de la siguiente manera:

"Los métodos son formas organizadas de producir software. Incluyen sugerencias para el proceso que se debe seguir, la notación que se va a utilizar, los modelos del sistema que hay que desarrollar y las reglas que gobiernan estos modelos y las pautas de diseño"

En otras palabras, la metodología es como una receta de cocina en la que se va indicando todas las actividades que deben realizarse para obtener el producto de software deseado desde los puntos de vista de objetivos de negocio, costos, funcionalidad, sencillez y capacidad de soporte. También indica el papel que debe tener en el desarrollo de las actividades cada una de las personas que participan en el proyecto.

El uso de la metodología correcta nos ayuda a evitar problemas como:

- Resultados finales impredecibles
- Detección tardía de errores
- No controlar lo que está sucediendo en el proyecto
- Introducción de nuevas herramientas que afecten perjudicialmente al proceso
- Cambios en la organización que afecte al proceso
- Resultados distintos con nuevas clases de productos

Para el desarrollo de este proyecto hemos decidido utilizar características de 2 metodologías, Proceso Unificado de Rational y Programación Extrema, las cuales se describen a continuación:

Proceso Unificado de Rational (Rational Unified Process, RUP)

El Proceso Unificado de Rational (RUP en inglés) es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) forman la metodología más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Es un proceso iterativo, el cual divide en cuatro fases el proceso de desarrollo de software. Estas son:

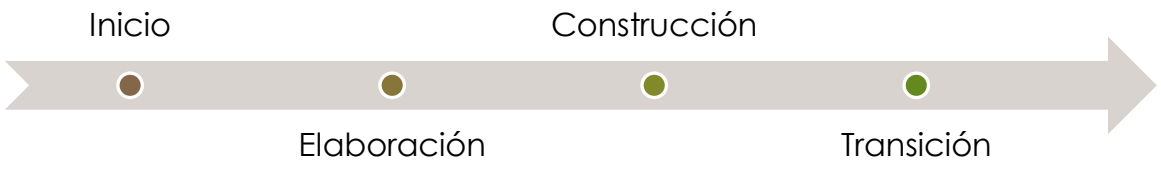

Figura 9. Fases del Proceso Unificado de Rational

1. **Inicio**. El objetivo de esta fase es determinar si vale la pena desarrollar el producto, en otras palabras si es económicamente viable el producto de software propuesto.

2. **Elaboración**. Los objetivos de esta fase son desarrollar una comprensión del dominio del problema, afinar los requerimientos iniciales, establecer un marco de trabajo arquitectónico para el sistema, producir el plan de administración de proyecto de software e identificar los riesgos clave del proyecto.
3. **Construcción**. Comprende esencialmente el diseño de sistema, la programación y las pruebas. Al término de esta fase se debe tener el sistema operando y la documentación correspondiente lista para entregarla a los usuarios.
4. **Transición**. Esta última etapa se encarga de mover el sistema desde la comunidad de desarrollo a la comunidad del usuario y hacerlo trabajar en su entorno real.

La perspectiva práctica en RUP describe buenas prácticas que son aconsejables en el desarrollo de sistemas. Se recomiendan seis buenas prácticas fundamentales:

1. **Desarrollar el software de forma iterativa**. Consiste en planificar los incrementos del sistema en base a la prioridades del usuario, desarrollando y entregando aquellas que son de más alta prioridad al inicio del proceso de desarrollo.
2. **Gestionar los requerimientos.** Documentar explícitamente los requerimientos del cliente, mantenerse al tanto de los cambios de estos requerimientos y analizar el impacto de los cambios en el sistema antes de aceptarlos.
3. **Utilizar arquitecturas basadas en componentes.** Hay que estructurar la arquitectura del sistema en componentes.
4. **Modelar el software visualmente.** Utilizar modelos gráficos de UML para presentar vistas estáticas y dinámicas del software.
5. **Verificar la calidad del software.** Asegurarse que el software cumple los estándares de calidad de la organización.

6. **Controlar los cambios del software.** Administrar los cambios en el software utilizando un sistema de gestión de cambios y procedimientos y herramientas de gestión de configuraciones.

Metodología ágil

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software. *Estas se caracterizan por enfatizar mucho menos el análisis y diseño a comparación de las metodologías tradicionales y por comenzar la implementación mucho antes en el ciclo de vida del software, porque considera más importante el software de trabajo que la documentación detallada. La sensibilidad al cambio es otro objetivo primordial de los procesos ágiles, como lo es la importancia de colaborar con el cliente.*

Principios de los métodos ágiles

Principio	*Descripción*
Participación del cliente	Los clientes deben estar fuertemente implicados en todo el proceso de desarrollo. Su papel es proporcionar y priorizar nuevos requerimientos del sistema y evaluar las iteraciones del sistema.
Entrega incremental	El software se desarrolla en incrementos, donde el cliente especifica los requerimientos a incluir en cada incremento.
Personas, no procesos	Se deben reconocer y explotar las habilidades del equipo de desarrollo. Se les debe dejar desarrollar sus propias formas de trabajar, sin procesos formales.
Aceptar el cambio	Se debe contar con que los requerimientos del sistema cambian, por lo que el sistema se diseña para dar cabida a éstos cambios.
Mantener la simplicidad	Se deben centrar en la simplicidad tanto en el software a desarrollar como en el proceso de desarrollo. Donde sea posible, se trabaja activamente para eliminar la complejidad del sistema.

*Metodología de Programación extrema (*Extreme Programming, XP)

La programación extrema (extreme programming) es una metodología ágil basada en el modelo iterativo e incremental, que considera las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, y el aprendizaje de los desarrolladores. Un factor clave en este método es la realimentación constante y fluida entre el equipo de desarrollo y el cliente, que pasa a formar parte importante del equipo de desarrollo.

En la programación extrema las necesidades del cliente no se expresan en una lista de requerimientos, en lugar de eso se utilizan tarjetas de papel llamadas historias de usuario, en las cuales el cliente describe las características que el sistema debe poseer. El equipo de desarrollo divide las historias de usuario en tareas y se estiman esfuerzos para su implementación, una vez hecho esto el cliente establece la prioridad de las historias a implementar. Si las necesidades cambian, las historias sin implementar cambian o pueden ser descartadas. Las tareas son realizadas por los programadores, los cuales trabajan en parejas, quienes realizan pruebas para cada tarea antes de escribir el código, con el fin de que el nuevo código se integre satisfactoriamente al sistema. En la figura 10 se ilustra el proceso de la metodología XP para producir un incremento en el sistema.

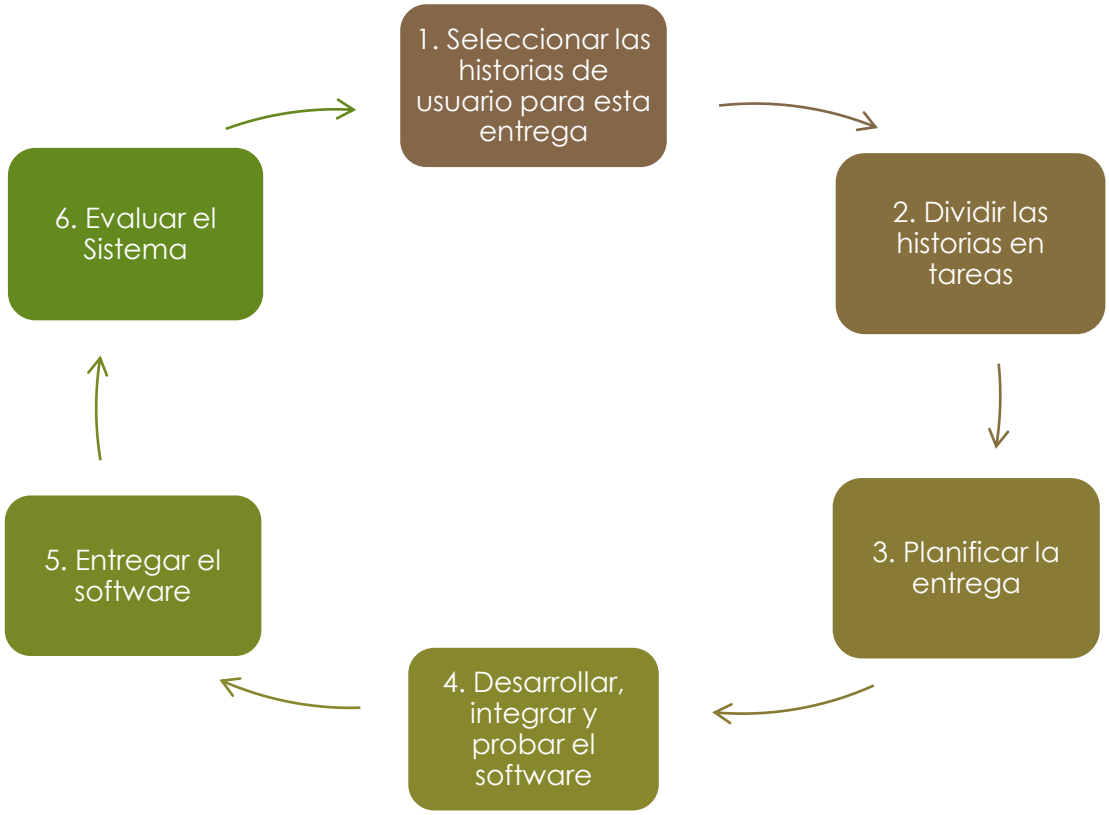

Figura 10. Ciclo de entrega en la programación extrema

A continuación se resumen las prácticas utilizadas en la programación extrema

Prácticas de la programación extrema

Principio o práctica	Descripción
Planificación incremental	Los requerimientos se registran en tarjetas de historias y las historias a incluir en una entrega se determinan según el tiempo disponible y su prioridad relativa.
Entregas pequeñas	El mínimo conjunto útil de funcionalidad que proporcione valor de negocio se desarrolla primero. Las entregas del sistema son frecuentes e incrementalmente añaden funcionalidad a la primera entrega.

Diseño sencillo	Solo se lleva a cabo el diseño necesario para cumplir los requerimientos actuales.
Desarrollo previamente probado	Se utiliza un sistema de pruebas de unidad automatizado para escribir pruebas para nuevas funcionalidades antes de que éstas se implementen.
Refactorización	Se espera que todos los desarrolladores reconstruyan el código continuamente tan pronto como encuentren posibles mejoras. Esto conserva el código sencillo y con fácil mantenimiento.
Programación en parejas	Los desarrolladores trabajan en parejas, verificando cada uno el trabajo del otro y proporcionando la ayuda necesaria para hacer siempre un buen trabajo.
Propiedad colectiva	Las parejas de desarrolladores trabajan en todas las áreas del sistema, de modo que no desarrollen islas de conocimientos y todos los desarrolladores posean todo el código. Cualquiera puede cambiar cualquier cosa.
Integración continua	En cuanto acaba el trabajo en una tarea, se integra en el sistema entero. Después de la integración se deben pasar al sistema todas las pruebas de unidad.
Ritmo sostenible	No se consideran aceptables grandes cantidades de horas extras, ya que a menudo el efecto que tienen es que se reduce la calidad del código y la productividad a medio plazo.
Cliente presente	Debe estar disponible al equipo de la XP un representante de los usuarios finales del sistema (cliente) a tiempo completo. En un proceso de la programación extrema, el cliente es miembro del equipo de desarrollo y es responsable de formular al equipo los requerimientos del sistema para su implementación.

Una vez que hemos analizados las características de estas 2 distintas metodologías decidimos utilizar en la realización de este proyecto principalmente la metodología RUP ya que nos ofrece la posibilidad de obtener un diseño estable y realizar la documentación necesaria para la fase de desarrollo aunque para la fase mencionada la combinaremos con la metodología de programación extrema utilizando características como la programación en parejas, todo con el propósito de generar un código libre de errores y así tener un sistema estable.

Esta decisión se tomó considerando que siendo un equipo pequeño de desarrollo, se pueden obtener buenos resultados experimentando ambas metodologías, sin embargo, una vez construida la primera versión, la documentación permitiría realizar cambios al sistema tal y como lo describe la metodología de RUP, esto con el fin de que los futuros requerimientos se piense en la continuidad de la operación y se verifique el impacto de los cambios.

Capítulo II. Reingeniería de los procesos actuales

Introducción

En este segundo capítulo abordaremos la solución que proponemos para el CEAQ, y para comenzar lo primero que haremos será aplicar un modelo de reingeniería a los procesos actuales con los que cuenta el centro.

Sin embargo, de esto nace una primera pregunta: ¿Qué es la reingeniería?

A continuación mostramos 2 visiones sobre este concepto:

Michel Hammer, uno de los fundadores del *Business Process Reengineering*[5], define la reingeniería de la siguiente forma:

"La reingeniería es empezar de cero, en una hoja en blanco, porque se considera que prácticamente todo lo que hacíamos antes, como empresas, parecería estar mal hecho, considerando los resultados obtenidos"

Esto nos dice que, según el autor, debemos considerar como reingeniería la eliminación de las prácticas actuales por considerarlas obsoletas y permitir el diseño de nuevos procesos y relaciones en pro de optimizar un negocio.

En contraste el Dr. Mauricio Lefcovich, consultor en administración de operaciones, nos dice:

"La reingeniería es la revisión fundamental y el cambio radical del diseño de procesos, para mejorar drásticamente el rendimiento en términos de costo, calidad, servicio y rapidez. La reingeniería de procesos es una especie de reinvención, más que un mejoramiento gradual."

La diferencia entre los 2 autores estriba en que mientras uno nos indica que la reingeniería es tirar lo que hemos hecho y empezar de nuevo (Michel Hammer) ,

[5] **Business Process Reengineering.** Método definido como la reconsideración fundamental y el reajuste radical de los procesos de organización, para lograr la mejoría drástica del desempeño actual en costo, servicio y velocidad.

el otro sustenta que lo que se debe hacer es una revisión del funcionamiento de los procesos actuales y de ahí mejorarlos (Dr. Mauricio Lefcovich).

Sin embargo, para esta tesis plantearemos una definición tomando lo mejor de cada una de las 2 posturas descritas anteriormente, enfocándola al contexto de la solución propuesta para el centro. Para nosotros la reingeniería es:

"El análisis y rediseño de los procesos de carácter fundamental de un negocio que proveen un servicio para mejorar drásticamente en términos de rapidez, calidad y servicio"

Esto nos dice que lo que haremos es retomar la forma en que actualmente se hacen las cosas, las analizaremos y rediseñaremos en forma radical, todo a favor de mejorar los servicios que proporciona el centro.

Ya que definimos lo que es la reingeniería, vamos a platicar un poco más sobre la forma en que la aplicaremos.

Los procesos que sean seleccionados para aplicarles reingeniería, como lo manejamos en la definición, deben ser de carácter fundamental. En el caso del centro serán las actividades mediante las cuales atiende a sus usuarios (a la comunidad de la Facultad de Medicina).

La reingeniería requiere la adopción de un enfoque centrado en el proceso elegido, empleando un equipo multidisciplinario, tecnologías de la información, liderazgo y análisis de procesos.

Vamos a dividir la reingeniería en 9 puntos esenciales que deberemos cubrir para adoptar los nuevos procesos de distribución de información, estos son:

1. **Procesos críticos.** Son aquellos que dadas sus características propias, le permiten lograr una ventaja competitiva en relación a sus competidores. Considerando el tiempo y los recursos del proyecto de reingeniería, estos deben ser el centro y la máxima prioridad.

2. **Fuerte liderazgo.** Es obligado contar con el apoyo y el ánimo de los directores de las empresas para efectos de hacer posible el proyecto de reingeniería. Los directores deben concientizar a todo el personal acerca de la necesidad urgente de efectuar los cambios profundos y radicales en la forma de generar el producto y los medios de distribución para el cliente. La reingeniería debe dar lugar a un nuevo perfil de directivo comprometido con la participación, inspiración, capacitación y creatividad aplicada.
3. **Equipos interdisciplinarios.** El proyecto debe ser llevado con la participación de las áreas involucradas en el proceso. La reingeniería da mejores resultados cuando se tiene una alta participación. Las iniciativas deben venir de los altos puestos a los bajos definiendo los objetivos y de los puestos bajos hacia los altos en función de buscar cómo obtener esos objetivos.
4. **Tecnologías de la Información.** Aunque la tecnología es uno de los principales detonantes de la reingeniería de procesos, se debe contemplar en el modelo las verdaderas necesidades de flujo de información y lo que se tiene actualmente para evitar fuertes inversiones en hardware y software que no sea lo que realmente el proyecto necesita.
5. **Filosofía de "borrón y cuenta nueva".** Esto es dejar a un lado lo actualmente existente y partir de lo que los clientes (en este caso, la comunidad) desean.
6. **Análisis de procesos.** El equipo encargado del proyecto debe conocer profundamente los procesos existentes en cuanto a: ¿Qué se produce en el centro?, ¿Qué tan bien lo está haciendo? Y ¿Qué factores lo afectan? Esto permitirá al equipo plantear nuevas configuraciones o el rediseño de procesos que permitirán un mayor valor agregado y rendimiento, y a su vez plantear objetivos claros sobre el rumbo a donde se quiere llegar.
7. **Tablero de Comando.** Se refiere en primer lugar a la utilización de una herramienta para determinar los procesos a rediseñar, en segundo lugar para monitorear los avances en el proceso de reingeniería y en tercer lugar

para medir los resultados de los procesos rediseñados, descubriendo al mismo tiempo nuevos procesos a recrear.

8. **Pensamiento triangular.** Combina 3 puntos o aspectos fundamentales que son: la búsqueda de un incremento absoluto (positivo o negativo) y en gran escala en cuanto a calidad, costos, productividad, tiempos de respuesta, entre otros. Otro punto fundamental es el conocimiento aplicado, y el tercer factor en consideración es la creatividad. El mero conocimiento no permite lograr el desafío de obtener un cambio radical en los procesos y sus resultados, la creatividad sin un objetivo claro e importante, no estará motivada, ni tendrá un rumbo claro y preciso. El sólo hecho de querer lograr importantes objetivos no ha de generar ningún resultado si ello no se apoya en el conocimiento y la creatividad. El líder debe sumar a la organización a aquellos individuos que poseen estas cualidades o bien debe capacitar y entrenar a sus empleados y obreros, cómo así mismo para encarar los grandes cambios estratégicos que la organización reclama.
9. **Aprendizaje organizacional.** Vinculado directamente al tema antes expuesto existe la necesidad imperiosa, de la organización y su personal en todos los niveles, de perfeccionarse tanto en sus conocimientos como del cambio.

La definición de estos puntos tiene como fin darnos cuenta que generar cambios radicales sin participación, ni objetivos estratégicos claros solo nos llevará al caos y al fracaso, y por consiguiente, no sabremos ni por qué hicimos este proyecto de reingeniería.

Una vez que ha sido planteada la teoría sobre el proceso de reingeniería, debemos decir que debe existir alguna forma para medir la calidad de nuestro resultado. Pero, ¿Cómo definir la calidad de un producto de software?

Podríamos escoger múltiples métricas para medir en función numérica la calidad de la solución, sin embargo, considerando que nuestro objetivo es la gestión de la información que genera el centro, el control de calidad debe venir del usuario a

quien el centro da el servicio. Recordemos que la solución está enfocada a varios actores, sin embargo, el control de calidad del producto debe reflejar la eficiencia en la distribución de información, por lo que la retroalimentación debe venir a través de cuestionarios y comentarios de los usuarios finales del sistema.

Uno de los procesos finales para los nuevos requerimientos será el análisis de los sistemas de calidad para conocer nuevas necesidades y el impacto que ha tenido el sistema sobre los usuarios.

Modelo de negocio

En esta etapa vamos a entender los objetivos del negocio para así generar un modelo de flujo de información correcto que nos permita proporcionar una solución integral sin alterar sus procesos ya existentes.

Sin embargo, es necesario responder a una cuestión para poder continuar; ¿Por qué modelar un negocio antes de modelar un sistema?

Para contestar esta pregunta, primero debemos entender que antes de proporcionarles una solución, el negocio ya existe, por lo que tienen personas, información y procesos, todos ellos conviviendo y dando resultados para cumplir sus objetivos planteados.

Para poder ofrecer una automatización de sus procesos, debemos entender perfectamente el objetivo de cada uno de los elementos que lo conforman y como interaccionan entre sí para llegar al fin propuesto.

Ahora, considerando la visión anterior, podemos decir que se requiere modelar el negocio para identificar con facilidad donde están sus problemas u oportunidades de crecimiento y mejora. No es posible automatizar procesos que no estén claramente definidos.

El objetivo del centro, en cuanto a información, es proveer a los alumnos y docentes de la Facultad de Medicina el apoyo para el aprendizaje quirúrgico y actualización, a través de congresos, prácticas e investigaciones. Cabe mencionar que el centro posee otros objetivos específicos (servicio a otros

sectores internos de la UNAM, servicios a universidades privadas y al sector privado) para investigación y prácticas, sin embargo la propuesta de reingeniería va enfocada únicamente a la gestión de su información.

Podemos definir gráficamente el negocio de la siguiente forma:

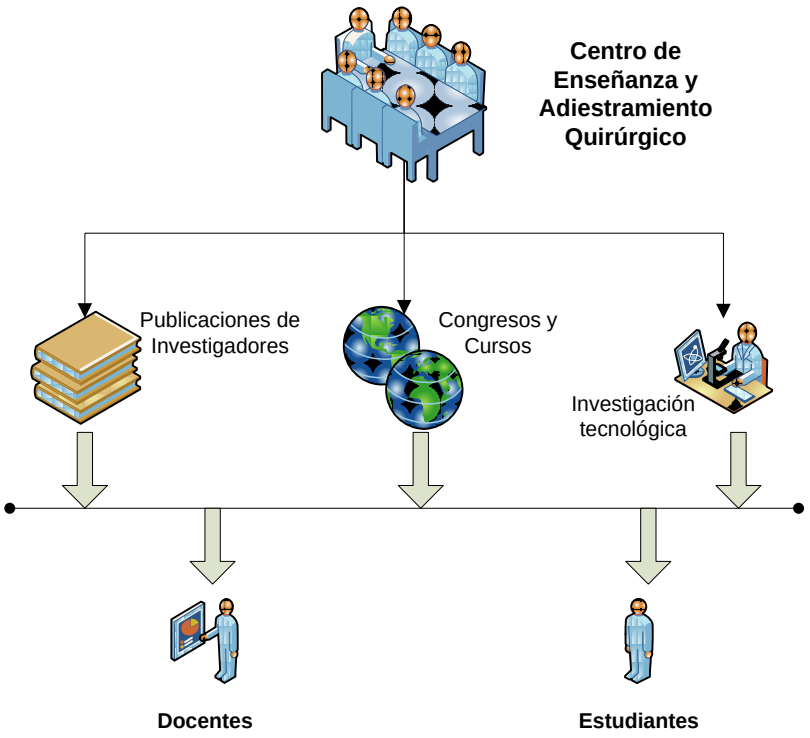

Figura 11. Modelo Actual de Negocio

Para cumplir uno de sus objetivos el personal del centro publica artículos e investigaciones, organiza congresos y cursos enfocados a sus diferentes usuarios y dedica tiempo de investigación a la aplicación de los recursos tecnológicos y a la enseñanza de sus temas de investigación.

La propuesta de reingeniería conserva sus procesos tal y como se encuentran actualmente pero agregan un nivel más, denominado Sistemas de Información y considera que debe existir retroalimentación por parte de los usuarios finales. Dichos sistemas pueden ser explotados por otras áreas que requieran información

estadística o comportamientos sobre lo que hace el centro, siempre con la filosofía de promover mejoras a favor de los propios usuarios finales.

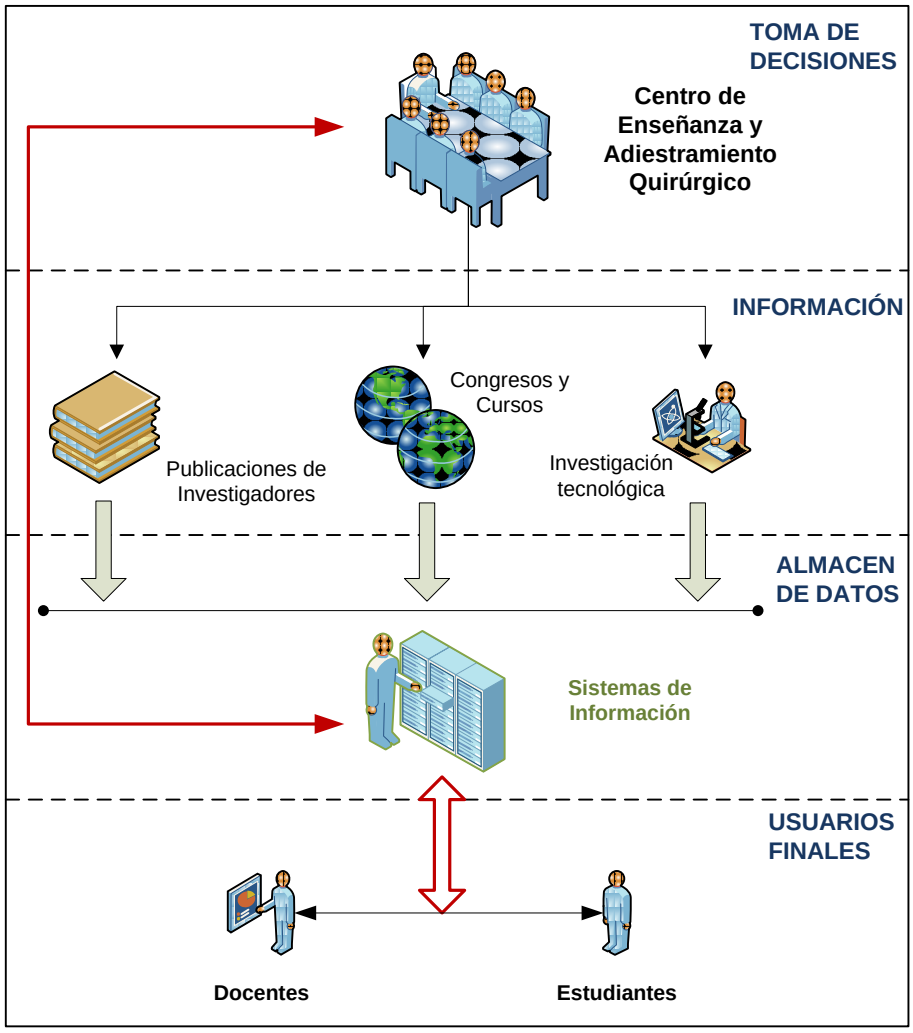

Figura 12. Propuesta de nuevo Modelo de Negocio

La propuesta de reingeniería crea un flujo de datos entre los Usuarios Finales y la Toma de Decisiones, procesados por un Sistema de Información; este flujo será representado por un sistema web que permitirá visualizar esa información almacenada ya de forma procesada.

Si consideramos éste nuevo modelo como el modelo del negocio, debemos formalizar la especificación de los requerimientos para así, en primer lugar, lograr que el equipo de trabajo se encuentre informado de lo que vamos a realizar, y en

segundo lugar, para tener documentación que avale el diseño que vamos a realizar más adelante.

Al finalizar esta parte del capítulo 2, debemos tener una visión clara y delimitada del proyecto, con el fin de que, a partir de este momento, realicemos únicamente tareas que harán que la solución avance hacia un objetivo (planteado desde el inicio) y no nos desviemos de éste.

En primer lugar vamos a definir un documento visión, que es lo que nos ayudará a dar una primera impresión de lo que realizará el sistema y como apoyará el mejoramiento del proceso de distribución de información.

Documento Visión

Introducción

Es un documento creado por el equipo del proyecto en los primeros días para luego construir los casos para realizarlo; negociar el alcance total del proyecto y los requerimientos de más alto nivel enfocados en el cliente; y lograr así en el equipo un lineamiento común y obtener un acuerdo común para la decisión del proyecto.

Este documento es de utilidad debido a que una temprana visión por niveles del trabajo ayuda a desmenuzar la definición del proyecto antes que el equipo del proyecto realice las especificaciones de requerimientos detalladas.

Permite también al equipo de trabajo darse cuenta rápidamente si está enfocado en lo que es más importante para el cliente. Asegura que las metas del proyecto estén correctamente definidas en términos de quienes son los clientes y que beneficios absolutos les provee este proyecto, evitando así que este escape de los límites globales.

Gracias a esto, el equipo de trabajo puede calcular el alcance, calendario y recursos del proyecto, así como los requerimientos críticos del cliente.

La forma de realizar un documento visión se puede definir de la siguiente forma:

1. Bosquejar una visión al inicio para cristalizar los objetivos del proyecto.
2. Cuando todo el equipo ha sido conformado, se deberá realizar una reunión con todos los involucrados o representantes de los grupos funcionales. Se crearán las secciones del documento con lo resultante de alguna acción llevada a cabo en la reunión (por ejemplo una lluvia de ideas). Si la visión ya existe deberá ser revisada, editada y enviada a todos los integrantes. De aquí se podrán documentar los problemas presentados y la forma en que fueron solucionadas.
3. El equipo deberá investigar alternativas para satisfacer las necesidades del cliente.
4. El documento deberá ser revisado de forma iterativa, analizando como las soluciones alternativas favorecen el proyecto y lo mejora acercándolo cada vez más al objetivo.
5. Al final de la investigación, el documento generado será la visión final que todos los involucrados deben tener sobre el proyecto que se iniciará. Este documento final podrá considerarse una carta de requerimientos, la cual deberá ser avalada por el cliente y será la base del diseño de la solución.
6. Es necesario que este documento sea visible para todos y que guie correctamente la etapa de diseño y mantenga los objetivos principales del proyecto actualizados para asegurar llegar al alcance correcto.

Podemos decir que, en el caso particular de esta tesis, el documento visión será el resultado del ciclo de levantamiento y análisis de requerimientos, y de la delimitación del alcance del sistema.

Será el resultado de una serie de reuniones que se tendrán con el cliente y otras reuniones con el grupo de trabajo que buscará puntualizar aquello que el cliente nos quiere expresar. Para esto se requieren de documentos que formalicen estas reuniones; estos documentos se denominan ***minutas***[6].

[6] Extracto o borrador que se hace de un contrato u otra cosa, anotando las cláusulas o partes esenciales, para copiarlo después y extenderlo con todas las formalidades necesarias para su perfección.

Una vez realizadas las minutas y que el equipo de trabajo se ha reunido para traducir lo que se vio en cada una de las juntas se genera el primer documento oficial que será aquel que servirá como guía durante el desarrollo de la solución y permitirá analizar el entregable o los entregables y validar si el desarrollo cumple con lo especificado.

En este momento es cuando se requiere darle un nombre clave al producto de software que estamos a punto de definir. El nombre permitirá considerar la solución como una entidad y facilitará su comprensión dentro de este trabajo.

El nombre clave del Sistema de Información para el Centro de Enseñanza y Adiestramiento Quirúrgico de la Facultad de Ingeniería será **SOPHIA Versión 1**, denominado así por su significado "Aquella que posee sabiduría" y su función dentro del centro que es gestionar el conocimiento que se crea.

Documento Visión

Introducción

Propósito

El sistema SOPHIA, será un administrador de artículos y material multimedia para proveer un apoyo adicional a los estudiantes utilizando los recursos informáticos con los que cuenta la facultad y los trabajos que han realizado los estudiantes e investigadores de la Facultad de Medicina.

El objeto de éste documento es recoger, analizar y definir las necesidades de alto nivel y las características del sistema. El documento se centra en las funcionalidades requeridas por los usuarios finales y la propuesta por los participantes en el proyecto.

Estas funcionalidades se basan principalmente en las mejores prácticas (criterios a tomar en cuenta, algoritmo a utilizar, recursos disponibles, etc.) para la eliminación de redundancia, la presentación de la información en el momento justo, toma de decisiones para los funcionarios del centro y lo más importante la eficiencia en el uso del recurso tiempo. La idea es que la información se

encuentre disponible en función de la disponibilidad de la red de datos de la Facultad de Medicina y no de los horarios laborales del centro.

Los detalles de cómo el sistema cubre los requerimientos se pueden observar en la especificación de los casos de uso y otros documentos adicionales.

Alcance

El documento Visión se ocupa, como ya se ha señalado, del sistema SOPHIA del Centro de Enseñanza y Adiestramiento Quirúrgico de la Facultad de Medicina, que presta el servicio de enseñanza.

El sistema permitirá, a los administradores, gestionar todo lo relativo a la creación y modificación de artículos y material, a los investigadores todo lo relativo a la creación y modificación de artículos propios, a los estudiantes y profesores la visualización de todo el material del sitio y la participación activa a través de cuestionarios, encuestas y comentarios, y al público general la visualización de material y sitios de interés.

Dado que el sistema será albergado en un servidor bajo el dominio de la Facultad de Medicina y de la UNAM, todo artículo y material publicado debe ser autorizado por un perfil especial que lo revisará y autorizará o bien, lo rechazará cumpliendo con los reglamentos que estipula esta facultad y nuestra universidad.

Definiciones, Acrónimos y Abreviaciones

- **RUP.** Son las siglas de Rational Unified Process. Se trata de una metodología para describir el proceso de desarrollo de software.
- **UML.** Son las siglas de Unified Modeling Language. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.
- **XP.** Son las siglas de eXtreme Programming. Es una *metodología ágil*, para el desarrollo y sistemas de software de mediana o baja escala.
- **Stakeholders.** Personas que se verán afectadas en un cambio.

Posicionamiento

Oportunidad de Negocio

SOPHIA permitirá a los investigadores dar a conocer artículos de interés para la comunidad de la Facultad de Medicina y público en general, a través de un portal donde para ellos será transparente la programación detrás del mismo para la publicación de su material.

También permitirá la publicación de ligas de interés y material multimedia que genera este centro (imágenes, videos y modelos tridimensionales), este último acercará al alumno de una forma sencilla al estudio de partes del cuerpo humano de forma autodidacta, a cualquier hora del día y lo más parecido a la realidad, en caso que el alumno desee mayor información, podrá acercarse al centro y asistir a una sesión de los modelos de mayor nivel de detalle y aprovechando los laboratorios de la UNAM.

El sistema debe ser accesible para todos los usuarios, y permitirá un mayor flujo de información en ambas direcciones, del centro a la comunidad con lo mencionado con anterioridad y de la comunidad al centro a través de encuestas, cuestionarios y buzón de sugerencias que permitirá saber lo que la comunidad quiere ver y saber acerca del CEAQ.

Sentencia del Problema

El problema de	La ausencia de un sistema, sin aprovechar los recursos tecnológicos actuales, hace que el proceso de distribución de información: • Se realice de una forma lenta. • No llegue a toda la comunidad. • Se pierda interés por parte de la comunidad hacía el trabajo realizado. • Que la comunidad no aproveche en su totalidad las herramientas que ofrece este centro de investigación. • Se pierdan noticias de interés. • No haya un posicionamiento mayor sobre la información generada.

afecta a	• Investigadores del Centro de Enseñanza y Adiestramiento Quirúrgico. • Alumnos de la Facultad de Medicina • Profesores de la Facultad de Medicina
El impacto asociado es	Planificar y gestionar la información que genera el centro, y que se realice de la manera más eficiente posible. Debido a que este proceso es de vital importancia hacia la enseñanza, se requiere un sistema que permita la publicación de esta información de manera sencilla al igual que su visualización, de forma que la infraestructura y él código sea transparente para los usuarios.
Una adecuada solución sería	Una interfaz que permita a los usuarios publicar, visualizar, autorizar y comentar la información que genera el centro utilizando la infraestructura de red que posee la Facultad de Medicina y nuestra universidad en pro de la enseñanza y las mejores prácticas. También poder conocer en todo momento los comentarios de los usuarios del sitio para así poder adecuarlo a las nuevas necesidades que vayan surgiendo. Generar interfaces amigables y sencillas con las que se pueda acceder a la información.

Descripción de Stakeholders y Usuarios

Para el desarrollo de un producto de software que se ajuste a las necesidades de los usuarios, es necesario identificar e involucrar a todas las personas en el proyecto como parte del proceso de modelado de requerimientos. También surge la necesidad de identificar a los futuros usuarios del sistema, asegurándose que el conjunto de participantes los representen adecuadamente.

En esta sección se pretende mostrar el perfil de los participantes y de los usuarios, además de los problemas más importantes que estos tienen actualmente, esto con el fin de enfocar la solución hacia ellos.

Resumen de Stakeholders

Nombre	Descripción	Responsabilidades
Personal del CEAQ	Director del Centro de Enseñanza y Adiestramiento Quirúrgico.	Aprobación e implantación del sistema.
	Representante de la generación de material multimedia	Encargado del proceso de creación y administración del material multimedia.
	Administrador del contenido del dominio de la Facultad de Medicina	Es el encargado de administrar lo que la Facultad de Medicina pública dado que el dominio representa una gran responsabilidad.
Investigadores del CEAQ	Personal a cargo de realizar artículos y resúmenes de sus investigaciones.	Son los encargados de realizar artículos y resúmenes acerca del trabajo que realizan en función de una mejor formación de los estudiantes.
Alumnos y Docentes	Estudiante y docentes de la Facultad de Medicina	Son las personas a las que está enfocada la información que genera el centro con el fin de buscar una mejor formación utilizando las nuevas herramientas tecnológicas.

Resumen de usuarios

Nombre	Descripción	Stakeholder
Administrador de SOPHIA	Es el encargado de administrar el contenido del sitio web, así como colocar información de forma general (ligas de interés, encuestas, cuestionarios, etc.). Será el encargado de administrar el contenido multimedia del sitio web y controlará a los usuarios del sistema.	
	Es el encargado de gestionar los artículos que generan los investigadores del centro para así poder controlar lo que se publica bajo el dominio de la Facultad de Medicina y de la UNAM.	
Investigadores	Son los creadores de informes y artículos para los alumnos y profesores de la Facultad de Medicina sobre sus investigaciones.	Investigadores del CEAQ
Personal de Servicio Social	Son el personal que apoya a la generación de material multimedia y ayudan a la administración de SOPHIA.	Personal del CEAQ
Profesores	Son los creadores de informes, artículos, encuestas y cuestionarios para los alumnos y profesores de la Facultad de Medicina.	Docentes

Alumnos	Son aquellos que pueden participar abiertamente en el sitio web con acceso a material exclusivo de la facultad.	Alumno
Público General	Aquellas personas que pueden entrar al sitio pero no pueden formar parte de él abiertamente.	

Entorno del usuario

Los usuarios podrán acceder al sistema a través de cualquier navegador de Internet, sin embargo y de acuerdo a su rol, para participar activamente deberán acceder con una cuenta de usuario y una contraseña.

El sistema permitirá a los usuarios participar enviando comentarios a los administradores y a los investigadores sobre los temas expuestos, podrán participar en encuestas y en cuestionarios que permitan conocer sus nuevos intereses para así poder enfocar el sitio a las necesidades reales, todo esto a través de simples cuestionarios web.

A los usuarios con roles administrativos y de investigadores se les permiten otras funciones las cuales deberán cubrir de igual forma a través de algún navegador.

Especificación de Requerimientos

Aunque la especificación de los requerimientos es parte del documento visión, en esta ocasión lo hemos separado para que sean un punto principal de este trabajo y nos permitirá identificar al final del proceso el cumplimiento de las metas planteadas por los usuarios del centro. Los requerimientos reales del sistema los hemos aterrizado en cinco puntos claves, estos son:

1. Rediseño del sitio web.
2. Implementación de canal de noticias (estándar RSS) y sitios de interés.
3. Distribución de material multimedia.
4. Control de acceso
5. Cuestionarios de seguimiento

A continuación podemos apreciar un listado por requerimiento sobre las tareas que componen a cada uno, esto con el fin de dimensionar los alcances y definir propiamente lo que se realizará para poder dar por concluido el sistema.

1. Rediseño del sitio web.
   1.1. Cambio de interfaz gráfica.
      1.1.1. Definición del sitio utilizando módulos para evitar la carga completa de cada página.
      1.1.2. Definición de un estándar para los colores y los tipos de letra, utilizando como referencia gráfica el logotipo del centro.
   1.2. Optimización del tiempo de descarga.
      1.2.1. Implementación de tecnología AJAX.
   1.3. Mejora de la navegación dentro del sitio web.
      1.3.1. Definición del árbol del sitio web.
   1.4. Buscador de noticias para el sitio web por fecha de creación, titulo o creador.
2. Implementación de canal de noticias y sitios de interés.
   2.1. Creación de estructuras de datos para noticias y sitio de interés.
   2.2. Interfaz para la administración de contenido.
   2.3. Interfaz para la captura de noticias.
   2.4. Interfaz para la administración de noticias a por usuario.
   2.5. Interfaz para la distribución de noticias dentro del sitio web y utilizando XML compatible con los lectores de noticias.
   2.6. Interfaz para la autorización de noticias.
   2.7. Interfaz para la captura de sitios de interés.
   2.8. Interfaz para la visualización de los sitios de interés.
3. Distribución de material multimedia.
   3.1. Diseño de estructura XML para la administración de los archivos multimedia para una fácil gestión.
   3.2. Visor para modelos tridimensionales con controles para rotación y zoom.
   3.3. Galería de imágenes.

3.4. Galería de videos y reproducción por streaming[7].

4. Control de acceso.
   4.1. Diseño de estructura XML para el control de perfiles.
   4.2. Interfaz para el alta de usuarios.
   4.3. Control de inicio y fin de sesión.
   4.4. Interfaz para recuperar contraseñas por el administrador y cambio de contraseña para el usuario.
   4.5. Interfaz para el reporte de usuarios.
5. Cuestionarios de seguimiento
   5.1. Front para la captura de encuestas.
   5.2. Reportes de encuestas.
   5.3. Interfaz para la visualización de encuestas para los usuarios.
   5.4. Interfaz para la captura de cuestionarios.
   5.5. Interfaz para la visualización de cuestionarios.
   5.6. Interfaz de reportes de cuestionarios.
   5.7. Interfaz para buzón de comentarios.
   5.8. Reporte sobre comentarios.

Con esta definición de requerimientos englobamos las necesidades del usuario y a través de ellos proporcionamos una solución la cuál cubre todas sus necesidades. Con la conclusión de estos puntos se podrá poner a disposición de los usuarios la versión 1.0 de SOPHIA.

Requisitos

Una vez que tenemos definidos el documento visión y los requerimientos, podemos pasar a la etapa de la elección de las tecnologías. Como ya se había comentado, esto depende de la solución que vamos a proporcionar y de la infraestructura tecnológica con la que cuenta el centro.

[7] Streaming. Consiste en la distribución de audio y video por internet, que no requiere completar su descarga para la reproducción, pues mientras se va reproduciendo, el archivo continúa descargándose.

Para la creación y puesta en marcha del sistema de información y la interfaz de visualización y captura de datos, se han definido los siguientes productos de software para las diferentes funciones que se van implementar, las aplicaciones principales que proponemos como solución son:

Software	Función
Microsoft Windows Server 2008 Standard Edition	*Sistema Operativo*
Internet Information Services 7	*Servidor Web y de aplicación*
ASP.NET	*Lenguaje de desarrollo*
.NET 3.5	*Framework*
Microsoft SQL Server 2008	*Motor de Base de Datos*
Photoshop CS4	*Diseño de sitio web*
Flash CS4	*Diseño de galerías de fotos, visor de modelos tridimensionales y de reproductor de videos*
Illustrator CS4	*Diseño de imágenes con vectores.*
Maya 2010 **Z Brush 3.1**	*Creación de modelos tridimensionales*
Papervision 3D	*Librería de visualización para modelos tridimensionales*
Microsoft Visual Studio 2008	Interfaz *de desarrollo*
Microsoft Chart Controls 3.5	*Librería para creación dinámica de gráficas*
ASP.NET Ajax Library	*Librería para la creación de elementos AJAX con .NET*
JQuery 1.3.2	*Framework para javascript.*

A continuación mostraremos un panorama general sobre las tecnologías que vamos a aplicar para poder tener en claro los alcances y las limitaciones de cada una de las plataformas.

Lenguaje Really Simple Syndication (RSS 2.0)

Es un lenguaje que surge de la aplicación del metalenguaje XML. Por tanto un archivo RSS es un documento de texto compuesto por etiquetas acotadas entre los símbolos de mayor y menor que son similares a las utilizadas por XHTML.

Figura 13. Logotipo estándar del lenguaje RSS

Los archivos RSS comúnmente se llaman feeds RSS o canales RSS y contienen un resumen de lo publicado en el sitio web de origen. Se estructura en uno o más ítems. Cada ítem consta de un título, un resumen de texto y un enlace a la fuente original en la web donde se encuentra el texto completo. Además puede incluir información adicional como el nombre del autor o la fecha y la hora de publicación del contenido. Por tanto, cualquier fuente de información susceptible de poder ser resumida en ítems (los mensajes de un foro, por ejemplo) pueden distribuirse utilizando RSS.

A pesar de que la sigla que representa este formato es de reciente creación, existe una controversia acerca de su significado. Para algunos autores, sería "Rich Site Summary" o Resumen Enriquecido de un Sitio y para otros "Really Simple Sindication" o Publicación Realmente Simple. Los que apoyan la primera alegan que, al observar el contenido de estos archivos, se observan los resúmenes del contenido y la ubicación de la información, para que otros sitios la hallen y puedan utilizarla teniéndola siempre actualizada.

Aunque esto sea cierto, los que abogan por la segunda posibilidad sostienen que el significado que ellos proponen es más representativo de la sigla, del formato y de la utilidad del archivo en sí, porque otros "Really Simple Sindication" o

Publicación Realmente Simple alude al hecho de que estos ficheros se utilizan para la publicación de contenidos de sitios web.

Para leer los canales RSS se requiere un programa especializado en este tipo de contenidos, actualmente, navegadores como Internet Explorer, Mozilla Firefox, Chrome y otros, contienen módulos para leer y actualizar este tipo de información.

Las noticias llegan al usuario cuando éste inicia su programa lector de feeds RSS, en lugar de que el usuario deba ir a leer las noticias a las diferentes fuentes o sitios web donde se publican. El uso de un lector de feeds supone un gran ahorro de tiempo y de incomodidades cuando se es lector habitual de un puñado de sitios web distintos ya que es posible echar un vistazo rápido a todos los contenidos nuevos publicados en varias decenas de sitios web distintos en prácticamente el mismo tiempo que supondría consultar vía web un único sitio.

Los archivos RSS, al igual que las páginas web, disponen de un URL o dirección de Internet, por ejemplo:

http://www.seguridad.unam.mx/xml.dsc?arch=bol_unam-cert

"Canal RSS del CERT de la UNAM"

Existen distintas versiones del lenguaje RSS: las versiones 0.90 y 0.91 son las primeras y fueron desarrolladas por Netscape. La versión 1.0 fue desarrollada por un grupo independiente basándose en el formato RDF. La versión 2.0 (y antes que ésta distintas versiones 0.9x) fue el resultado de la adopción de la tecnología de Netscape por parte de la empresa UserLand Software. Todas ellas están basadas en el lenguaje originalmente definido por Netscape pero no todas son compatibles entre sí.

- **RDF** — La versión 1.0 del lenguaje RSS también es conocida como RDF. Por este motivo algunos feeds o canales RSS están etiquetados como "RSS 1.0" o "RDF" y guardados en archivos con extensión ".rdf".

- **RSS2** — A la versión 2.0 del lenguaje RSS también se llama RSS2. Por tanto algunos feeds o canales RSS están etiquetados como "RSS2" o "RSS 2.0".
- **Atom** — Atom también es un sublenguaje XML. No se corresponde ni se basa en ninguna versión de RSS, pero es un formato muy similar a éste y que sobre todo tiene el mismo objetivo: permitir la distribución de contenidos y noticias de sitios web. La versión más actual es la 0.3 de febrero de 2004. Las mejoras que supone respecto a RSS (en cualquiera de sus versiones) hacen que su uso se extienda rápidamente a pesar de ser algo más complicado. Un documento Atom puede contener más información (e incluso más compleja) y es más consistente que un documento RSS.

A continuación colocaremos un ejemplo de una noticia a través de un RSS, en este ejemplo podemos apreciar las etiquetas y puede ser generado de manera dinámica, utilizando una base de datos y un servidor de aplicación (en este caso ASP.NET).

```
<rss version="2.0">
  <channel>
    <title>
      Noticias RSS.
    </title>
    <link>
      http://www.unam.mx
    </link>
    <description>
      Este es un ejemplo sobre un archivo RSS
    </description>
    <cloud domain="unam.mx" port="80" path="/RPC2"
registerProcedure="datos.rssNotificar" protocol="xml-rpc"/>
    <docs>
      http://www.unam.mx/
    </docs>
    <generator>
      CEAQ 1.0
    </generator>
    <ttl>
      60
    </ttl>
    <item>
      <title>
        Tesis sobre el CEAQ de la Facultad de Medicina de la UNAM
      </title>
      <link>
```

```
        http://www.unam.mx
      </link>
      <description>
        Contenido sobre el articulo RSS
      </description>
      <pubDate>
        Mon, 3 Nov 2007 06:57:42 GMT
      </pubDate>
    </item>
  </channel>
</rss>
```

Código de canal RSS de ejemplo

Este es un ejemplo sencillo, pero que nos muestra la sencillez con la que podemos proveer nuestra solución de este tipo de tecnologías para hacer más sencilla la distribución y acceso a la información.

Manipulación de Modelos Tridimensionales

La creación de modelos tridimensionales que serán desplegados en SOPHIA, requerirá de software especializado para su creación, este se describe a continuación mencionando su utilización en el proceso de elaboración de modelos. Posteriormente detallaremos los pasos a seguir para crear un modelo tridimensional que se despliegue en el visor de la aplicación desarrollada.

Autodesk Maya

Este es un programa orientado al desarrollo de gráficos tridimensionales, efectos especiales y animación. Originalmente fue desarrollado por Alias Systems Corporation y en la actualidad la empresa Autodesk es la encargada de su desarrollo y distribución.

Maya trabaja con cualquier tipo de superficies NURBS[8], polígonos[9] y subdivisión surfaces[10], para el desarrollo de los modelos en baja resolución que se realizaron se utilizaron polígonos, debido que es más fácil su manipulación al momento de exportar a otro formato.

[8] Acrónimo inglés de la expresión *Non Uniform Rational B-Splines*. Modelo matemático muy utilizado en la computación gráfica para generar y representar curvas y superficies.
[9] Los polígonos son usados en la computación gráfica para la composición de objetos en tercera dimensión, representan las caras de los objetos.
[10] En cómputo gráfico en tercera dimensión es la representación de una superficie lisa o suavizada a partir de una malla de polígonos de menor detalle.

Elegimos trabajar con Maya porque posee una interfaz muy amigable lo que facilita el trabajo además existen los aditamentos que permiten exportar geometrías a formato collada, el cual se explicará más adelante, que es necesario para poder implementarlos en una página web.

Otra razón por la cual se eligió este programa es que este es el utilizado por el personal asignado de la generación y manipulación de modelos tridimensionales en el CEAQ.

Con Maya crearemos la geometría base de nuestro modelo tridimensional la cual se llevará a otro programa para agregarle detalles y posteriormente regresar a Maya para su exportación al formato que leerá el visor de modelos tridimensionales de SOPHIA.

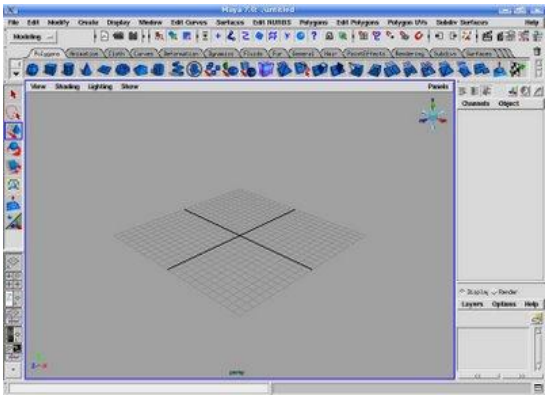

Figura 14. Ambiente de trabajo de Maya

ZBrush

Es un software de modelado 3d, texturizado y pintura digital que constituye un nuevo paradigma dentro del ámbito de la creación de imágenes de síntesis gracias al original planteamiento de su proceso creativo.

Zbrush comenzó como un original programa que permitía crear pinturas digitales e insertar en ellas objetos tridimensionales, que podían ser simples primitivas originadas en el propio programa, o podían ser importadas en formato "obj". Poco después, la versión 2.5 canalizó la clave diferencial de este software de un modo más claro al ser usado en fase beta por los artistas de Weta digital para detallar y

esculpir diferentes personajes de la segunda y tercera entrega de "El señor de los Anillos". El descubrimiento de Zbrush como un software capaz de esculpir detallados modelos de un modo semejante a pintar en los mismos facilitó su popularización entre los artistas 3d de las industrias del cine, videojuego e ilustración.

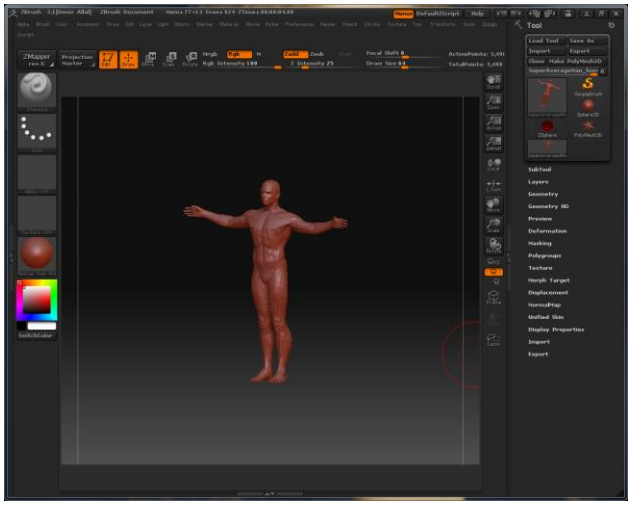

Figura 15. Ambiente de trabajo de ZBrush

Papervision3D

Es una librería que permite trabajar y generar objetos tridimensionales en Flash.

Esta librería surge de la mente del español Carlos Ulloa, quien como proyecto personal se fijó la meta de poder visualizar objetos tridimensionales en Flash. Cuando su proyecto comenzó a crecer decidió liberarlo convirtiéndose así en una librería de código abierto.

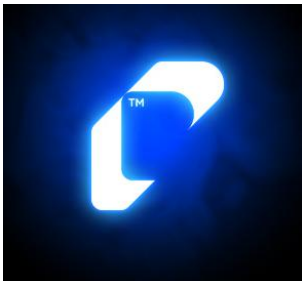

Figura 16. Logotipo de Papervision3D

El flujo de trabajo que se utiliza para la implementación de modelos tridimensionales en flash es la siguiente:

Se generan los modelos tridimensionales, en un programa modelador, en este caso Maya, son texturizados y detallados los modelos con la ayuda de Photoshop y Zbrush, después desde Maya exportamos los modelos al formato COLLADA (.dae), se utiliza este formato porque Papervision3D cuenta con las rutinas que permiten cargar modelos en dicho formato de manera directa.

Tecnología de Streaming de Video

Streaming es un término que se refiere a ver u oír algún contenido multimedia directamente en una página web, sin necesidad de descargarlo previamente en el dispositivo. Hablando un poco en el lenguaje de los negocios, se refiere a una estrategia *sobre demanda*[11] para la distribución de contenido multimedia a través de Internet.

Esta tecnología basa su funcionamiento en que, previo a la ejecución y durante la misma, se guarde un *buffer*[12] de información sobre el contenido multimedia de lo que se va a escuchar un tiempo después. Así, se permite, escuchar música o ver video sin necesidad de descárgalos previamente.

Desde la aparición, en 1995, del Real Audio 1.0 de la compañía Real Networks, esta tecnología ha evolucionado desde el radio en internet, hasta actualmente videos en *High Definition (HD)*[13].

La ventaja de esta tecnología es que, un archivo multimedia va siendo descargado al mismo tiempo que el usuario lo está utilizando, permitiendo aprovechar mejor el tiempo del usuario.

[11] Sobre demanda. Conocido en el lenguaje inglés como Just in Time (JIT) encierra un conjunto de técnicas concebidas para mejorar la calidad y eficiencia de un diverso rango de operaciones.
[12] Es información almacenada temporalmente en espera de ser procesada.
[13] Alta Definición. Sistema de video con una mayor resolución que la definición estándar (720 × 486), alcanzando resoluciones como 1280 x 720 o 1920 x 1080 (Full HD)

La filosofía de los streaming es clara: "Proporcionar al usuario un acceso claro, continuo y sin interrupciones de contenido multimedia"

Para llevar a cabo su objetivo, esta tecnología se basa de los siguientes componentes:

- **Codecs.** Son archivos que residen en nuestros equipos de cómputo o dispositivos electrónicos que permiten interpretar el contenido determinado de un tipo de un archivo multimedia y poder ser presentado al usuario.
- **Protocolos Ligeros.** Por ejemplo, el protocolo UDP [14]que hacen entrega de datos de forma más rápida a comparación de los TCP, esto porque no envían un mensaje de confirmación a la entrega del paquete.
- **Precarga.** Los datos que entrega el servidor pueden estar sujetos a demoras por el ancho de banda o bien, por la alta concurrencia que tiene el servidor al momento de enviarlos, esto provoca pausas y demoras en la reproducción de los contenidos, debido a que los datos escasean. Para eso, los programas que utilizan esta tecnología, utilizan un buffer de datos en memoria que permiten continuar con la reproducción en caso que la conexión sufra alguna falla.
- **Red de distribución de contenido.** Este es el medio de transmisión que utilizan los programas para comunicarse con los servidores de datos. Estos pueden ser compartidos, si la carga multimedia es baja o bien, dedicados si es que muchos usuarios utilizan este servicio.

Este tipo de tecnologías desechan el contenido descargado una vez que ha sido reproducido y que el usuario ha dejado de verlo, sin embargo, actualmente existen programas para rescatar estos archivos y guardarlos en formatos dentro de los clientes.

[14] User Datagram Protocol. Es un protocolo del nivel de transporte del modelo OSI que permite el envió de datagramas por una red sin que se haya establecido una comunicación previa.

Un streaming se puede definir gráficamente de la siguiente manera:

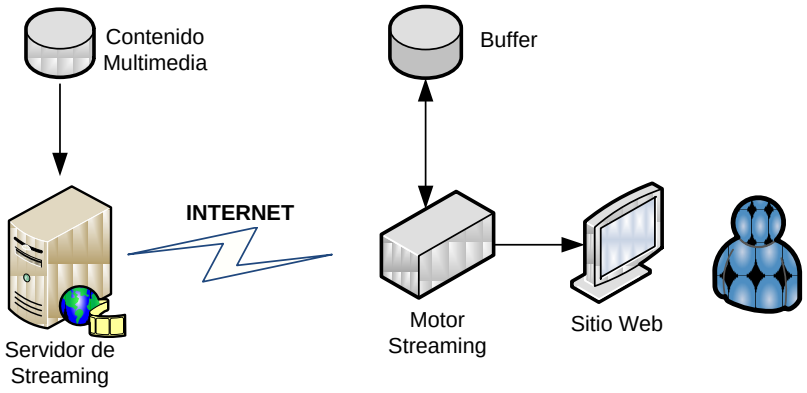

Figura 17. Diagrama de servidor Streaming y Cliente

Lenguajes para crear sitios dinámicos

AJAX

El término AJAX se anunció por primera vez en un artículo escrito por Jesse James Garrett el 18 de Febrero de 2005, lo cual hizo que apareciera por primera vez un término normalizado a este tipo de aplicaciones que estaban apareciendo.

El artículo define a AJAX, no como una tecnología en sí misma, sino como varias tecnologías independientes unidas entre sí de formas nuevas y sorprendentes.

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear presentación basada en estándares.
- DOM[15], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de la información.
- JavaScript, para unir todas las demás tecnologías.

[15] DOM, acrónimo de Document Object Model. Es una interfaz de programación de aplicaciones que proporciona un conjunto estandarizado de objetos para representar documentos HTML o XML. El responsable es el World Wide Web Consortium. Esta API puede ser aprovechada con lenguajes como Javascript

AJAX se define de la siguiente forma:

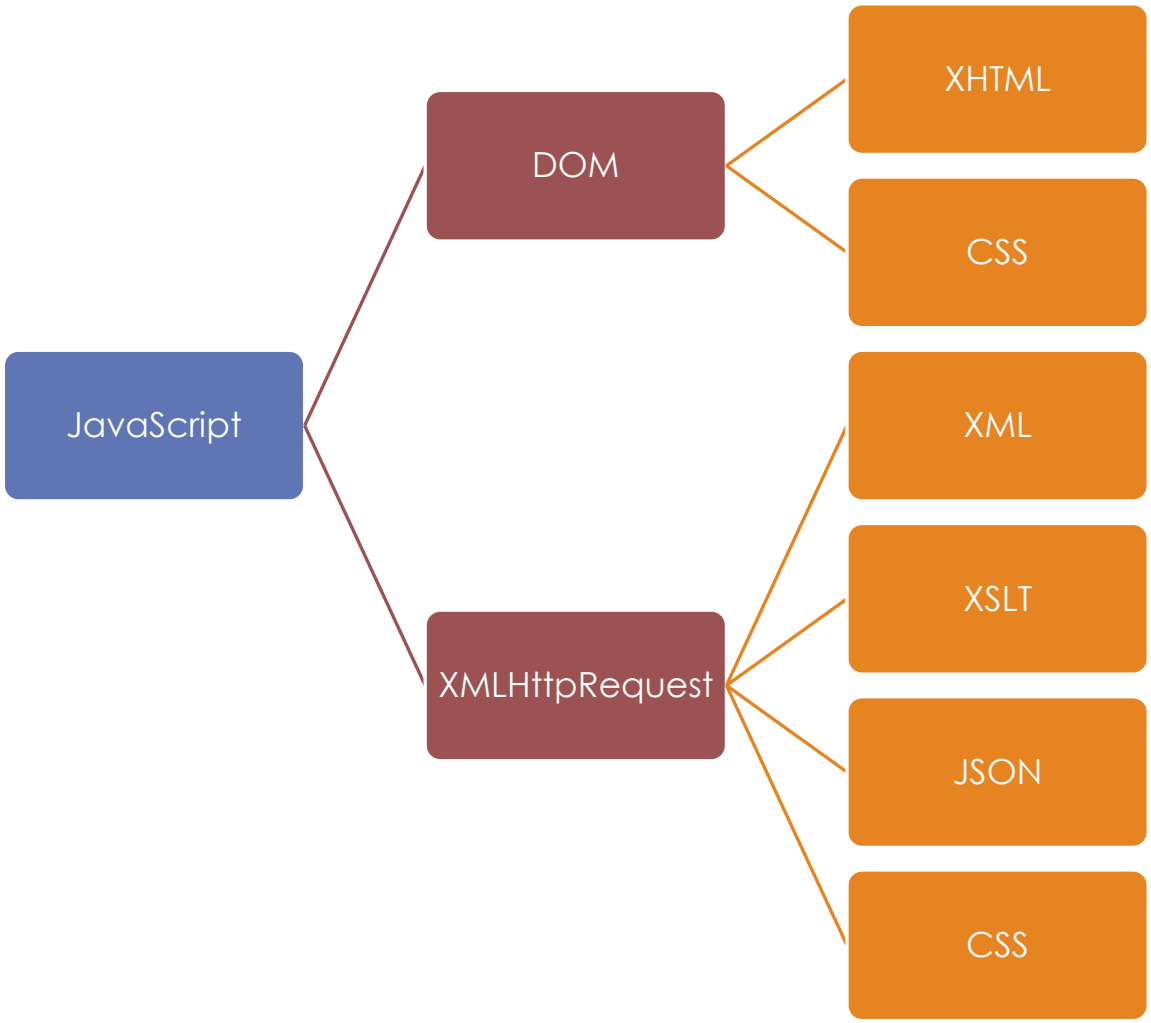

Figura 18. Definición de AJAX

El siguiente diagrama representa la forma tradicional que trabajan las aplicaciones WEB contra la forma en que trabajan las aplicaciones utilizando AJAX.

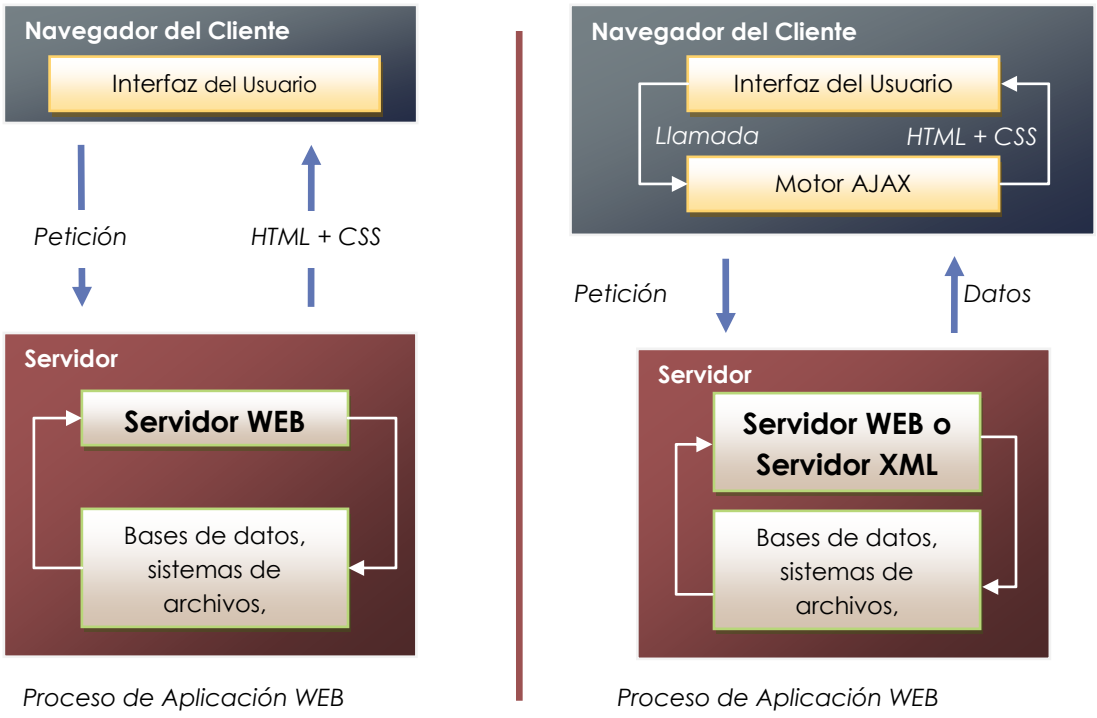

Figura 19. Comparación de proceso Clásico y proceso AJAX

La técnica tradicional para crear aplicaciones web no crea buena presentación si es que se tiene que enviar y recibir constantemente información del servidor, pues el cliente tendrá que esperar a que se recargue la página a cada solicitud de información, lo que puede llegar a ser molesto.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. Esta capa mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra esperando la recarga de todos los elementos de una página web.

Gracias a la utilización de esta técnica se pueden construir páginas web más avanzadas que pueden llegar a sustituir a las aplicaciones de escritorio.

Una vez que hemos definido esta parte que será la que proporciona los datos y el transporte de los mismos en el sistema, ahora vamos a definir las tecnologías con las que el usuario se relacionará directamente.

A continuación vamos a definir el software y las tecnologías que vamos a utilizar o bien, que ya se utilizan en el centro para el mantenimiento del sitio web.

A continuación presentamos una breve descripción de las tecnologías que utilizaremos en el diseño gráfico del portal.

Framework .NET

Es un conjunto de herramientas de desarrollo de software que permite un rápido desarrollo de aplicaciones, provee una serie de soluciones pre-codificadas que hacen un desarrollo de manera transparente entre la interfaz de usuario, acceso a datos, conectividad a base de datos, criptografía, aplicaciones web, algoritmos numéricos y comunicación de redes, y gestionan la ejecución de programas escritos específicamente para este framework.

El desarrollo de aplicaciones con este framework permite crear librerías que contengan imágenes, iconos, música, archivos de texto, etc.; Estos recursos forman parte del mismo ensamblado y permite portar recursos a cualquier equipo que ejecute el framework. Gracias a esta funcionalidad permite el encapsulamiento total de recursos y trabajar bajo el paradigma de orientación a objetos y desarrollo por capas.

Es la respuesta de Microsoft al creciente mercado de aplicaciones web y compite directamente a la plataforma Java de Sun Microsystem.

Permite la utilización de distintos lenguajes de desarrollo dentro de la misma aplicación a través de la generación de un código precompilado denominado código CIL (Common Intermediate Language) y un ensamblado.

Cuando la aplicación es ejecutada, el código CIL pasa por el CLR[16], el cual lo compila en tiempo de ejecución y lo transforma en código nativo que puede

[16] CLR. Common Language Runtime es un componente de la máquina virtual de la plataforma .NET que transforma código intermedio (generado de la compilación de los lenguajes de .NET) a código nativo para el sistema operativo. Es un compilador en tiempo de ejecución.

ejecutar el Procesador, siendo este proceso transparente para cualquier infraestructura de cómputo que ejecute el Framework.

Actualmente, existen implementaciones que permiten la ejecución de código CIL en otros sistemas operativos de software libre gracias al proyecto Mono.

El framework de .NET ya se incluye en Windows Server 2008, Windows Vista y Windows 7, y existe una versión "reducida" para la plataforma Windows Mobile. Sin embargo también puede ser instalada en Windows XP y Windows Server 2003.

ASP.NET

ASP.NET es un conjunto de herramientas que fue desarrollado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML[17]. Es la tecnología sucesora de la tecnología Active Server Pages (ASP).

ASP.NET está construido sobre el CLR, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

El modelo de despliegue de una página web, desarrollada en ASP.NET es el siguiente:

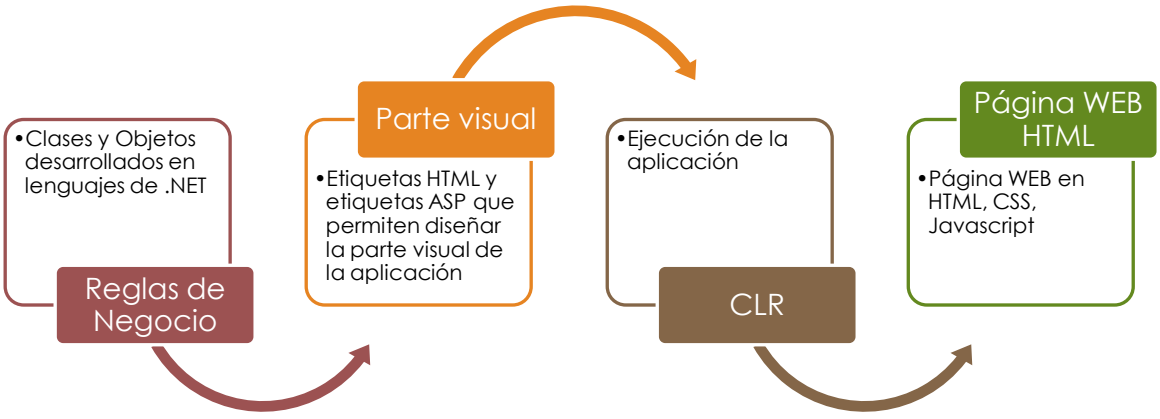

Figura 20. Proceso de ejecución de aplicación web

[17] XML. Acrónimo de eXtensible Markup Language. Es un metalenguaje desarrollado por la W3C que permite definir lenguajes para diferentes necesidades.

SQL Server 2008

SQL Server es un motor de base de datos producido por Microsoft basado en el modelo relacional y constituye una alternativa a otros potentes sistemas gestores de bases de datos como *Oracle*[18], *PostgreSQL*[19], *MySQL*[20] u otros. Los lenguajes que utiliza para consultas son T-SQL y ANSI SQL.

SQL posee dentro de su suite de herramientas:

- La capacidad de programación de tareas de mantenimiento, cargas masivas de información a través de procesos DTSX (a través de su *Integration Services*).
- Desarrollo y explotación de bases de datos *OLAP*[21] y *Minería de Datos*[22] (a través de su *Analisys Services*)
- Reportes directos de bases de datos, bases OLAP, estructuras de minería de datos y rendimiento del servidor (A través de su Reporting Services).
- Incluye un potente entorno gráfico de administración y ejecución de consultas (SQL Managment Studio).
- Total integración con *Microsoft Office*[23]
- Integración con *Microsoft Visual Studio*[24] para el desarrollo de bases de datos y la optimización de consultas.

[18] Oracle. Motor de base de datos relacional desarrollado por Oracle Corporation
[19] PostgreSQL. Motor de base de datos relacional libre desarrollada por PostgreSQL Global Development Group.
[20] MySQL. Motor de base de datos relacional subsidiada por Sun Microsystem y Oracle Corporation.
[21] OLAP. Del Acrónimo On-Line Analytical Processing es una solución utilizada en el campo de la Inteligencia de Negocios cuyo objetivo es agilizar la consulta de grandes cantidades de datos.
[22] Minería de Datos. Técnica de búsqueda de información que reside implícitamente en los datos.
[23] Microsoft Office. Es un paquete de programas para oficinas. Por defecto se incluye un procesador de Texto, una hoja de Cálculo y presentación de Diapositivas, aunque existen otros programas más complejos y de actividades específicas.
[24] Microsoft Visual Studio: Entorno de desarrollo de aplicaciones utilizando el framework de .NET.

La versión de SQL Server 2008 posee herramientas de administración muy potentes sobre los recursos del equipo donde se encuentra alojado, permitiendo con esto gran escalabilidad del sistema de información y una mayor explotación de datos a través de sus múltiples herramientas.

El lenguaje T-SQL es una extensión de SQL propietaria de Microsoft y *Sybase*[25], por lo que posee un mejor control del flujo del lenguaje, el uso de variables locales, soporte de funciones para cadenas, fechas, matemáticas y funciones desarrolladas por los usuarios y cambios en las funciones de borrado y actualización de registros (búsqueda de registros por consultas específicas para el control de cambios en tablas relacionales).

La elección de SQL Server se debe a la total integración con el *Framework de .NET*[26], la administración de recursos del servidor para explotación de datos en versiones posteriores del sistema e integración total con herramientas de Office (herramientas comúnmente utilizadas por los usuarios del centro).

Visual Studio2008

Es un entorno de desarrollo integrado (IDE[27], por sus siglas en inglés) para sistemas operativos Windows. Es un entorno multilenguaje pues soporta Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, y permite el desarrollo de extensiones para el soporte de otros, por ejemplo PHP y Java.

Visual Studio permite a los desarrolladores crear aplicaciones de escritorio, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET por lo que se pueden crear servicios que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

[25] Sybase. Compañía dedicada al desarrollo de tecnología para la movilización de grandes cantidades de datos.
[26] Framework de .NET. Conjunto de herramientas para el desarrollo ágil de aplicaciones.
[27] IDE. Acrónimo de Interface Developer Enviroment

Visual Studio 2008 tiene una completa integración con el framework 3.5 de .NET y permite crear servidores virtuales WEB, así como proyectos multicapa lo cual lo hizo la solución ideal para el sistema de información SOPHIA.

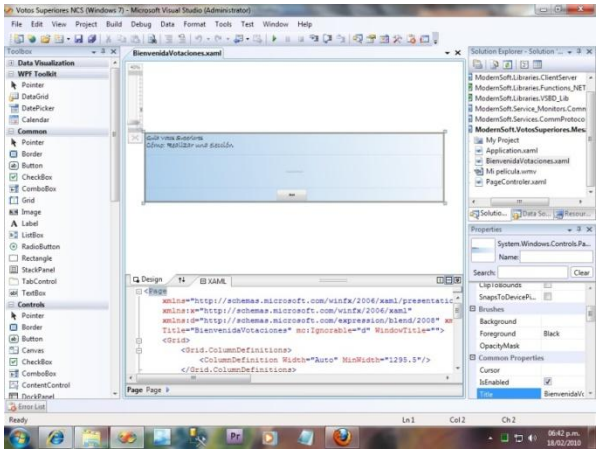

Figura 21. Interfaz de desarrollo

Adobe Flash

Es un programa de edición multimedia creado por Macromedia en 1996 y actualmente desarrollado y distribuido por la empresa Adobe. Flash se ha convertido en la herramienta más popular para agregar animaciones e interactividad a las páginas web.

Flash puede manipular gráficos vectoriales[28] e imágenes en mapa de bits[29] y soporta streaming bidireccional de audio y video. Adicionalmente Flash incorpora

[28] En estos una imagen es representada a partir de líneas (o vectores) que poseen determinadas propiedades (color, grosor, etc.). la calidad de este tipo de gráficos no dependen del nivel de acercamiento o de la resolución con la cual se esté mirando el gráfico, ya que la computadora traza automáticamente las líneas, adaptándolas a cualquier cambio.

[29] Se asemejan a una especie de cuadrícula en la cual cada uno de los cuadrados (píxeles) muestra un color determinado. La información es guardada individualmente para cada píxel y es definida por las coordenadas y color de dicho píxel. Estos gráficos son dependientes de la variación del tamaño y resolución, pudiendo perder calidad al modificar sucesivamente sus dimensiones.

un lenguaje de script[30] llamado Action Script, el cual expande las posibilidades de los proyectos.

Para ejecutar un programa en esta tecnología es necesario instalar una herramienta llamada Flash Player, que es una aplicación que permite la reproducción de contenido multimedia. Este funciona tanto en aplicaciones de escritorio como en ambientes web a través de un navegador.

Action Script

Es un lenguaje de programación orientado a objetos, utilizado especialmente en aplicaciones web animadas realizadas en el entorno Adobe Flash.

Es un lenguaje de script, esto significa, que no requiere de la creación de un programa completo para que la aplicación alcance los objetivos para los que fue desarrollada. Actualmente se encuentra en la versión 3.0

Entre el material generado por el CEAQ se desarrollan modelos tridimensionales del cuerpo humano, pero tienen la limitante de que para poder ser visualizados se necesita equipo de cómputo de muy alto rendimiento, algo que la mayoría de las personas no posee, por eso tomamos la decisión de distribuirlos por medio de videos y además crear modelos de baja resolución con los que los visitantes al portal del CEAQ puedan interactuar.

Análisis y Diseño

Diagrama de Casos de Uso

Un diagrama de casos de uso tiene como función proporcionarnos una visión sobre el comportamiento de un producto de software y nos permite poder ir construyendo la documentación necesaria para la etapa de desarrollo.

Un buen diagrama de casos de uso nos permitirá anticiparnos a dudas o errores durante el flujo de información, lo que se verá reflejado al ahorrar costos y corregir

[30] También conocido como lenguaje interpretado, es un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete.

el camino mientras no se ha empezado a escribir el sistema, generando un costo menor de corrección.

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista de los usuarios finales, por lo que nos permiten identificar los requerimientos funcionales y representan las funciones que el sistema puede ejecutar.

Para poder realizar un diagrama de casos de uso que sea funcional para el proyecto y generé una utilidad, tenemos que seguir los siguientes pasos:

1. Identificar a los actores del negocio. Un actor del negocio representa un rol jugado por alguna persona externa al negocio y que interactúa o se relaciona con él. Un actor se beneficia o se afecta por los resultados del proceso.
2. Identificar casos de uso del negocio. Un caso de uso identifica un proceso específico del negocio que produce un resultado de valor medible y esperado por un actor en particular.
3. Modelo de casos de Uso. Muestran la agregación de procesos entre actores y casos de uso del negocio. Esté modelado muestra el contexto del negocio y muestra las asociaciones entre los actores y los casos de uso.

Cabe mencionar que un diagrama de casos de uso no constituye un Diagrama de Flujo de Datos, por lo tanto, no se representa la secuencia entre los procesos.

Para este proyecto hemos identificado a 6 actores principalmente, estos son:

- **Administrador de SOPHIA.** Es la persona encargada de los sistemas de información del CEAQ.
- **Investigador.** Es un usuario que pertenece al CEAQ y sus funciones en el centro son generar material sobre sus investigaciones, proponer nuevas actividades y apoyar académicamente a la generación del material multimedia.

- **Profesor.** Es un usuario que pertenece a la Facultad de Medicina, pero no al CEAQ, su labor es impartir clases a los alumnos y generar material didáctico para el aprendizaje.
- **Alumno.** Es un usuario que pertenece a la Facultad de Medicina, pero no al CEAQ, su labor es utilizar el material que se genera en pro de su aprendizaje. Este es el usuario que nos genera la retroalimentación de la evaluación del material publicado.
- **Apoyo de Servicio Social de Sistemas.** Es un usuario que pertenece al CEAQ y apoya al centro en la administración de los sistemas de información y a la generación del material multimedia.
- **Público en general.** Es cualquier usuario que aprovecha los artículos y la publicidad del centro en beneficio propio.

De estos actores hemos encontrado quince casos de uso que nos permiten delimitar el sistema de información de SOPHIA. Estos casos de uso son resultado del documento visión y de la especificación de los requerimientos.

Los casos de uso identificados son:

1. **Gestionar Material Publicado.** Se refiere al proceso principal de autorización de material público.
2. **Administrar Comentarios.** Se refiere a la revisión de los comentarios dejados por los usuarios activos de SOPHIA.
3. **Administrar Artículos.** Se refiere a la creación, modificación y publicación de notas de interés, noticias y artículos escritos por los responsables de la generación de información.
4. **Administrar Cuestionarios.** Se refiere a la creación y publicación de elementos de evaluación que proporcionen al CEAQ métricas de análisis de sus usuarios y estos resultados sirvan de retroalimentación para proporcionar información más efectiva.
5. **Administrar Encuestas.** Se refiere a la creación y publicación de sistemas de medición de preferencias, para que los usuarios retroalimenten al CEAQ sobre el material generado.

6. **Administrar Sitios de Interés.** Permite al centro mantener informados a sus usuarios sobre nuevos sitios que los apoyen en la investigación de sus áreas de estudio.
7. **Administrar Usuarios.** Se refiere al control de acceso para realizar actividades dentro de SOPHIA.
8. **Administrar Contenido Multimedia.** Se refiere al proceso en el cuál el CEAQ actualiza sus imágenes, videos y modelos tridimensionales actualizados.
9. **Escribir Comentarios.** Se refiere al proceso mediante el cual, los usuarios activos retroalimentan al personal del CEAQ.
10. **Resolver Cuestionario.** Se refiere a los procesos en el cuál, los alumnos, proporcionan al centro métricas para evaluar su contenido y a sus usuarios activos.
11. **Ver Videos.** Se refiere a la visualización de videos utilizando tecnología de Streaming.
12. **Leer Artículos.** Permite la lectura de material publicado, así como su búsqueda y presentación dentro de SOPHIA.
13. **Ver Contenido Tridimensional.** Se refiere a la utilización del material tridimensional a través del sitio web.
14. **Ver Imágenes.** Se refiere a la visualización de imágenes a través de galerías por tema.
15. **Ver RSS.** Es el proceso de publicación de información utilizando clientes de servicio RSS.

El diagrama de casos de uso generado se presenta dentro de la sección de apéndices, y representa la relación que existe entre los actores y los casos de uso.

De este diagrama podemos concluir los alcances de SOPHIA, y nos ayuda a identificar quienes en el sistema son entidades que poseen características y acciones específicas.

Hemos identificado 7 flujos principales, cada uno es una entidad primaría de SOPHIA, estos son:

1. Usuarios
2. Noticias
3. Comentarios
4. Sitios de Interés
5. Cuestionarios
6. Multimedia
7. Encuestas

Para cada una de estas entidades es necesario conocer su flujo de información, para eso nos vamos a apoyar en otro diagrama que se utiliza para RUP, este es el diagrama de estados.

Diagrama de Estados

Los diagramas de estado muestran los diferentes pasos de una entidad durante su vida y los estímulos que provocan sus transiciones.

Para SOPHIA, vamos a generar un diagrama de estados por cada entidad que hemos identificado, con el fin de generar un flujo sobre los datos que van a intercambiarse y poder analizar que eventos detonan el cambio de estado.

El diagrama de estados nos permitirá entender que acciones o métodos deberá tener cada una de estas entidades, que posteriormente se convertirán en clases y estructuras de datos para el almacenamiento de datos.

Vamos a analizar la gestión de usuarios. La entidad que define este proceso es el Usuario y debe seguir una serie de pasos para poder utilizarse dentro de SOPHIA. Considerando el caso donde el usuario no existe en el sistema, los pasos son los siguientes:

1. El usuario realiza una solicitud al administrador de SOPHIA

2. El administrador gestiona a los usuarios del sistema (Altas, Bajas, Modificación de Datos y Recuperación de Contraseñas)
3. Si es un alta de usuario, se capturan sus datos y se le asigna un **Perfil.**
4. Si es una baja de usuario, se cancela a este usuario, sin borrarlo del sistema, esto por cuestiones de auditoría.
5. Si es una modificación, se capturan sus datos correctos o se le reasigna un **Perfil.**
6. Si es una recuperación de contraseña, se le notifica al sistema que el usuario requiriere dar de alta una nueva contraseña.
7. Una vez que se agregaron/actualizaron los datos del usuario, se le notifica al mismo que ahora puede hacer uso del sistema.
8. El usuario ingresa al sitio de SOPHIA y teclea su correo electrónico y como contraseña, su mismo correo electrónico.
9. El sistema valida que exista un usuario registrado con ese correo y le pide confirmar su fecha de nacimiento.
10. Se actualiza la contraseña y el usuario ahora puede utilizar los servicios de SOPHIA.

Este proceso se resume en un diagrama de estados de la siguiente forma.

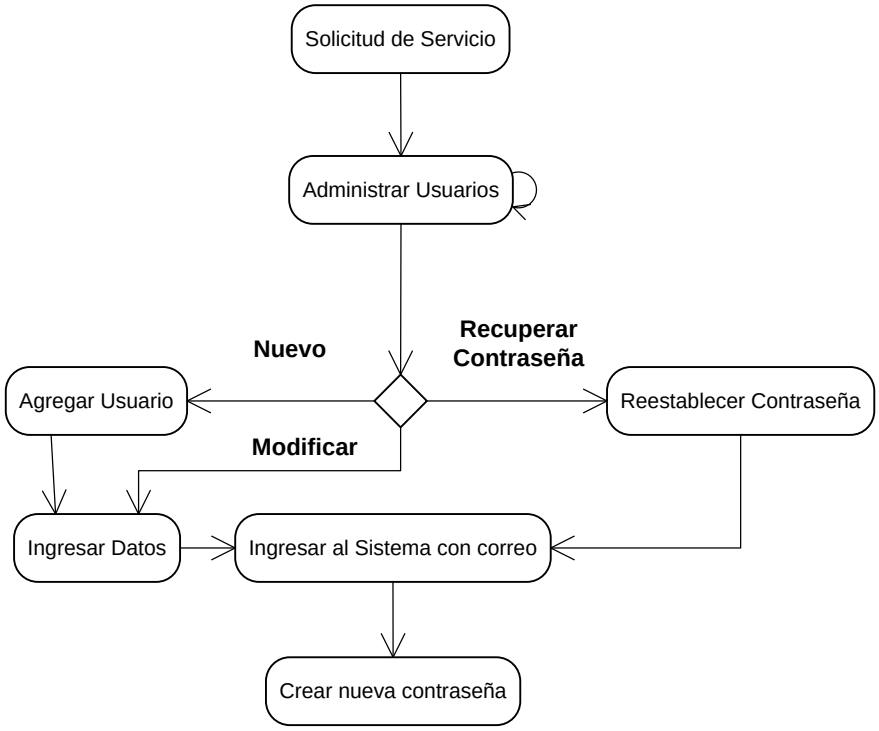

Figura 22. Diagrama de estados de Usuarios

De este análisis encontramos una entidad llamada Perfil, esta es una característica de cada usuario, pero es finita, por lo que la consideraremos un **catálogo.** Los catálogos nos permitirán mantener integridad en nuestra información, sin embargo podremos verlos en el diseño de clases y estructuras de almacenamiento.

Ahora analizaremos a la entidad Noticia, vamos a considerar el proceso para la creación, modificación y publicación de una noticia. Los pasos son los siguientes:

1. Un **Usuario** comienza a escribir una nueva noticia que desea publicar en el sistema de SOPHIA.
2. El **Usuario** decide si desea publicar o guardar el artículo como un borrador. De acuerdo a la elección el Artículo toma un nuevo **Estado.**
3. El **Usuario** puede realizar actualizaciones a su artículo mientras que no ha sido publicado en el sitio.

4. Una vez que el **Usuario** finaliza y publica, el Articulo cambia de **Estado**
5. Si el Artículo ha sido liberado, un auditor de contenido lo autoriza siempre y cuando cumpla las reglas necesarias para estar bajo el dominio de la UNAM.
6. Si el artículo fue autorizado, se publica en el sitio web y en la fuente RSS.
7. Si los demás **Usuarios** desean dejar una nota, publican un **Comentario.**

El diagrama se representa de la siguiente forma:

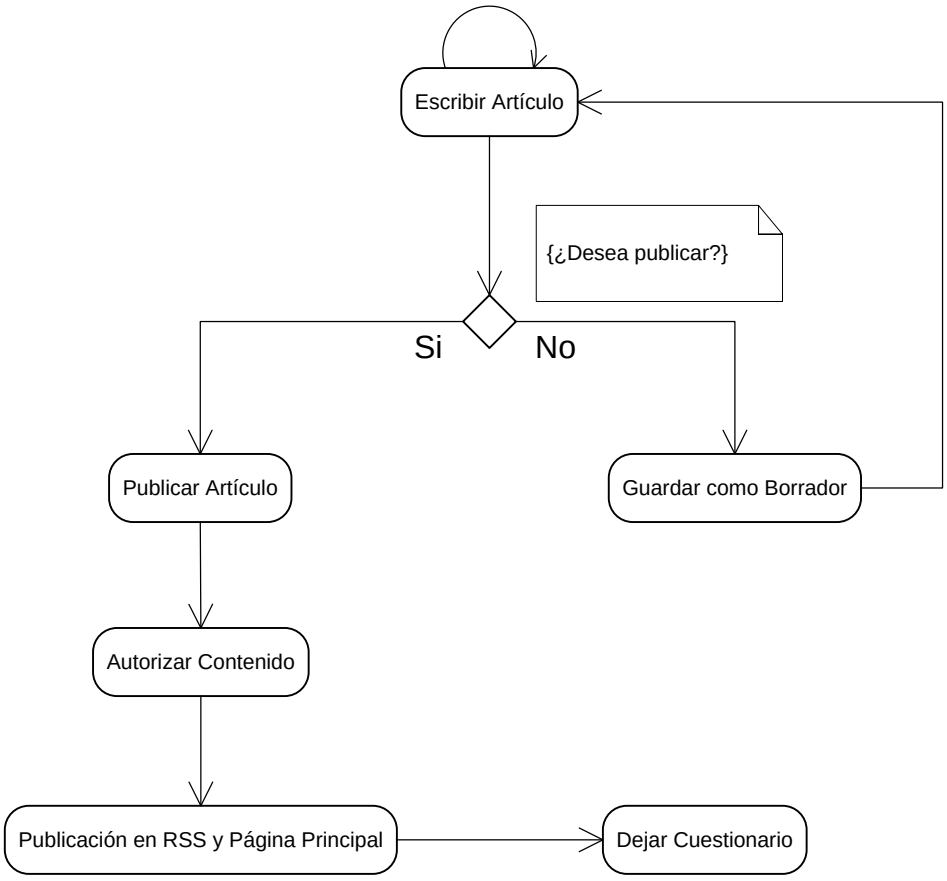

Figura 23. Diagrama de estados de artículo

En este diagrama podemos encontrar como se identifica a una **Noticia**, con un **Usuario** porque nos permite visualizar cómo se van relacionando las entidades entre sí. También podemos encontrar que una Noticia pasa por distintos estados, muy diferentes a los del Usuario, por lo que, aquí tenemos una entidad.

Ahora analizaremos el flujo de un comentario. Para poder realizar un comentario se requieren seguir los siguientes pasos:

1. Ingresar al sistema de SOPHIA
2. Navegar por el contenido que ofrece.
3. Si el **Usuario** desea comunicarse con el CEAQ, agregar un **comentario**

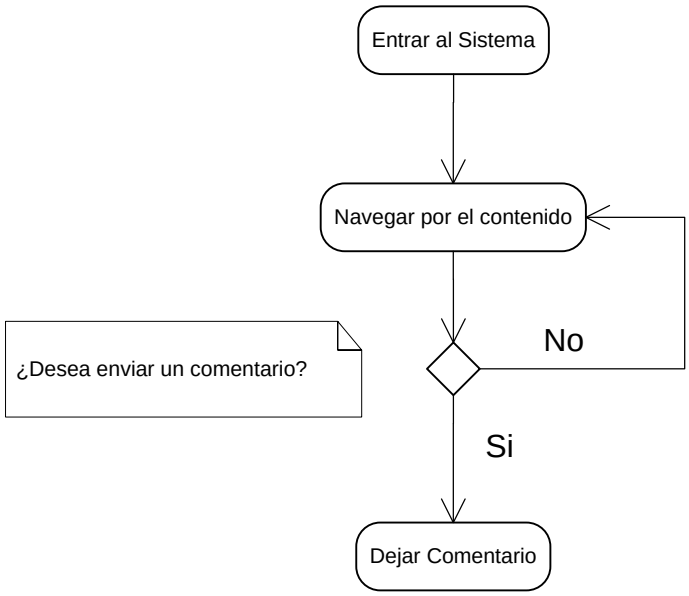

Figura 24. Diagrama de estados de Comentarios

El flujo asocia a la entidad Comentario con el Usuario, no puede existir un comentario de un Usuario público, esto nos ayudará a definir el nivel de acceso de la información.

El siguiente flujo que vamos a analizar es la gestión de sitios de Interés. Un sitio de interés solo puede ser agregado o modificado por el administrador de SOPHIA o su personal de apoyo. El proceso debe ser simple, pues lo que buscamos en este sistema es que los usuarios finales inviertan mayor tiempo en la búsqueda de herramientas que apoyen a los alumnos, docentes y administradores y el mínimo de tiempo en el entendimiento de la plataforma tecnológica.

El proceso se puede dividir en los siguientes pasos:

1. El administrador verifica que los sitios de interés contengan acceso a información útil y actualizada para los usuarios.
2. El administrador puede agregar un nuevo sitio, con una descripción con estilos e imágenes que sean atractivas para el usuario. Una vez terminado, se publica en el sitio.
3. El administrador puede modificar la descripción o el título de un sitio web buscando hacerlo más atractivo o proporcionando mejor información a sus usuarios.
4. El administrador puede quitar un sitio recomendado del sitio web.

Es proceso tiene un impacto inmediato en el sitio web, por lo que toda modificación es realizada en tiempo real y se debe tener cuidado, pues son fuentes externas de las cuales no se tiene control, por lo que pueden desaparecer o mostrar errores. Por esto, la gestión sobre los sitios de interés debe ser una tarea constante del o de los administradores.

El proceso puede analizarse gráficamente con el siguiente diagrama:

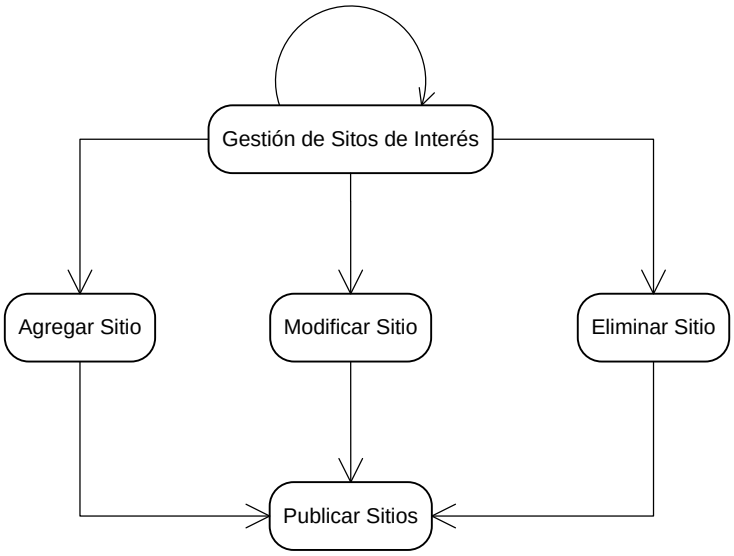

Figura 25. Diagrama de estados de sitios de interés

Esta entidad no está relacionada con ninguna otra, por lo que, para controlar un cambio en ella solo se debe considerar garantizar información real en el sitio, pero no impacto sobre otros objetos de SOPHIA.

Ahora vamos a analizar el proceso de la entidad Cuestionario, este es un proceso interesante dado que vamos a descubrir como a partir de él se generan nuevas entidades que dependen de Cuestionario y forman un árbol dinámico de objetos, por lo que se aprovechará al máximo la utilización de un lenguaje con paradigma orientado a objetos.

El proceso para la gestión de un cuestionario inicia con la necesidad de plantear a los usuarios de SOPHIA una serie de preguntas para evaluar sus conocimientos sobre algún tema. En primer lugar debemos especificar que, la información del sistema fluye a todos los usuarios registrados, por lo que, en el caso de dar de alta un cuestionario, todos los alumnos dados de alta podrán contestarlo y proporcionaran a los administradores, profesores e investigadores métricas de evaluación.

Vamos a definir el proceso paso a paso:

1. Se agrega un **Cuestionario** al sistema, con un nombre que identifique el tema a evaluar.
2. Un **Cuestionario** está constituido por una serie de **Preguntas** (no existe un límite de preguntas para un cuestionario), las cuales se tendrán que ir capturando.
3. Cada pregunta tiene, por regla de negocio, **3 Respuestas Incorrectas** y **1 Respuesta Correcta**, por lo que se deben capturar las 4 respuestas, de acuerdo a su categoría.

4. Si el número de preguntas es el total planeado, se finaliza la captura. Si faltan preguntas o se desea modificar alguna. Se continúa con la modificación.
5. Si ya se ha finalizado con la captura se procede a la publicación del **Cuestionario**.
6. Mientras el cuestionario se encuentre publicado, todos los **Usuarios** con el perfil de alumno, podrán resolverlo.
7. Para resolver un cuestionario, el **Usuario** lo selecciona y de manera dinámica, se muestran en pantalla las **Preguntas** aleatoriamente, al igual que las **Respuestas**.
8. El **Usuario** selecciona sus respuestas y las envía al CEAQ a través de SOPHIA.
9. El sistema almacena las respuestas y genera el **Resultado**.
10. Cuando el administrador del sitio considera que el cuestionario ha finalizado se lo indica al sistema, a través de su módulo de gestión.
11. Finalizado, el administrador y profesores pueden ver los resultados de la evaluación.

Gráficamente lo podemos ver de la siguiente forma:

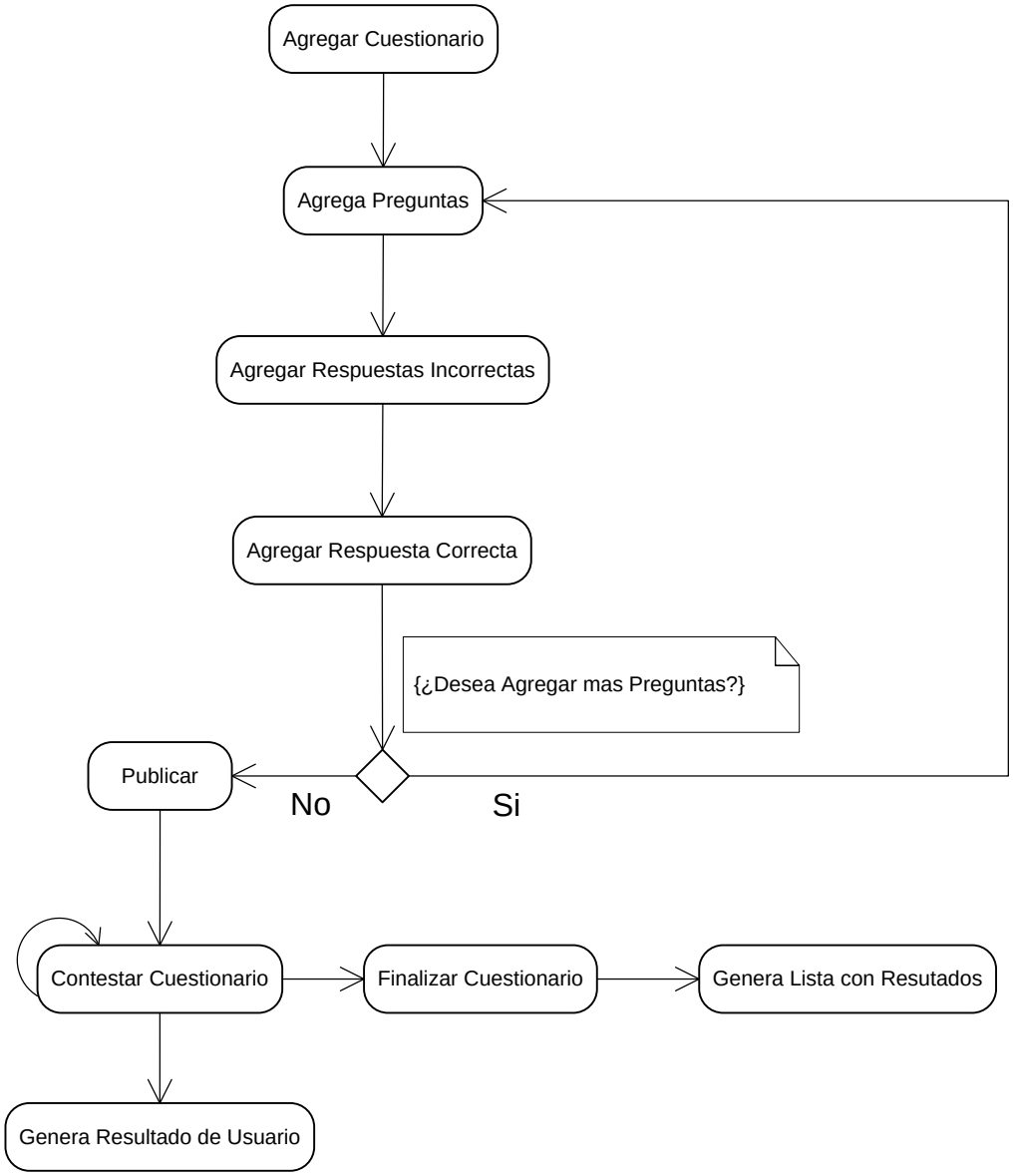

Figura 26. Diagrama de estados de Cuestionarios

De este proceso descriptivo, podemos encontrar nuevas entidades y reglas, para iniciar, encontramos que un Cuestionario solo puede ser resuelto por un Usuario con Perfil de Alumno.

En cuanto a las nuevas entidades, podemos ver que el Cuestionario está formado por una nueva entidad llamada Pregunta y cada Pregunta está formada por 3

entidades llamadas Respuesta Incorrecta y por una llamada Respuesta Correcta. También nos damos cuenta que cada Cuestionario resuelto, genera una entidad denominada Resultado que tiene un Usuario asociado. Cabe mencionar que, no se definen diagramas de estados para estas 4 nuevas entidades porque están implícitas en el proceso de cuestionario.

Para el proceso de gestión de material multimedia, vamos a definir un proceso sencillo, buscando hacerlo de forma masiva, para esto se ha propuesto crear una estructura en XML que almacene los datos de los objetos multimedia (la fuente del archivo, su nombre y descripción) y puedan agruparse en categorías para una mejor administración.

Con esto, lograríamos que un administrador pudiera hacer cargas masivas con este archivo y no requiera dar de baja el sitio para su actualización, esto, siguiendo el siguiente proceso:

1. Generación del material multimedia (Videos, Imágenes y modelos Tridimensionales).
2. Gestión de archivos multimedia en el sitio web de SOPHIA.
3. Actualización de archivos XML con los cambios en el material.
4. Publicación automática de las galerías.

El proceso de gestión de material multimedia se puede representar en el siguiente diagrama de estados:

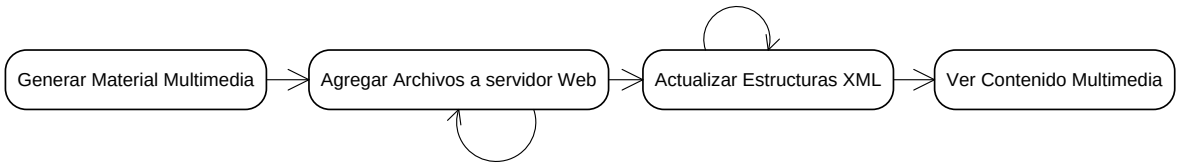

Figura 27. Diagrama de estados de Multimedia

Con este proceso se buscó hacer de forma muy sencilla, dado a que la experiencia en otros proyectos nos indica que gestionar archivo por archivo es un proceso tedioso que después de la carga inicial tarda mucho tiempo en ser actualizado a causa del tiempo que se requiere invertir.

En la etapa del desarrollo se van a definir las estructuras XML, pero desde este momento se va a considerar dentro de las estructuras de almacenamiento.

La última entidad que vamos a definir con un diagrama de estados son las Encuestas; este proceso permite al CEAQ obtener información sobre opiniones sobre preguntas específicas que el centro quiera proponerles a sus usuarios.

El proceso de votación permitirá a los usuarios expresar su opinión sobre una pregunta, con respuestas propuestas por el mismo centro.

El proceso inicia con la necesidad de conocer la tendencia sobre una pregunta en específico que se plantee el CEAQ. Los pasos a seguir son los siguientes:

1. El administrador de SOPHIA agrega una encuesta y le pone una fecha de vigencia.
2. El administrador publica la encuesta en el sitio web.
3. El administrador de SOPHIA agrega las posibles **Opciones.**
4. Mientras la encuesta esta activa, el usuario puede votar y ver los resultados parciales.
5. Cuando la encuesta caduca, pasa a un estado de inactiva y el administrador puede ver los resultados finales.

El diagrama de estados para la entidad Encuesta es el siguiente:

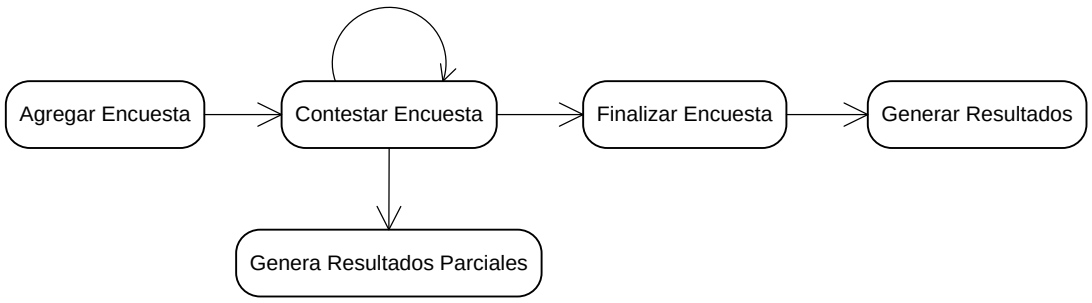

Figura 28. Diagrama de estados de Encuestas

Esta entidad no posee relación con otras entidades ya definidas debido a que la retroalimentación de los usuarios al CEAQ debe venir de todos los usuarios, con cualquier perfil o con acceso público. Sin embargo, si tiene relación con una nueva entidad llamada **Opción**.

Una vez definidos todos los procesos de las entidades primarias, debemos pasar a la definición del diagrama de clases y las relaciones que existen entre ellas.

Modelo de clases

Una vez que hemos analizado el flujo de datos de las entidades, como se procesan y se convierten en información útil para los usuarios, podemos determinar nuestro modelo de clases.

Del modelo anterior han saltado entidades que de alguna forma están relacionadas entre sí, y también derivan a otras o bien, algunas poseen características similares, estas relaciones darán vida al Sistema de Información de SOPHIA, pero en esta etapa es donde debemos poner más atención, pues un error de entendimiento generará un producto de mala calidad que no será satisfactoria para la solución de reingeniería.

Debemos pensar que el producto final debe cumplir 3 objetivos principales:

1. Es necesario incluir todo aquello que se encuentra en el documento visión y debe cumplir cada uno de los requerimientos planteados en la lista.
2. El diseño debe ser robusto para que en la etapa de mantenimiento y generación de nuevos requerimientos, los sistemas de información mantengan su integridad y las reglas de negocio deben mantenerse a menos que por una necesidad deban ser cambiados.
3. La información que fluye del usuario al centro de almacenamientos, debe ser integral en todo momento, no nos interesan los datos, nos interesa el análisis de estos.

La propuesta para el CEAQ es integrar una serie de capas que encapsulen los sistemas de almacenamiento y cada capa tenga sus propios filtros de acuerdo a su función. La idea es, que el usuario no tenga acceso directo a los sistemas de almacenamiento, sino a una capa externa que proporcionará lo que el usuario necesite de acuerdo a los requerimientos.

Cada capa podrá acceder a su capa inmediata interior, y únicamente podrá ser accedida por su capa inmediata exterior.

Las capas se han definido de la siguiente forma:

- **Zona 0. Base de Datos y Sistema I/O.** Son los sistemas de almacenamiento que forman parte del repositorio de información de SOPHIA. Las actualizaciones de ésta dependen del proveedor del motor y del sistema operativo (SQL Server 2008 y Windows 2008 respectivamente). Son considerados la Zona 0 porque es la parte más importante de nuestro modelo y es la capa que será explotada por los usuarios, lo que representa una oportunidad para generar una utilidad al CEAQ.
- **Capa 1. Procedimientos Almacenados y Manejo de Archivos.** Son el medio de acceso a los sistemas de información, son secuencias de SQL y accesos a archivo. Las secuencias SQL son encapsuladas en Procedimientos Almacenados y el manejo de archivos son controlados por clases nativas del Framework.

- **Capa 2. Control de Acceso.** Este es el primer nivel desarrollado utilizando el Paradigma Orientado a Objetos y provee al sistema un conjunto de herramientas para ingresar de forma transparente a los procedimientos almacenados y sistemas de archivos de SOPHIA. Su diseño facilita las actualizaciones del manejador de DB y del sistema de I/O, pues solo es necesario modificar este nivel y mientras se respeten los métodos y atributos de las clases, el cambio será transparente.
- **Capa 3. Entidades.** Una vez que identificamos las entidades que definen a SOPHIA, se crearan una serie de clases que proveerán al sistema de la funcionalidad necesaria para conservar la integridad de los datos. Esta capa representa el modelado de la base de datos.
- **Capa 4. Modelo de Negocios y Objetos Auxiliares**. En este nivel se definen las reglas de negocio, los flujos de información y provee un conjunto de herramientas que permiten cambiar las estructuras de datos por código HTML o XML para mostrar en un browser. Esta capa controla los eventos de SOPHIA y controla la sesión de los usuarios para mantener el control de Acceso.
- **Capa 5. Front de la aplicación.** Es el nivel donde el usuario convive directamente con la aplicación. Lo nutre y lo provee de información procesada de acuerdo a sus requerimientos y necesidades. En esta capa se tienen hojas de estilo CSS, código HTML y código Javascript.

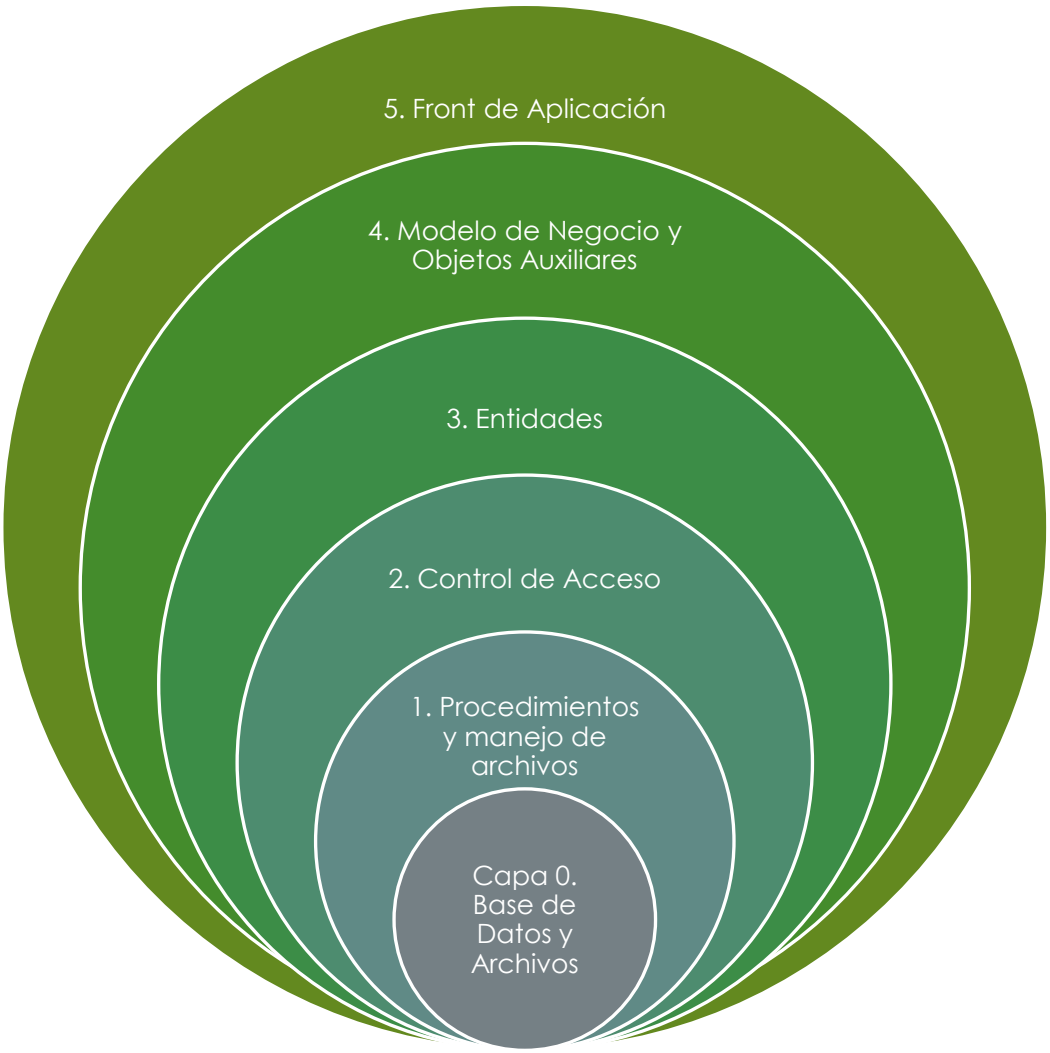

Figura 29. Modelo de Capas de SOPHIA

Sin embargo, cabe mencionar que debido a que estamos realizando innovaciones en el área de las metodologías de diseño de software, vamos a aplicar las reglas del modelo de clases por capa, pues de acuerdo a la definición, cada capa solo tiene relación con su inmediata interna y su inmediata externa.

Cabe mencionar que todo modelo planteado en esta etapa, estará sujeto a mejoras, pues la filosofía de RUP nos indica que siendo este un proceso iterativo incremental, al finalizar cada interacción, se regresa a la etapa de diseño para cambiar los modelos en pro de llegar a un sistema cada vez más robusto.

Sin embargo, es hora de definir reglas para tener un sistema con un número de iteraciones finito, por lo que, las reglas son las siguientes:

1. En cada iteración, no se podrán cambiar entidades a menos que éstas generen un beneficio en el requerimiento planteado.
2. Solo se cambiara el modelo de clases y el modelo relacional.
3. Se podrán agregar métodos y propiedades a las clases siempre que éstas cumplan con el modelo de capas.
4. No se pueden agregar requerimientos a los planteados en el documento visión.
5. No se pueden modificar procesos de tratamiento de información.
6. Todo cambio será documentado y actualizado en los diagramas finales.

Estas seis reglas permitirán que la metodología de RUP, apoyada por las buenas prácticas de XP, cree un sistema estable, que mientras se desarrolle sea probado y modificado para eliminar el mayor número posible de errores, creando un producto de buena calidad e invirtiendo la mayor parte en el diseño y desarrollo de la aplicación.

Modelo Entidad Relación

El Modelo Entidad-Relación (E-R[31]) es un concepto de modelado para bases de datos que fue propuesto por el Dr. Peter Chen[32] en 1976. Mediante éste, se pretende 'visualizar' los objetos que pertenecen a la Base de Datos como entidades[33] las cuales tienen ciertos atributos y se vinculan mediante relaciones.

Es una representación conceptual de la información. Mediante una serie de procedimientos se puede pasar del modelo E-R a otros, como por ejemplo el modelo relacional.

El modelado entidad-relación es una *técnica* para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más

[31] Del inglés Entity Relationship
[32] Profesor de la Universidad de Louisiana cuyos trabajos fueron piedras angulares de la Ingeniería de Software.
[33] Corresponde al concepto de objeto de la Programación Orientada a Objetos.

utilizada. En el caso de este sistema, seguimos los siguientes pasos para poder obtenerlo.

1. Se parte de una descripción textual del problema o sistema de información a automatizar (En nuestro caso partimos del documento visión y la definición de los requerimientos).
2. Analizamos los sustantivos y verbos que podemos localizar de la etapa anterior.
3. Los sustantivos son posibles entidades o atributos, confirmados por los diagramas anteriores y definidos en el diagrama de estados.
4. Los verbos son posibles relaciones.
5. Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.
6. Se elabora el diagrama (o diagramas) entidad-relación.

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo que se pueda cargar directamente en una base de datos.

Antes de iniciar con el proceso para obtener este modelo, debemos entender y definir los componentes del mismo.

Entidad

Aunque hemos utilizado ya este concepto, es necesario definirlo propiamente para este diagrama. Representa una "cosa" u "objeto" del mundo real con existencia independiente, es decir, se diferencia unívocamente de cualquier otro objeto o cosa, incluso siendo del mismo tipo.

Por ejemplo, un usuario, a pesar que en concepto, todos son iguales, tienen atributos propios (nombre de usuario, contraseña, etc.)

Atributos

Los atributos son las propiedades que describen a cada entidad en un conjunto de entidades.

Un conjunto de entidades dentro de una entidad, tiene valores específicos asignados para cada uno de sus atributos, de esta forma, es posible su identificación univoca.

Cada una de las entidades pertenecientes a este conjunto se diferencia de las demás por el valor de sus atributos. Nótese que 2 o más entidades diferentes pueden tener los mismos valores para algunos de sus atributos, pero nunca para todos.

Relación

Describe cierta dependencia entre entidades o permite la asociación de las mismas.

Por ejemplo, Un alumno Pedro posee el perfil de Administrador. Esto quiere decir que un elemento que pertenece a la entidad alumno (cuyo atributo de nombre es Pedro) está relacionado con la entidad Perfil (cuyo atributo que lo identifica es Administrador).

Correspondencia de cardinalidades

Dado una relación, en el que participan 2 o más conjuntos de entidades, la correspondencia de cardinalidad, indica el número de entidades con las que puede estar relacionada una entidad dada.

Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, la correspondencia de cardinalidades puede ser:

- **Uno a uno (1,1):** Una entidad de A se relaciona únicamente con una entidad en B y viceversa.

- **Uno a varios (1, n):** Una entidad en A se relaciona con cero o muchas entidades en B. Pero una entidad en B se relaciona con una única entidad en A.
- **Varios a uno (n, 1):** Una entidad en A se relaciona exclusivamente con una entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A.
- **Varios a varios (n, n):** Una entidad en A se puede relacionar con 0 o muchas entidades en B y viceversa.

Por ejemplo: Un usuario puede tener solo un perfil, pero un perfil puede tener muchos usuarios, esto nos indica que la relación es uno a varios (1, n).

De los análisis anteriores, ya habíamos encontrado las entidades; el propósito de este diagrama es crear una estructura de datos para almacenar la información de estas entidades. Para esto, es necesario documentar las relaciones que existen entre ellas, determinar los catálogos y por último, determinar su cardinalidad para optimizar el almacenamiento de los datos y garantizar información que conserve su integridad en todo momento.

Las entidades que vamos a analizar son:

- Usuario
- Comentario
- Sitios de Interés
- Cuestionario
    - Pregunta
    - Respuesta
- Encuesta
    - Opción
- Noticia

La entidad Multimedia no será analizada debido a que, como ya se vio anteriormente, se decidió considerar otro tipo de estructura de datos para esta información.

Primero vamos a plantear las nueve entidades encontradas y su relación que existe entre ellas, éste será el primer acercamiento al modelo de negocio transformado en un repositorio de datos, por lo que es importante incluir en él todas las reglas para que en un futuro no tengamos información errónea, o tengamos duplicidad en los datos.

El primer acercamiento es el siguiente:

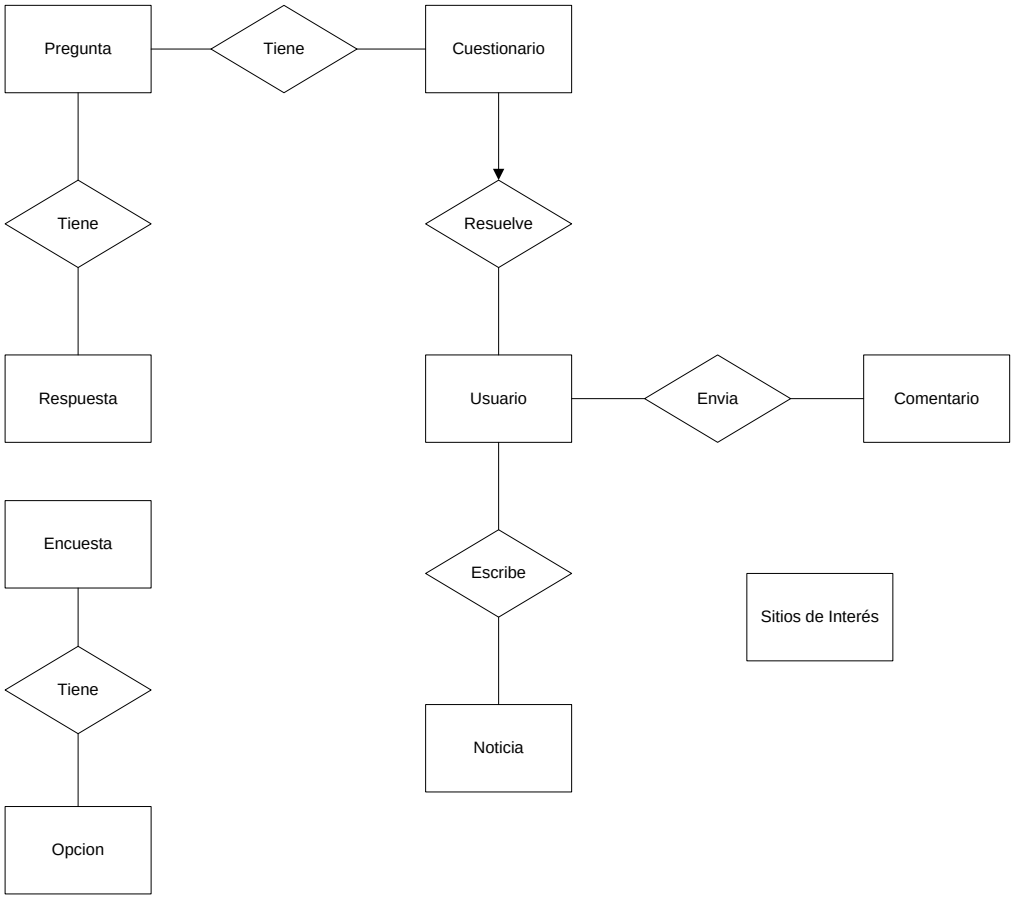

Figura 30. Diagrama de CHEN. Primer acercamiento

De aquí podemos determinar que las reglas de negocio nos indican las siguientes cosas:

- Un Usuario debe tener un **Perfil.**
- Una Noticia puede estar como Borrador, en Proceso de Autorización, Publicada o Rechazada, por lo tanto, tiene un **Estado**.

- Una Encuesta puede estar Activa, Finalizada o en proceso de Publicación, por lo tanto, tiene un **Estado.**
- Un Cuestionario puede estar en proceso de Publicación, Activo o Finalizado, por lo tanto tiene un **Estado.**

Esto nos indica que debemos tener 4 nuevas entidades que serán *catálogos* [34]de nuestras entidades.

Por lo tanto el diagrama cambia a la siguiente forma:

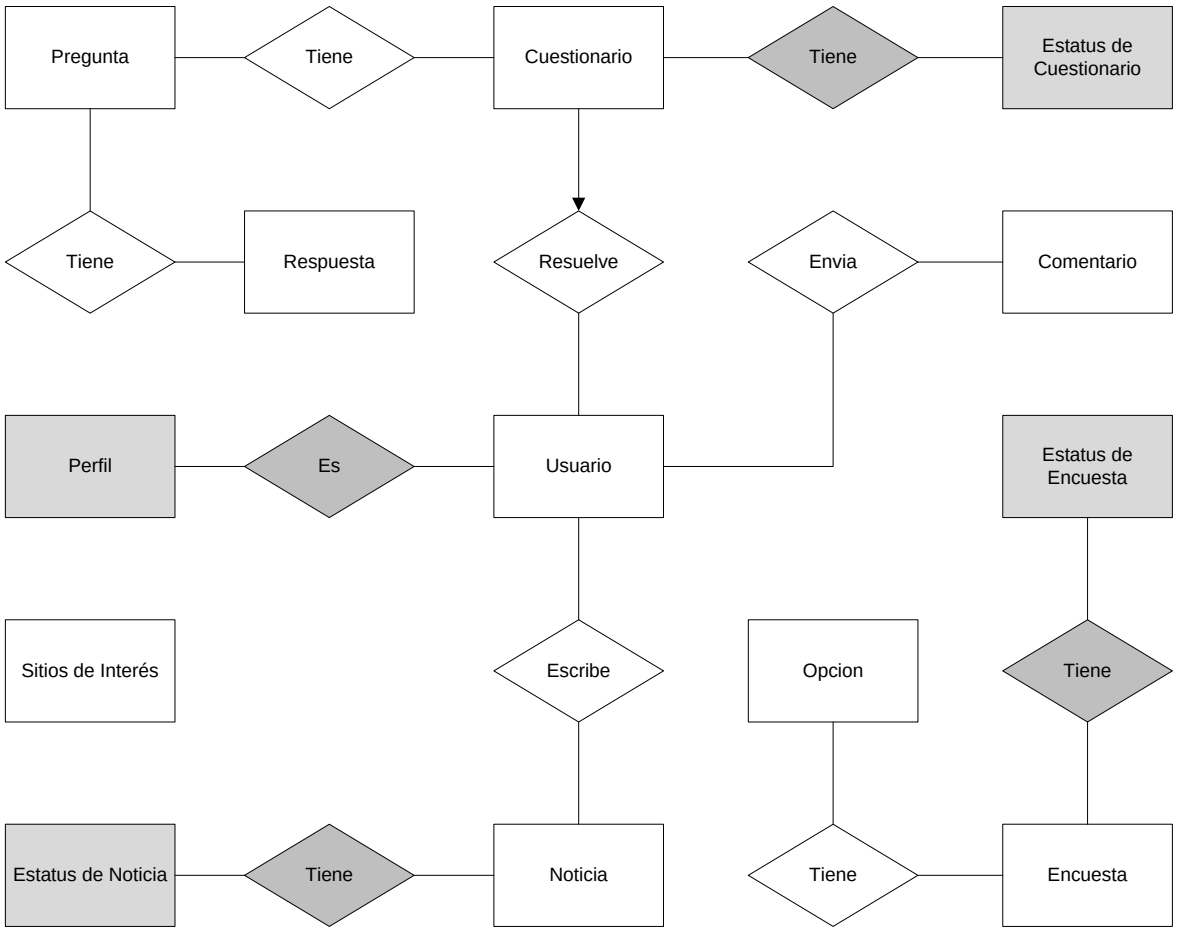

Figura 31. Diagrama de CHEN. Agregando primeras reglas de negocio.

[34] Catálogo. Elemento que define una característica de una entidad, dicha característica es fija y puede ser igual para uno o más elementos definidos con esa entidad.

Aquí podemos ver relacionadas las cuatro nuevas entidades, por lo que ahora tendríamos que calcular la cardinalidad del diagrama, pero vamos a crear un paso más radical, y de este diagrama crearemos un **modelo relacional**.

El **modelo relacional** para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados[35] y en la teoría de conjuntos[36]. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente.

En 1970 el Dr. Edgar Frank Codd, de los laboratorios IBM en San José (California), postuló 12 reglas que permitían determinar la fidelidad de un sistema relacional. Las reglas son las siguientes:

- **Regla 0.** Para que un sistema se denomine *Sistema de Gestión de Bases de Datos Relacionales*, este sistema debe usar exclusivamente sus capacidades relacionales para gestionar la base de datos.

- **Regla 1.** *Regla de la Información.* Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico mediante tablas y sólo mediante tablas.

  Por tanto los *metadatos* (diccionario, catálogo) se representan y se manipulan exactamente igual que los datos de usuario, usando quizás el mismo lenguaje (ejemplo SQL)

- **Regla 2.** *Regla del acceso garantizada*. Para todos y cada uno de los datos (valores atómicos) de una base de datos relacional se garantiza que son accesibles a nivel lógico utilizando una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

[35] Es un lenguaje matemático que permite en sistemas de cómputo hacer referencia directamente a 3 o más valores discretos, y no solo la limitación booleana de verdadero o falso.
[36] Es una división de las matemáticas que permite el estudio de una colección de objetos.

Cualquier dato almacenado en una base de datos relacional tiene que poder ser direccionado unívocamente. Para ello hay que indicar en qué tabla está, cuál es la columna y cuál es la fila (mediante la clave primaria).

- **Regla 3.** *Tratamiento Sistemático de valores nulos.* Se debe disponer de valores nulos (distintos de la cadena vacía, blancos, 0, etc.) para representar información desconocida o no aplicable de manera sistemática, independientemente del tipo de datos.

- **Regla 4.** *Catálogo dinámico en línea basado en el modelo relacional.* La descripción de la base de datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, igual que lo aplican a los datos normales.

- **Regla 5.** Regla del sublenguaje de datos completo. Un sistema relacional debe soportar varios lenguajes y varios modos de uso de terminal. Sin embargo, debe existir al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida, como cadenas de caracteres y que sea completo, soportando:
    - Definición de datos
    - Definición de vistas
    - Manipulación de datos (interactiva y por programa)
    - Restricciones de integridad
    - Restricciones de transacciones (*begin*, *commit*, *rollback*).

- **Regla 6.** *Regla de actualización de vistas.* Todas las vistas que son teóricamente actualizables se pueden actualizar también por el sistema.

- **Regla 7.** *Inserción, actualización y borrado de alto nivel.* La capacidad de manejar una relación base o derivada como un solo operando se aplica

no sólo a la recuperación de los datos (consultas), sino también a la inserción, actualización y borrado de datos.

- **Regla 8.** *Independencia Física de Datos.* Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cualesquiera sean los cambios efectuados, tanto en la representación del almacenamiento, como en los métodos de acceso.
  El modelo relacional es un modelo lógico de datos, y oculta las características de su representación física.

- **Regla 9.** *Independencia Lógica de Datos.* Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cualesquiera sean los cambios que se realicen a las tablas base que preserven la información.
  Ejemplos de cambios que preservan la información:
  - Añadir un atributo a una tabla base.
  - Sustituir dos tablas base por la unión de las mismas. Usando vistas de la unión se pueden recrear las tablas anteriores.

- **Regla 10:** Independencia de Integridad. Las restricciones de integridad específicas para una determinada base de datos relacional deben poder ser definidos en el sublenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.
  El objetivo de las bases de datos no es sólo almacenar los datos, sino también sus relaciones y evitar que estas restricciones se codifiquen en los programas. Por tanto en una base de datos relacional se deben poder definir restricciones de integridad.
  Cada vez se van ampliando más los tipos de restricciones de integridad que se pueden utilizar en los Sistemas de Gestión de Bases de Datos Relacionales, aunque hasta hace poco eran muy escasos.

Como parte de las restricciones inherentes al modelo relacional (forman parte de su definición) están:

- Integridad de Entidad: Toda tabla debe tener una clave primaria.
- Integridad de Dominio: Toda columna de una tabla contendrá valores exclusivamente de un determinado dominio (conjunto de valores válidos)
- Integridad Referencial: Toda clave foránea no nula debe existir en la relación donde es clave primaria.

- **Regla 11.** *Independencia de Distribución*. Una Base de Datos Relacional es independiente de la distribución. Las mismas órdenes y programas se ejecutan igual en una base de datos centralizada que en una distribuida.
  Esta regla es responsable de tres tipos de transparencia de distribución:
  - Transparencia de Localización. El usuario tiene la impresión de que trabaja con una base de datos local. (Regla de Independencia Física)
  - Transparencia de Fragmentación: El usuario no se da cuenta de que la relación con que trabaja está fragmentada. (Regla de Independencia Lógica).
  - Transparencia de Replicación: El usuario no se da cuenta de que pueden existir copias (réplicas) de una misma relación en diferentes lugares.

- **Regla 12.** *Regla de la No subversión*. Si un sistema relacional tiene un lenguaje de bajo nivel (un registro a la vez), ese bajo nivel no puede ser usado para subvertir (saltarse) las reglas de integridad y las restricciones expresadas en los lenguajes relacionales de más alto nivel (una relación a la cada vez).

La ventaja de trabajar con este modelo es que, puede ser codificado inmediatamente en un motor de base de datos relacional y nos permite tener un mejor control de versiones en caso que durante su diseño y desarrollo se requieran

cambiar algunas cosas, recordemos que la metodología RUP es iterativa e incremental.

Otra ventaja es que, las personas involucradas en el diseño del sistema de almacenamiento en la base de datos tenemos mayor experiencia en este modelo relacional para resolver incongruencias, duplicidad y garantizar a integridad de los datos.

Sin embargo, para poder realizar este cambio, vamos a aplicar ciertas reglas que hemos desarrollado durante el análisis de otros sistemas y que nos han llevado a diseñar bases de datos funcionales. Los pasos que llevamos a cabo fueron:

1. Toda entidad, es una tabla.
2. Definimos los atributos de cada entidad y los agregamos al modelo.
3. Definimos su cardinalidad.

También es tiempo que definamos la nomenclatura correcta para los objetos que vamos a crear, pues esto garantizará una fácil lectura durante la etapa de desarrollo y agilizará el proceso.

Las siguientes reglas tendrán que respetarse durante la etapa de diseño y la implementación de la base de datos, se han considerado todos los elementos que se piensa que se utilizaran, para que no haya duda durante la implementación y tengamos que volver a la etapa del diseño estructural:

1. Los nombres de todos los elementos deben ser descriptivos.
2. Las tablas, vistas y catálogos deberán ser nombradas en singular, describiendo al objeto que almacenan.
3. Los procedimientos almacenados y funciones deberán contener un verbo que describa su acción.
4. El nombre de todo elemento de la base de datos será precedido por las siglas “Spha” que identificará que éstos pertenecen propiamente al sistema SOPHIA.

Por ejemplo, la tabla que almacena los usuarios del sistema deberá llamarse: SphaUsuario.

5. Los catálogos serán nombrados anteponiendo a su nombre las siglas "CAT".

   Por ejemplo, el catálogo que almacene los perfiles de los usuarios del sistema deberá llamarse: SphaCATPerfil.
6. Las vistas serán nombradas anteponiendo a su nombre las siglas "VW" (Indicando que se trata de una "VieW".

   Por ejemplo, la vista que traiga de la base de datos a todos los Usuarios y su Perfil deberá llamarse "SphaVWUsuarioPerfil".
7. Los procedimientos almacenados serán nombrados anteponiendo las siglas "SP" (Indicando que se trata de un Stored Procedure).

   Por ejemplo, el procedimiento almacenado que almacena a un usuario en la base de datos deberá llamarse "SphaSPAlmacenaUsuario".
8. Las funciones serán nombrados anteponiendo las siglas "FC" (Indicando que se trata de una FunCtion).

   Por ejemplo, la función para traer el número total de usuarios de la base de datos deberá llamarse "SphaFCObtenerTotalUsuarios".
9. Los Disparadores de acciones serán nombrados anteponiendo las siglas "TG" (Indicando que se tratan de un TriGger).

   Por ejemplo, el disparador que alerte sobre el cambio del contenido de una noticia deberá llamarse "SphaTGNoticiaModificada".
10. Los campos deberán ser descriptivos y contener las 3 primeras letras del nombre de la tabla (no incluyendo las siglas "Spha" y en el caso de los catálogos, omitiendo también las siglas "CAT".

Por lo tanto, después de aplicar esas reglas al modelo E-R obtenido, tenemos como resultado el primer modelo relacional que deberá ser resuelto, de acuerdo a su cardinalidad, aplicando las siguientes reglas para los siguientes casos que se presenten:

1. Primero se ubica en cada entidad, un atributo que haga único a cada objeto creado dentro de esta estructura, en caso que no exista se revisa si la entidad debe existir, o se le asigna un numérico incremental, a este atributo se le denomina **llave primaria**.
2. Si la relación es uno a uno (1,1) entre 2 entidades, los atributos de la entidad A se colocan en la entidad B, y se toman como una sola entidad. El o los atributos que sean únicos en la nueva tabla será considerada una **llave primaria**.
3. Si la relación es uno a varios (1,n), se toma la **llave primaria** a la entidad cuya cardinalidad es n, y se coloca en la entidad cuya cardinalidad es 1 (a este atributo se le conoce como **llave foránea**).
4. Si la relación es varios a varios, se crea una tabla intermedia, en donde se colocan las llaves primarias de cada entidad y si esta relación de llaves se convierte en la llave primaría de esta nueva tabla y se le colocan los atributos necesarios que describan la relación.

La resolución de estos conflictos ofrecerá que el modelo sea integral y no exista duplicidad de la información almacenada, lo que se convierte en un mejor aprovechamiento del espacio y en la optimización de las búsquedas, pues los motores relacionales crean estructuras que permiten mejor manejo de la información cuando existen estas reglas.

Una vez que se tiene el modelo relacional y se comienzan a trabajar con la resolución de los conflictos que existen comienza a tener forma el sistema de almacenamiento. La primera versión del Sistema de Información de SOPHIA, queda definido de la siguiente forma:

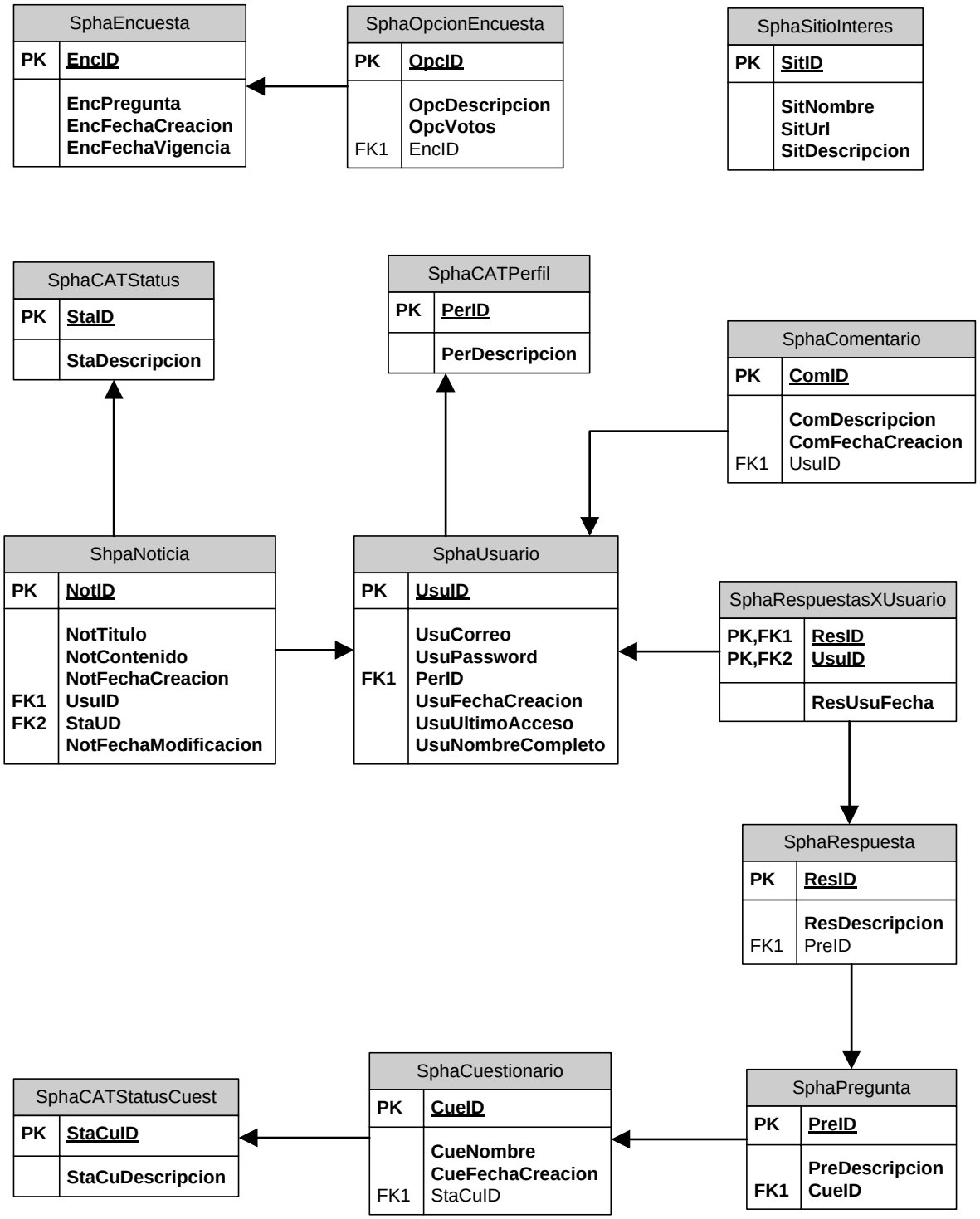

Figura 32. Modelo relacional de SOPHIA

El modelo relacional de SOPHIA debe pasar por un último proceso denominado Normalización de bases de datos.

El proceso de *normalización de bases de datos* consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. Esto tiene como objetivo:

- Evitar la redundancia de los datos. Tener redundancia implica mayor espacio de almacenamiento y la probabilidad de que una entidad posea 2 descripciones distintas, por ejemplo un mismo usuario con 2 perfiles.
- Evitar problemas de actualización de los datos en las tablas. Al actualizar una tabla que no se encuentra normalizada se pueden pasar por alto atributos que deben cambiarse dado que están dispersos en otras tablas.
- Proteger la integridad de los datos. Se refiere a que en todo momento, la información almacenada en una base de datos sea confiable y se pueda utilizar sin temor a errores de estructuras.

En el modelo relacional es frecuente llamar *tabla* a una relación, aunque para que una tabla sea considerada como una relación tiene que cumplir con algunas restricciones:

- Cada columna debe tener su nombre único.
- No puede haber 2 filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

La normalización de SOPHIA tendrá por objetivo encontrarse en la Forma Normal de Boyce – Codd.

En la teoría de bases de datos relacionales, las **formas normales** (**NF**) proporcionan los criterios para determinar el grado de vulnerabilidad de una tabla a inconsistencias y anomalías lógicas. Mientras sea más alta la forma normal aplicable a una tabla, es menos vulnerable a inconsistencias y anomalías. Cada tabla tiene una "forma normal más alta" (HNF): por definición, una tabla siempre satisface los requisitos de su HNF y de todas las formas normales más bajas que su

HNF; también por definición, una tabla no puede satisfacer los requisitos de ninguna forma normal más arriba que su HNF.

Las formas normales son aplicables a tablas individuales; decir que una base de datos entera está en la forma normal n es decir que todas sus tablas están en la forma normal n.

La normalización procede de una manera iterativa, es decir un diseño en **primera forma normal** primero se normaliza a **segunda formal normal**, entonces a **tercera forma normal**, etcétera.

Una tabla sensiblemente diseñada es probable que esté en 3NF en la primera tentativa; además, si está en **Tercera Formal Normal**, también es extremadamente probable que tenga una forma HNF de **Quinta Forma Normal**. Conseguir formas normales "más altas" (sobre **Tercera Formal Normal**) usualmente no requiere un gasto adicional de esfuerzo por parte del diseñador, porque las tablas usualmente no necesitan ninguna modificación para satisfacer los requisitos de estas formas normales más altas.

Edgar F. Codd originalmente definió las 3 primeras formas. Estas formas normales se han resumido como requiriendo que todos los atributos no-clave sean dependientes en "la clave, la clave completa, y nada excepto la clave". Las cuarta y quinta formas normales se ocupan específicamente de la representación de las relaciones muchos a muchos (n,n) y uno a muchos (1,n) entre los atributos.

Para alcanzar la forma normal de Boyce – Codd (o FNBC), se requiere que las tablas cumplan con las 3 primeras formas normales (1FN, 2FN y 3FN).

La Primera Forma Normal establece básicamente, que debemos asegurarnos que una tabla es representación fiel de una relación y está libre de grupos repetitivos. Sin embargo existen diferencias entre los autores sobre en qué momento una tabla se descalifica como 1FN, por lo que nosotros utilizaremos el criterio de Edgar F. Codd y establecemos que, nuestras tablas están en 1FN si:

1. Toda tabla tiene una llave candidata a ser primaría.

2. No están permitidos los grupos repetitivos.
3. Todas las columnas son regulares, por lo tanto, no aceptan valores nulos.
4. Todo campo debe ser atómico, lo que significa que no puede ser descompuesto en campos menores.

Con estas reglas, podemos garantizar que nuestra base de datos se encuentra en 1FN y podemos comenzar a aplicar las condiciones para llevarla 2FN.

La Segunda Forma Normal establece que, una tabla está en 2FN si primero está en 1FN y dada cualquier llave candidata y cualquier atributo que no constituya una llave candidata, el atributo no clave depende de toda a llave primaria en vez de solo una parte de ella.

Vamos a ejemplificar esta regla de la siguiente forma:

Nombre	Idioma	Área
Juan López García	**Inglés**	Sistemas
Raymundo Pérez López	**Inglés**	Sistemas
Raymundo Pérez López	**Francés**	Sistemas

La llave candidata es {Nombre, Idioma} por lo que el atributo restante área depende únicamente del Nombre, esto ocasiona que, en una actualización, Raymundo Pérez López pertenece 2 veces al área de Sistemas, para que está tabla cumpla con 2FN tiene que dividirse de la siguiente forma:

Nombre	Área
Juan López García	Sistemas
Raymundo Pérez López	Sistemas

Nombre	Idioma
Juan López García	Inglés
Raymundo Pérez López	Inglés
Raymundo Pérez López	Francés

La Tercera Forma Normal establece que, para que una tabla se encuentre en 3FN debe cubrir 2 condiciones: la tabla debe encontrarse en 2FN y ningún atributo no-primario de la tabla es dependiente transitivamente de una clave candidata.

Esto significa que, ningún atributo Z que depende de X debe estar en la tabla si existe una relación donde el atributo Z depende de un atributo Y y a su vez, el atributo Y depende de X.

Para manejarlo de una forma más clara, podemos ejemplificarlo de la siguiente forma:

Año Escolar	Materia	Alumno	Fecha de Nacimiento
2010	Matemáticas	Juan López García	01/01/2000
2010	Matemáticas	Raymundo Pérez López	04/06/2000

Como podemos ver, la fecha de nacimiento depende del alumno, y el alumno depende del año y la materia donde se encuentra inscrito, por lo tanto, esta tabla no cumple con la 3FN. Para que cumpla, debe estructurarse de la siguiente forma:

Año Escolar	Materia	Alumno
2010	Matemáticas	Juan López García
2010	Matemáticas	Raymundo Pérez López

Alumno	Fecha de Nacimiento
Juan López García	01/01/2000
Raymundo Pérez López	04/06/2000

La Forma Normal de Boyce- Codd nos indica que, para que una tabla esté en FNBC, si consideramos las siguientes condiciones: Si no existen llaves candidatas compuestas, está en FNBC y Si existen varias claves candidatas compuestas y éstas no tienen un elemento en común, no está en FNBC.

Es muy difícil que una tabla en 3FN no satisfaga los requerimientos de la FNBC, sin embargo, en SOPHIA existe una relación que puede caer en este caso, la idea de que cumpla con esta forma normal es para garantizar que las tablas no sean vulnerables a inconsistencias lógicas.

En el caso de SOPHIA; para la tabla de Respuestas por Usuario (una tabla que salió de eliminar un conflicto de cardinalidades), tiene una llave candidata compuesta que permite, si consideramos esta llave, que un Usuario capture más de una respuesta, aunque éstas pertenezcan a una misma respuesta.

La solución para que cumpliera con la última forma normal fue, traer la llave de la tabla de Preguntas y lograr que con la llave del Usuario se convirtiera en una llave candidata y que la llave Respuesta sea un atributo no candidato, con esto aseguramos que cada pregunta resuelta por cada usuario, solo pueda existir una vez, y ésta tenga una única respuesta, garantizando la integridad de los datos y que no existan inconsistencias lógicas.

Para lograr que la base de datos de SOPHIA cumpliera con las formas normales planteadas, se generaron siete iteraciones, lo que resultó en siete versiones de la base de datos, siendo la última la que se implementó en el motor de base de datos, sin embargo se respetó su nomenclatura para que existiera un antecedente del trabajo realizado que cumpliera con su proceso de normalización.

La base de datos resultante fue:

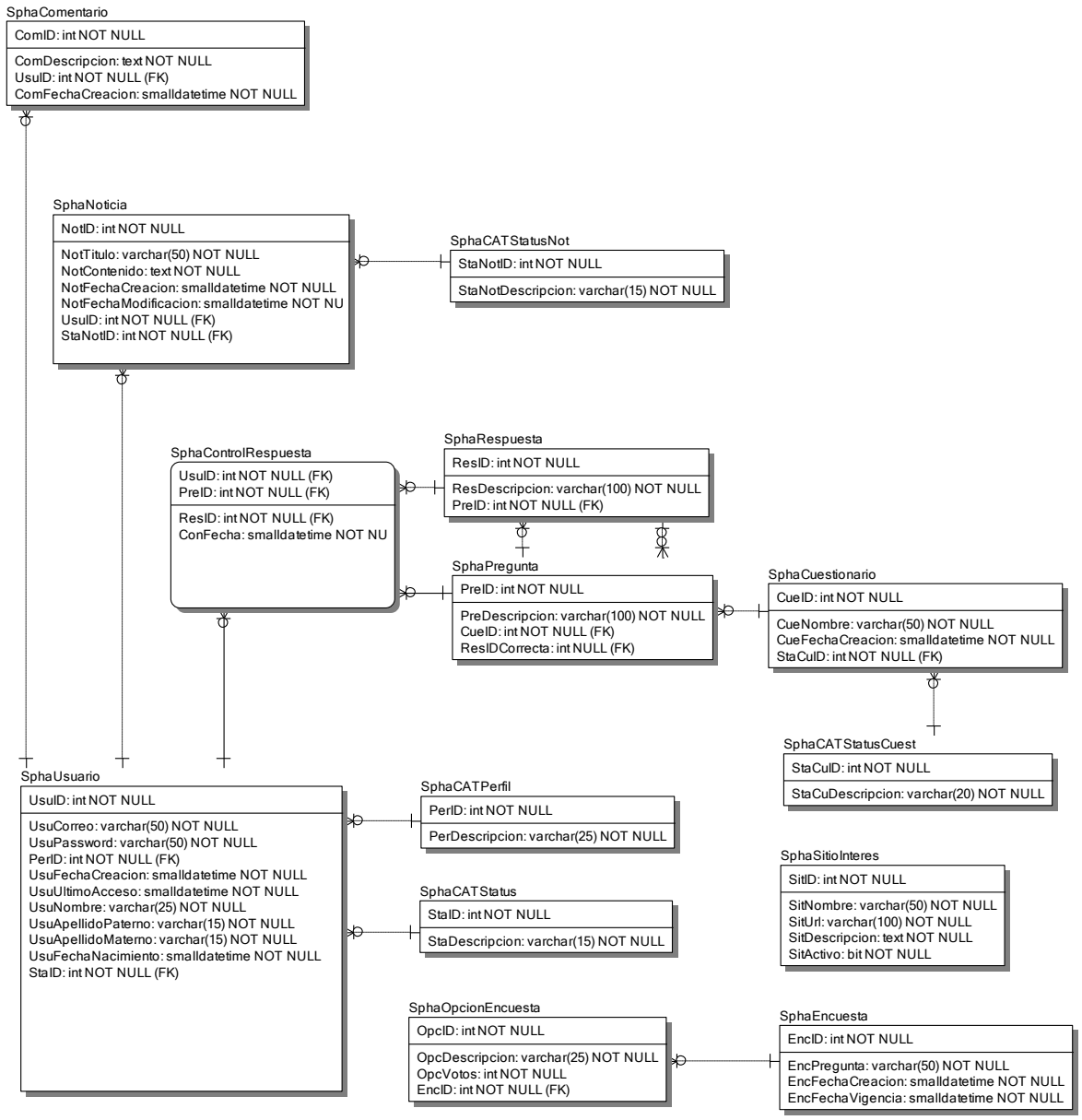

Figura 33. Versión final del Modelo Lógico Relacional de SOPHIA

Matriz de Acceso

Este modelo lo traemos del área de seguridad de la información porque nos permitirá ver, de forma sencilla, el alcance que debe tener cada usuario en el sistema de información de SOPHIA y a que está o no limitado.

Dentro del modelo por capas, la matriz de acceso será la documentación necesaria para proteger nuestros datos de visitas no autorizaras. Cabe mencionar que la matriz de acceso es a nivel de aplicación y de este diagrama no depende la seguridad física del equipo, la administración de bases de datos, la administración del sistema de entrada y salida del servidor y los accesos lógicos a los servidores.

La seguridad del equipo dependerá del administrador del servidor y del aseguramiento que haga del mismo, lo cual podrá analizarse en la etapa de la implementación y no es parte fundamental durante el desarrollo, nuestro objetivo como diseñadores en este momento es garantizar que desde la aplicación, la información posea un control de acceso.

Una matriz de acceso es una representación conceptual del nivel de acceso de cada usuario y se conforma de una matriz de 2 dimensiones, donde, en una dimensión se localizan los perfiles del sistema y en la otra dimensión los puntos de gestión de información a los cuales se tiene acceso.

Para el sistema de información de SOPHIA vamos a considerar los siguientes elementos para la dimensión de perfiles:

- Administrador del Sitio
- Investigador
- Profesor
- Alumno
- Apoyo de Servicio Social
- Público

Para la dimensión de la gestión de información se consideraran los siguientes puntos:

- Lectura de Artículos.
- Visualizar Material Multimedia.
- Publicación de Artículos.

- Publicación de Comentarios.
- Gestión de Material Multimedia.
- Autorización de contenido.
- Gestión de Comentarios.
- Agregar y modificar artículos.
- Gestión de Cuestionarios.
- Gestión de Comentarios.
- Gestión de Encuestas.
- Resultados de Encuestas.
- Visor RSS.
- Visor de Sitios de Interés.
- Gestión de Usuarios.

Por lo tanto, la matriz de acceso para el sistema quedaría de la siguiente forma:

Acciones/Perfil	Lectura de Artículos	Visualizar Material Multimedia	Publicación de Artículos	Publicación de Comentarios	Gestión de Material Multimedia	Autorización de contenido
Administrador del Sitio	X	X	X	X	X	X
Investigador	X	X	X	X		
Alumno	X	X		X		
Profesor	X	X	X	X		
Apoyo de Servicio Social	X	X		X	X	
Público	X	X				

Tabla 1. Matriz de Acceso 1

Acciones/Perfil	Gestión de Comentarios	Gestión de Encuestas	Resultados de Encuestas	Visor RSS	Visor de Sitios de Interés	Gestión de Usuarios
Administrador del Sitio	X	X	X	X	X	X
Investigador				X	X	
Alumno				X	X	
Profesor		X	X	X	X	
Apoyo de Servicio Social	X	X	X	X	X	X
Público			X	X	X	

Tabla 2. Matriz de Acceso 2

Acciones/Perfil	Contestar Encuestas	Ver Resultado de Encuestas	Gestión de Comentarios	Agregar y modificar artículos	Gestión de Cuestionarios
Administrador del Sitio	X	X	X	X	X
Investigador	X	X		X	X
Alumno	X	X			
Profesor	X	X		X	X
Apoyo de Servicio Social	X	X			X
Público	X	X			

Tabla 3. Matriz de Acceso 3

Esta matriz nos permitirá a lo largo del desarrollo colocar los candados necesarios para evitar que un usuario no autorizado tenga acceso a información privada para el CEAQ, también nos permitirá visualizar una forma de presentar la información respetando las reglas planteadas desde esta etapa de diseño.

Mapa de comportamiento de hardware

En este modelo podremos darnos cuenta cómo es que la información viajará dentro del hardware y su interacción física con otros elementos.

El alcance de este modelo está en definir desde un inicio la arquitectura física y las consideraciones que debemos tomar en cuenta durante el desarrollo de la aplicación para lograr que la información fluya correctamente desde los actores hasta los sistemas de información.

Los elementos que conforman este mapa de comportamiento son los siguientes:

- Un servidor de base de datos (con el software SQL Server 2008)
- Un servidor de archivos (con el software Windows Server 2008)
- Un servidor de aplicación (con el Software Internet Information Services)
- La red de la Facultad de Medicina
- La red de la Universidad Nacional Autónoma de México
- Internet
- Equipos de conexión a internet (puntos de acceso inalámbricos, módems, líneas telefónicas o líneas celular)
- Equipo de Escritorio, Equipos Portátiles o Equipos Móviles (con cualquier navegador de páginas web, optimizado para Internet Explorer 8 y Mozilla Firefox y reproductor Flash).

Todo el hardware mencionado se relaciona como se muestra en la figura 34:

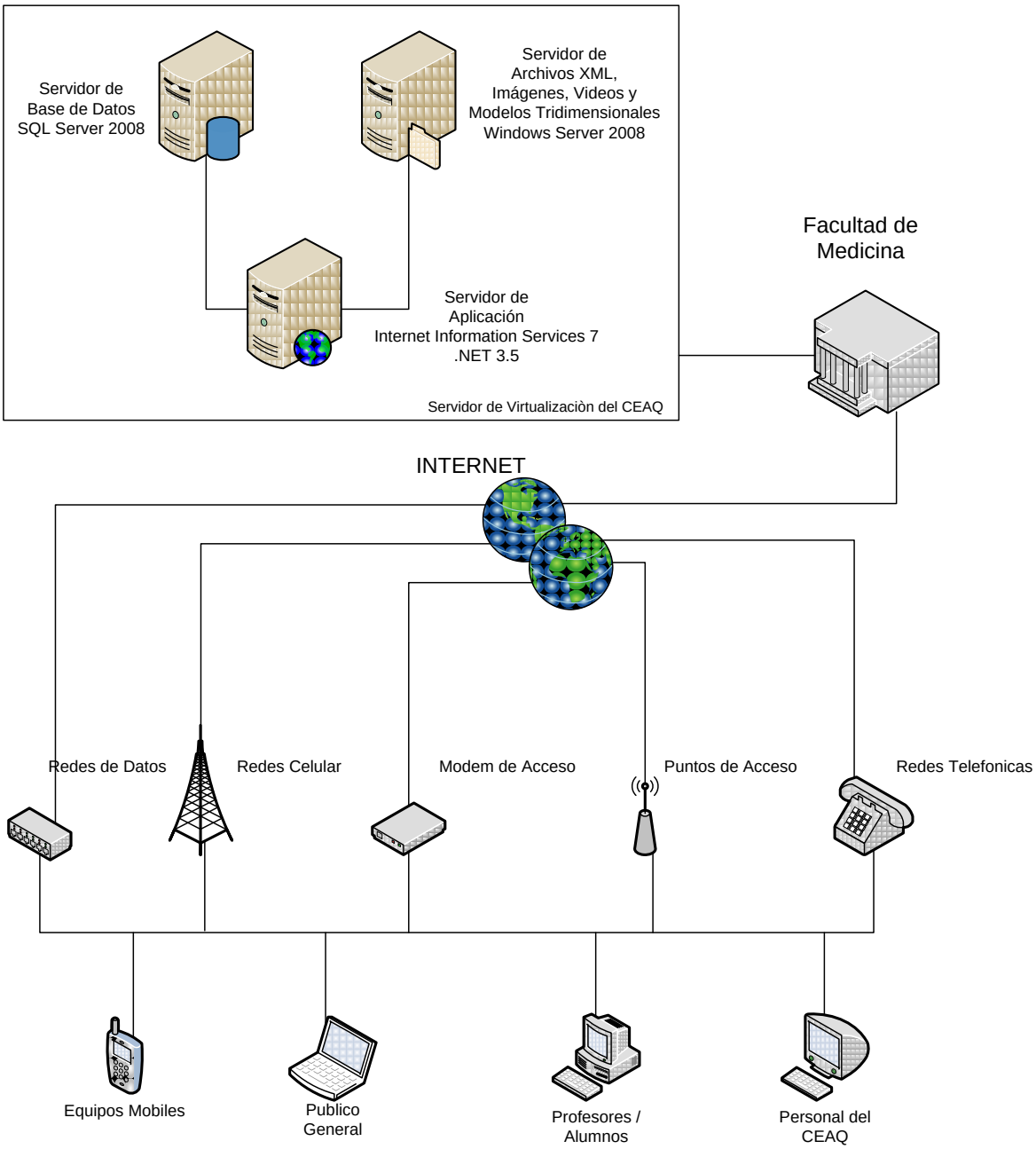

Figura 34. Modelo de Comportamiento de Hardware

De este diagrama podemos ver que el canal de distribución está enfocado a todo equipo que se conecte a internet, actualmente la evolución de la navegación por sitios web son los dispositivos móviles, ya existen equipos que son compatibles con sitios como la aplicación de SOPHIA, sin embargo, al no haberse especificado desde un inicio, solo se consideraron equipos móviles muy potentes.

Modelo de estructura archivos XML

Existen cuatro estructuras de datos que van a definir 2 módulos de SOPHIA, el control de acceso a usuarios y las galerías multimedia (Imágenes, Video y Modelos Tridimensionales); estas estructuras fueron creadas en archivos XML para dar un mantenimiento rápido y sin afectar la operación del sistema.

Control de acceso

Para la estructura de este archivo, vamos a utilizar el valor numérico del catálogo de perfiles, considerando el valor de 0 como el usuario público.

La estructura será la siguiente:

```
<perfiles>
  <menu perfil="Identificador del perfil">
    <opcion name="Nombre de la etiqueta en el Menu Principal 1">
      <subopcion1 name="Etiqueta del submenu 1" page="Página Web 1" />
      <subopcion1 name="Etiqueta del submenu 2" page="Página Web 2" />
      …
    </opcion>
    …
   </menu>
   …
</perfiles>
```

A continuación vamos a ver un ejemplo del menú público, para poder entender cómo funciona esta estructura de archivo:

```
<perfiles>
  <menu perfil="0">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
```

```
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
</perfiles>
```

Este menú se debe generar de la siguiente forma:

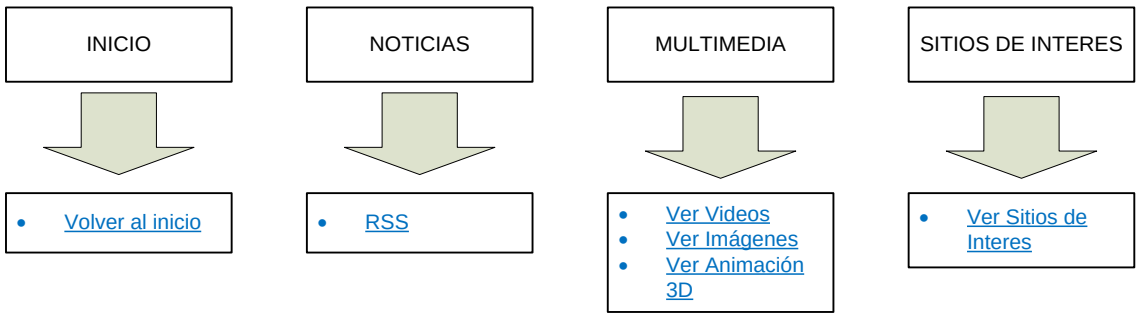

Figura 35. Diagrama de Menú Visual

Multimedia

Para los elementos multimedia se va a trabajar con una misma estructura de datos y cada tipo tendrá un archivo específico, esto nos permitirá un mejor orden al definir los elementos multimedia y un control de archivos.

Un archivo multimedia consta de 6 elementos:

1. Una categoría. Es un agrupador de elementos del mismo tipo, por ejemplo: Sistema Óseo, Sistema Nervioso, Sistema Muscular, etc.
2. Un subdirectorio donde irán los archivos de esta categoría
3. Un número identificador. Es un campo numérico para poder darle un orden a cada galería multimedia.
4. Un Título. Es un campo de texto que nos permite identificar rápidamente al elemento.

5. Una Descripción. Es un campo de texto con más detalle sobre el elemento multimedia que se está evaluando.
6. Un archivo. Es un campo de texto donde se coloca la ruta donde se localiza el archivo multimedia.

Por lo tanto, una estructura que define estos elementos es la siguiente:

```
<Collection>
  <Category name="Nombre de la Cateogia" subdirectory="Subdirectorio
donde iran estos archivos">
    <item id="Identificador Númerico" name="Titulo de elemento"
file="Archivo Multimedia">Descripción detallada del archivo</item>
    …
  </Category>
  …
</Collection>
```

Para entender el funcionamiento de esta estructura vamos a ver ejemplo sobre la utilización de la misma:

```
<?xml version="1.0" encoding="utf-8" ?>
<Collection>
  <Category name="Sistema Oseo" subdirectory="oseo">
    <item id="1" name="Craneo" file="videoCrane.flv" >Este video muestra
la estructura de una enfermedad que atacá el craneo.</item>
    <item id="2" name="Rodilla" file="videoRodilla_20100106.flv" >Este
video muestra el desgaste natural de la rodilla.</item>
    <item id="3" name="Femur Roto por Accidente"
file="femur_20100202.flv" >Este video muestra la anomalia de un femur
despues de un accidente de bicicleta.</item>
  </Category>
  <Category name="Sistema Neurologico" subdirectory="neurologico">
    <item id="1" name="Neurona con Citicercosis" file="citiNeurona.flv"
>Este video muestra la estructura de una enfermedad que atacá el
cerebro.</item>
    <item id="2" name="Muerte Cerebral" file="muerteCerebral.flv" >Este
video muestra la anomalia de un cerebro despues de un accidente de auto
por alcohol.</item>
  </Category>
<Collection>
```

Con estas definiciones hemos terminado el modelado de todo el sistema de SOPHIA, con lo cual tenemos la documentación necesaria para entrar a la etapa del desarrollo, la codificación de todos estos lenguajes y reglas en varios tipos de lenguajes de programación.

Elaboración de modelos tridimensionales para SOPHIA

Los modelos tridimensionales que forman la colección del CEAQ no fueron pensados para su difusión por Internet, por lo cual es necesario crear unos que se adapten a las necesidades y limitantes que nos impone el compartir material de este tipo por la red.

El proceso original para la creación de modelos en el centro parte de la generación de la geometría a partir de imágenes tomográficas utilizando el software de visualización y análisis Amira, éste nos entrega un modelo muy cercano a su contraparte real, pero con un conteo de polígonos muy alto. Para la creación de material para la web se tendrá que seguir un nuevo proceso el cual requerirá de una estrecha colaboración entre el modelador, quien creará el modelo tridimensional y el experto del centro, quien lo asesorará en las partes técnicas para que el modelo muestre, en lo posible, lo que se requiere. El proceso se describe a continuación:

1. El primer paso es crear la geometría base, esto se realiza en el programa Maya, se toman como referencia ya sean imágenes o vistas del modelo de alta resolución original. Se debe procurar que nuestra geometría base cuente con el menor número de polígonos necesario para dar detalle a nuestro modelo. el usar modelos con un gran número de polígonos hará que tarde más tiempo en cargarse y dificulta su manipulación.
2. El siguiente paso es generar los mapas UV del modelo para la creación de texturas, este proceso también lo realizamos dentro de Maya. Una vez que tenemos nuestra geometría y su mapa UV exportamos el

modelo a formato OBJ para continuar con el proceso de creación de modelos.

3. El modelo en formato OBJ lo pasamos a Zbrush, utilizamos este formato ya que es la única forma de importar modelos hechos en otras aplicaciones a Zbrush. En este programa agregamos detalles a nuestro modelo, como porocidades, relieves, abolladuras. Al final obtenemos un modelo de alta resolución detallado y cercano al objeto original que se quiere representar.
4. Dentro de Zbrush generamos el mapa de normales que dará la apariencia de un modelo detallado a nuestra geometría base de bajo número de polígonos. La razón de usar Zbrush es el obtener este mapa de normales para nuestra geometría base.
5. Ahora el siguiente paso es texturizar nuestro modelo. Este proceso también lo podemos realizar en Zbrush, pero es recomendable que nuestra textura pase por Photoshop para ajustar detalles y agregar más efectos, lo cual dará más realismo a nuestro modelo. Otra manera es crear nuestra textura totalmente en Photoshop apoyándonos del mapa UV.
6. Una vez que tenemos nuestra textura y mapa de normales regresamos a Maya para integrar todo a nuestra geometría base, que lucirá con mucho más detalle.
7. Dentro de Maya realizamos un proceso de Baking texture, esto se hace con la finalidad de obtener una sola textura que reuna los detalles que agrega el mapa de normales a nuestro modelo, así como efectos de iluminación.
8. El último paso es exportar nuestro modelo con su nueva textura al formato COLLADA que podrá ser cargado por el visro de modelos tridimensionales de Sophia.

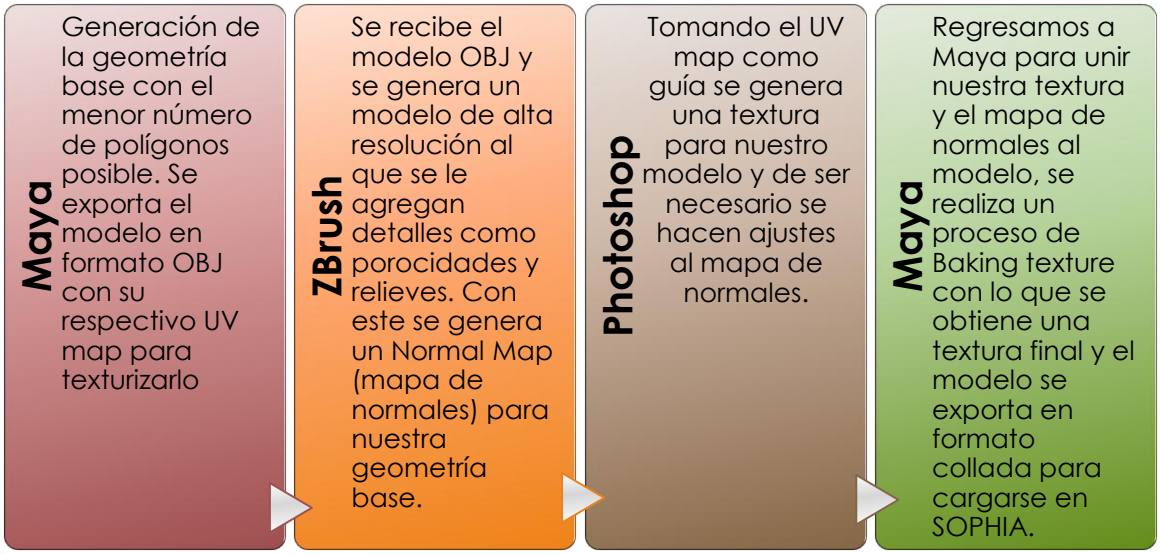

Figura 36. Proceso de creación de modelos tridimensionales

Al final de este proceso obtenemos un modelo que se puede cargar y visualizar sin problemas a través de Internet.

Conceptos para la creación de modelos tridimensionales

Formato OBJ

La estructura del archivo OBJ es un formato tridimensional creado por Wavefront para su producto Advanced Visualizer.

La estructura OBJ es un formato de datos simple que representa una geometría tridimensional, el archivo contiene la posición de cada vértice, la posición de cada coordenada de textura (UV), las normales, y las caras que componen a cada polígono compuesto por una lista de vértices y coordenadas de textura.

Estos archivos están basados en texto y soportan geometrías de formas libres y poligonales. Un archivo .mtl complementario describe los materiales definidos en el archivo .obj.

El formato básico del archivo OBJ es el siguiente:

```
# Un archivo OBJ contiene varias definiciones
# Lista de vértices, en coordenadas x, y, z.
v 0.123 0.234 0.345
v 0.124 0.541 0.750
# Coordenadas de textura
vt 0.500 0.850
vt 0.451 0.869
# Normales en la forma (x, y, z)
vn 0.707 0.000 0.707
# Las caras son definidas como lista de vértices.
f v1 v2 v3 v4
#cuando se texturiza el modelo se coloca define de la siguiente forma
f v1/vt1 v2/vt2 v3/vt3
```

Mapa Normal

Es la aplicación de una técnica que permite dar una iluminación y relieve mucho más detallado a la superficie de un objeto tridimensional sin utilizar más polígonos.

Un mapa de normales suele ser una imagen RGB que corresponde a las coordenadas cartesianas (X, Y, Z) de una superficie a partir de una versión más detallada del objeto, la imagen guarda la información sobre la dirección de las normales de cada cara de un objeto.

El objetivo de utilizar mapas de normales es reducir el número de polígonos (y tiempo) que se necesitan para modelar un objeto complejo. Normalmente se utilizan en aplicaciones que se ejecuten en tiempo real (juegos generalmente) pues disminuyen la carga del procesador. Sin embargo, pueden ser bastante útiles, en el caso de escenas estáticas o animaciones.

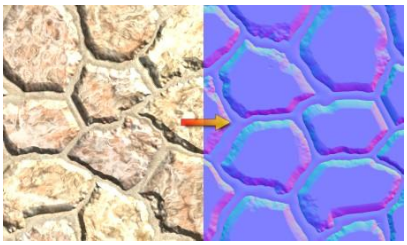

Figura 37. Creación de mapas normales

UV map

Consiste en representar un modelo 3D en una imagen 2D

Las coordenadas cartesianas (x, y, z) que representan al modelo tridimensional original en el espacio de modelo son transformadas a un espacio de 2 dimensiones representado por las letras U, V.

Esto nos permite pintar con color el modelo tridimensional con una imagen. Esta imagen recibe el nombre de mapa de textura UV, pero no es más que una imagen ordinaria. El proceso de mapeo UV implica la asignación de píxeles de la imagen en la malla que forma el modelo tridimensional.

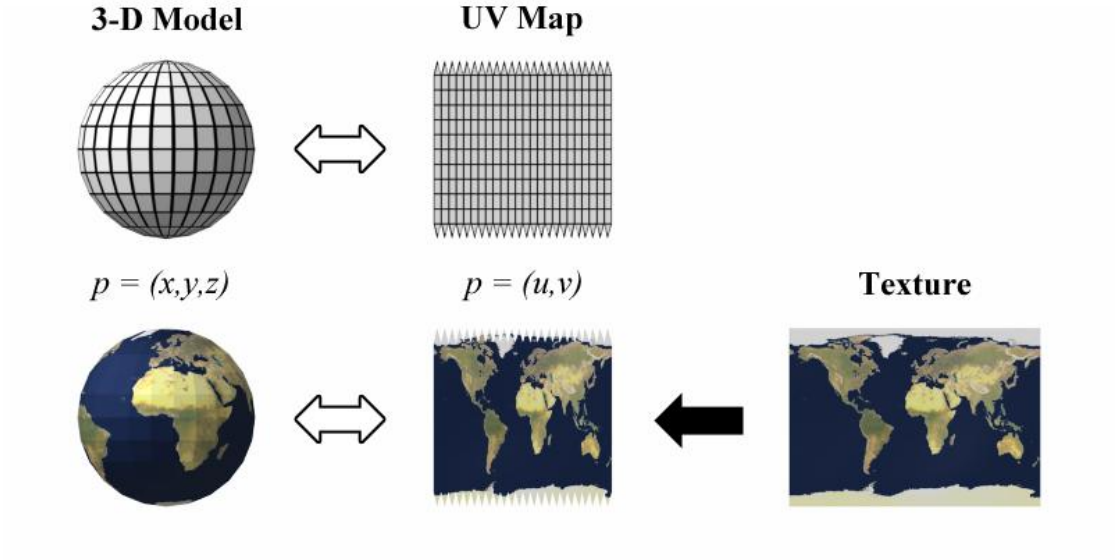

Figura 38. Creación de Textura para modelos tridimensionales

Baking texture

Es un método que permite crear mapas de texturas en base a la apariencia de un objeto en la escena con respecto a la iluminación, proyección de sombras, ajuste de opacidad, materiales y cualquier otro aspecto que afecte la superficie del objeto. Las texturas resultantes pueden ser utilizadas posteriormente para texturizar los mismos objetos para simulaciones en tiempo real (como videojuegos o visualización de arquitectura).

Capítulo III. Implementación

Introducción

Actualmente el proceso de reingeniería ha pasado por su etapa de planteamiento y diseño de una solución informática para proporcionar un servicio más óptimo al CEAQ, esto ha consumido la mayor parte de nuestro tiempo, pues se requiere una estructura firme para poder solidificar un sistema que cumpla con las expectativas del centro.

En este momento conocemos como funciona el negocio, las reglas que nos definen los límites del sistema y hemos entendido que un proceso de reingeniería no iba a impactar únicamente la página web, sino que también va a impactar los procesos internos del CEAQ, sin embargo es un tema que guardaremos para las conclusiones, debido a que de toda esta investigación hallamos vértices muy interesantes de la solución.

Una vez que entendimos el negocio y logramos plantear modelos que gráficamente nos permiten visualizar aquello que entendimos pudimos pasar a la etapa de diseño.

La etapa de diseño fue la que consumió la mayor parte de nuestro tiempo, pues fue el lapso durante el cual transformamos las reglas de negocio en diagramas funcionales para que pudieran transformarse en un sistema a través de una serie de lenguajes de programación.

El hecho de haber tenido una etapa de planeación nos permite llegar a esta etapa con un modelo a seguir, el cual nos ayudará a concretar nuestro objetivo de manera más ágil.

Sin embargo, tenemos que considerar 3 cosas antes de proseguir:

- Esta etapa es donde se aplicaran las buenas prácticas de la metodología XP.

- La metodología RUP nos indica que el proceso debe ser iterativo incremental, por lo que, podemos regresar a la etapa de diseño si esto es necesario para mejorar la implementación.
- Debemos desarrollar todos los objetos del sistema considerando las reglas que nos planteamos en el paradigma de programación por capas.

Una vez que tenemos claros estos 3 puntos, es momento de entender el proceso que seguiremos para desarrollar el sistema.

Vamos a trabajar el sistema por fases tomando como modelo de partida el orden del modelo de capas, iniciaremos por el núcleo e iremos construyendo los objetos hasta llegar al nivel superior, por lo que el proceso será el siguiente:

1. Implementación del Modelo Relacional en el motor de la base de datos.
2. Creación de estructuras de directorios para el sitio
3. Creación de Procedimientos Almacenados y Clases de Acceso.
4. Creación de Control de Acceso.
5. Creación de Clases de Entidades.
6. Creación de Clases de apoyo.
7. Creación de páginas web y Clases de Funcionalidad.
8. Creación de elementos para reproductor Flash.
9. Creación de Estilos y programación en el Cliente.

Una de las buenas prácticas de la metodología XP nos indica que, debemos desarrollar en conjunto la aplicación para disminuir así el número de errores en el código, sin embargo, explotando las capacidades de desarrollo de los involucrados decidimos desarrollar en conjunto, pero iniciando por los 2 extremos del proceso, esto permite compartir lo que se está realizando en cada momento y avanzar en las áreas donde cada uno tenemos conocimientos más sólidos.

Sin embargo, para cumplir con esta buena práctica, la compilación y solución de errores se hace en conjunto buscando no solo encontrar la solución, sino optimizar los recursos con los que se cuenta y desarrollar una aplicación desde 2 perspectivas diferentes pero con un objetivo en común.

Actualmente ya conocemos las tecnologías que vamos a utilizar, sin embargo, es necesario hacer una selección más específica sobre los lenguajes que propiamente vamos a utilizar y por qué lo vamos a utilizar.

Selección de Lenguajes y Software

En el capítulo 2 definimos la plataforma tecnológica que vamos a utilizar y seleccionamos algunos de los lenguajes en los que vamos a desarrollar, sin embargo en este momento vamos a ampliar la selección a definirlos por cada capa y el resultado que esperamos de cada uno.

Este análisis lo iniciaremos con la zona cero de nuestro modelo de capas, que es donde viven los sistemas de almacenamiento.

En la base de datos utilizaremos como ya lo habíamos dicho, el motor de base de datos relacional **Microsoft SQL Server 2008**, dado que, el sistema de información de SOPHIA debe evolucionar para poder explotar sus datos y tener a la mano elementos para la toma de decisiones, este software tiene una amplia gama de herramientas que permiten estos análisis detallados de información. Esto no quiere decir que otros motores no cuenten con estas aplicaciones, pero la experiencia que tenemos en desarrollo nos hace inclinarnos por aquella solución que conozcamos mejor y podamos dar una mejor asesoría.

Debido a la selección de este motor de base de datos, tenemos que elegir como lenguaje de desarrollo el **Transact-SQL (T-SQL)**, dado que es el lenguaje con el cual trabaja este motor, en este lenguaje vamos a desarrollar los procedimientos almacenados y funciones que utilicemos para el sistema.

Para los archivos de estructura, utilizamos, como ya lo vimos en el capítulo 2, estructuras de archivos **XML** debido a que son un estándar en el desarrollo y nos permiten crear estructuras propias para la representación ordenada de información y explotarla con la mayoría de lenguajes actuales. En este apartado es importante aclarar que el uso de estructuras **XML** en inglés se hace con la finalidad identificar las estructuras fácilmente para el desarrollador y que durante el proceso de desarrollo no limite sus posibilidades de variables a un solo idioma,

durante la etapa del desarrollo se verá un poco más de esta selección de idiomas que permite hacer sistemas más claros en la parte del desarrollo interno y que al comunicarse con otros módulos se hace en el idioma de los programadores.

En la etapa de la programación de clases, como ya lo habíamos mencionado, utilizaremos el Framework de .NET, seleccionando el lenguajes de programación **C# (C Sharp**), su sintaxis deriva de los lenguajes C/C++ y es muy similar al lenguaje Java. Esto nos permite un mantenimiento más sencillo y que otros desarrolladores puedan entenderlo fácilmente, gracias a su similitud con lenguajes comunes.

La ventaja de elegir a C# como lenguaje de desarrollo para las clases y objetos radica en que, es un estándar aprobado por ECMA [37](ECMA-334) e ISO [38](ISO/IEC 23270:2003), lo cual nos garantiza un menor costo en mantenimiento por el cambio de sintaxis o por cambio de versión.

Otra ventaja está en que, actualmente los esfuerzos de Microsoft están en este lenguaje más que en cualquier otro de su Framework, lo cual nos da mejores garantías de pasar periodos más largos antes de tener que dar mantenimiento de la aplicación.

La utilización de este lenguaje lo haremos a través de la interfaz de desarrollo llamada Visual Studio 2008, el cual nos permitirá desarrollar, compilar, probar e implementar el desarrollo de SOPHIA.

Este lenguaje cubrirá al menos cuatro capas de nuestro paradigma y es donde principalmente se llevaran a cabo las reglas de acceso a las distintas capas de elementos.

[37] Ecma International es una organización internacional basada en membrecías de estándares para la comunicación y la información.

[38] La **Organización Internacional para la Estandarización** (acrónimo del inglés *International Organization for Standardization*), es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.

Todo el código generado en C# será pre compilado y se crearan librerías las cuales se instalaran en el sitio web y podrán ser utilizadas a través de un espacio de nombres, todos los niveles de modelo de capa existirán dentro de una librería principal, sin embargo, no se podrán tener acceso a las clases de los modelos más interiores, pues estarán protegidas con los identificadores de acceso propios de .NET, estos son:

- public: acceso no restringido, puede ser accedido desde cualquier aplicación que cargue la librería que lo contiene.
- protected: acceso limitado a la clase contenedora o a los tipos derivados de esta clase.
- Internal: acceso limitado al ensamblado actual, o bien, acceso a las clases de la misma librería.
- protected internal: acceso limitado al ensamblado actual o los tipos derivados de la clase contenedora.
- private: acceso limitado al tipo contenedor o los procesos dentro de la misma clase.

La ventaja de la utilización de C# es los beneficios de la programación orientada a objetos, pero con algunas mejoras respecto a otros lenguajes similares.

Podemos utilizar 3 características importantes en este lenguaje de programación:

- Polimorfismo: Se refiere a que comportamientos diferentes, asociados a objetos distintos pueden utilizar el mismo nombre, en el caso de C# lo vemos en la sobrecarga de métodos y constructores que permiten aplicar más fácilmente ciertos procesos del negocio.
- Herencia: Las clases relacionadas entre sí, pueden heredar propiedades y métodos.
- Encapsulamiento. Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.

- Principio de ocultación. Cada objeto está aislado del exterior y provee interfaces de comunicación, estas pueden ser controladas a través de procesos previos de validación, en C# se utilizan unas estructuras denominadas propiedades.

La ventaja de este lenguaje de programación, utilizando la IDE Visual Studio es la utilización de una herramienta muy poderosa llamada *IntelliSense*, que es la aplicación para autocompletar líneas de código, además de completar el símbolo de los nombres que el programador está escribiendo, IntelliSense analiza previamente las clases que se están utilizando y proporciona los nombres de todos los elementos con iconos de imaginación para hacer el desarrollo muy visual.

La ventaja de la utilización de esta herramienta, está en que, si las clases son documentadas con etiquetas XML, el sistema IntelliSense proporciona al usuario la descripción específica del objeto que se pretende utilizar. SOPHIA por supuesto, estará documentada en sus librerías de las capas más internas para proporcionar en la etapa de mantenimiento descripciones precisas de cada elemento desarrollado.

Los lenguajes mencionados anteriormente se refieren a la aplicación que se ejecutara del lado del servidor, ahora vamos a definir los lenguajes y tecnologías que van a ejecutarse del lado del cliente.

El lenguaje para desarrollar las aplicaciones para el reproductor Flash será el que utiliza de forma nativa, utilizaremos el lenguaje **Action Script** y nos apoyaremos en algunas librerías que ya existen como Papervision, todo esto utilizando Adobe Flash CS3 como interfaz de desarrollo.

La ventaja de la utilización del reproductor flash es que actualmente existen implementaciones para la mayoría de los navegadores y se pueden diseñar mejores estructuras visuales gracias a toda la suite de Adobe.

Para la parte del sitio web, vamos a utilizar 3 lenguajes que son un estándar en el desarrollo de sitios para internet, estos son:

- HTML
- CSS
- Javascript utilizando el framework jQuery.

La utilización de estos lenguajes para la parte de la presentación de los datos es porque son un estándar y pueden ser ejecutados por cualquier navegador web del mercado, lo que nos permite llegar a más usuarios, que es el objetivo principal de este proceso de reingeniería.

En la etapa de desarrollo se crearan estructuras que permitan un fácil mantenimiento a la aplicación en un futuro, una rápida expansión a nuevos requerimientos y la implementación de nuevas tecnologías e análisis de datos.

A continuación presentamos un diagrama donde se puede ver, el tipo de servicio que se proporciona, y el lenguaje que se utiliza:

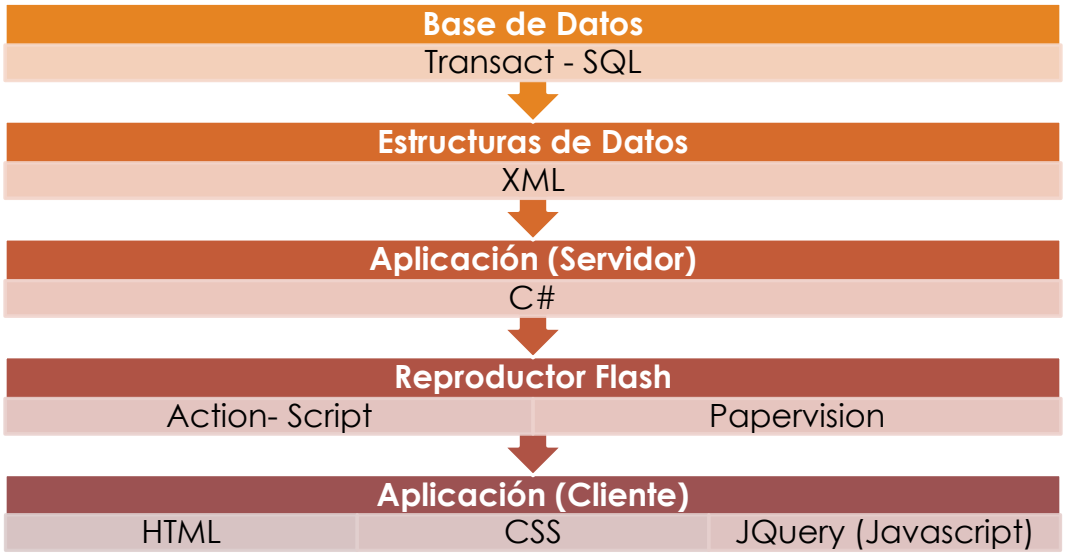

Figura 39. Lenguajes de programación por Capa

Programación y Documentación

Una vez que hemos elegido las tecnologías y los lenguajes necesarios, vamos a iniciar la etapa de desarrollo, en primer lugar tenemos que generar un ambiente

para concebir la creación de los objetos y también servirá para realizar pruebas, dicho ambiente debe ser controlado y tener un repositorio del código.

La opción fue la utilización de un equipo virtual, con las tecnologías que se usaran en el ambiente productivo.

El concepto de virtualización consiste en la creación de una capa abstracta entre el sistema operativo y el hardware de la maquina física, logrando simular el hardware suficiente para la operación de un nuevo sistema operativo.

La virtualización tiene las siguientes ventajas, desde la solución que estamos proporcionando:

- Rápida incorporación de nuevos recursos para los servidores virtualizados. Podemos crear más de un servidor para las pruebas, de forma rápida.
- Reducción de los costes de espacio y consumo eléctrico. No requerimos tener varios servidores para el desarrollo de la aplicación lo que reduce los costos de equipos adicionales.
- Administración global centralizada y simplificada.
- Mejora en los procesos de clonación y copia de sistemas: Mayor facilidad para la creación de entornos de test que permiten poner en marcha nuevas aplicaciones o bien, realizar respaldos constantemente sobre las distintas etapas del desarrollo.
- Aislamiento: un fallo general de sistema de una máquina virtual no afecta al resto de máquinas virtuales, lo que permite no parar en ningún momento el desarrollo y nos permite encontrar errores en paralelo al desarrollo.

Para la virtualización se utilizó una herramienta llamada Virtual PC 2007 de Microsoft. El equipo virtual tenía las siguientes características:

- 512 MB en RAM
- 16 GB en Disco Duro con expansión dinámica.
- Tarjeta de Red 10/100 (con IP fija)

Una vez creado el servidor virtual se le instaló en primer lugar Windows Server 2008. Concluida la instalación se configuró una IP fija y se inició la instalación del motor de la base de datos.

El motor de la base de datos ejecutó pruebas sobre el sistema recién instalado y realizó la instalación exitosamente, sin embargo, en ese momento el servidor únicamente funcionaba de manera local.

Se necesitó realizar la apertura del puerto por el cual, el motor proporciona su servicio al CEAQ y a la aplicación de SOPHIA. En el servidor se permitió el acceso al puerto 1433 y se logró tener acceso desde aplicaciones externas al motor.

Una vez que tuvimos acceso al motor, pudimos iniciar con la instalación de la herramientas necesarias para aplicación, para ello fue instalar en el servidor la herramienta del Internet Information Services y se activaron los elementos necesarios para la ejecución de páginas ASP.NET.

Una vez concluida esta etapa, se instaló el Framework 3.5 y una herramienta llamada Microsoft Chart, que proporciona al servidor de aplicación las utilidades necesarias para generar gráficas que serán necesarias para un requerimiento de SOPHIA.

A continuación se creó un sitio virtual, que contendrá la aplicación, y se le proporcionó una ruta dentro del servidor, donde se alojaran los archivos. Este directorio será el ambiente de desarrollo de la aplicación y permitirá realizar pruebas necesarias sobre la estabilidad del mismo.

Para la etapa de desarrollo, se tendrán repositorios en cada cliente y una vez que se tenga una versión estable, se incorporaran dentro del sitio virtual y se revisaran los cambios buscando aplicar las buenas prácticas de la metodología XP y se liberara el cambio para las pruebas necesarias.

Sin embargo, cada desarrollador podrá realizar las pruebas en su equipo, pues al utilizar Visual Studio 2008, este sistema genera un servidor virtual web, que permite

la compilación del código, la realización de pruebas conforme se van desarrollando los elementos y los cambios en las librerías.

Únicamente se utilizará el motor de base de datos del servidor virtual, para no tener que instalarlo en los equipos cliente y tener las mismas características en cuanto a la codificación de la base de datos.

Por esto mismo, e iniciando con el programa planteado originalmente, comenzaremos con la creación de los objetos en la base de datos.

Primero crearemos la base de datos y el usuario con el que la aplicación accederá al motor de la base de datos, estas tareas son parte de las actividades que realizará la parte administrativa de los servidores, por lo que en desarrollo tendrán un nombre, pero en el ambiente productivo se adaptaran a los que nos asigne éste ente administrativo del CEAQ.

En el ambiente de desarrollo, el nombre de la base de datos será "*BDSOPHIA*" y el usuario para el desarrollo será "*usrsophiadb*", siendo el mismo nombre de usuario su contraseña, recordemos que, este usuario y está base de datos, no serán los productivos, solo la estructura contenida en ellos.

El proceso para la creación de los objetos en la base de datos será:

- Generar todas las tablas
- Agregar sus llaves primarias
- Agregar sus relaciones (llaves foráneas)
- Llenar de catálogos.

Para la implementación de la base de datos se requiere la creación de un único documento con las líneas de código necesarias para que, al liberar la base de datos, se creen los objetos de una sola ejecución.

El desarrollo de las líneas de código lo iniciamos los 2 desarrolladores y fue la traducción a código T-SQL del Diagrama Relacional, el resultado de esta codificación fue el siguiente:

1. Primero indicamos la base de datos que vamos a utilizar:

```
USE [DBSOPHIA]
GO
```

2. Creamos las tablas y sus llaves primarias, a continuación colocaremos unicamente un ejemplo sobre la creación de estos objetos, para visualizar el código:

   El primer ejemplo, muestra la creación de la tabla que contendrá el catálogo con los perfiles.

```
CREATE TABLE [dbo].[SphaCATPerfil](
	[PerID] [int] NOT NULL,
	[PerDescripcion] [varchar](25) NOT NULL,
 CONSTRAINT [PK_SphaCATPerfil] PRIMARY KEY CLUSTERED
(
	[PerID] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

   El segundo ejemplo, muestra la creación de la tabla Usuario, se eligieron estas 2 tablas, dado que una es un catálogo y la otra una tabla, que están relacionadas entre sí.

```
CREATE TABLE [dbo].[SphaUsuario](
	[UsuID] [int] NOT NULL,
	[UsuCorreo] [varchar](50) NOT NULL,
	[UsuPassword] [varchar](50) NOT NULL,
	[PerID] [int] NOT NULL,
	[UsuFechaCreacion] [smalldatetime] NOT NULL,
	[UsuUltimoAcceso] [smalldatetime] NOT NULL,
	[UsuNombre] [varchar](25) NOT NULL,
	[UsuApellidoPaterno] [varchar](15) NOT NULL,
	[UsuApellidoMaterno] [varchar](15) NOT NULL,
	[UsuFechaNacimiento] [smalldatetime] NOT NULL,
	[StaID] [int] NOT NULL,
 CONSTRAINT [PK_SphaUsuario] PRIMARY KEY CLUSTERED
(
	[UsuID] ASC
```

```
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY],
 CONSTRAINT [IX_SphaUsuario] UNIQUE NONCLUSTERED
(
      [UsuCorreo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

3. Agregamos sus relaciones, que en este caso, existe una entre las 2 tablas:

```
ALTER TABLE [dbo].[SphaUsuario]  WITH CHECK ADD  CONSTRAINT
[FK_SphaUsuario_SphaCATPerfil] FOREIGN KEY([PerID])
REFERENCES [dbo].[SphaCATPerfil] ([PerID])
GO
ALTER       TABLE       [dbo].[SphaUsuario]       CHECK       CONSTRAINT
[FK_SphaUsuario_SphaCATPerfil]
```

4. Agregamos los datos a los catálogos. En este caso ejemplificaremos como agregar uno de los perfiles a este catálogo.

```
INSERT INTO [dbo].[SphaPerfil] VALUES(1,'Administrador');
```

El documento que contiene el código necesario para la generación de la estructura de datos contiene más de 1800 líneas, sin embargo consideramos colocar solo un ejemplo para este documento pues con el diagrama relacional es suficiente para codificarlo en este motor o cualquier otro.

En esta etapa no tuvimos complicaciones sobre el desarrollo del script, solamente errores de sintaxis, los cuales eran corregidos en el momento de detectarlos con la herramienta administrativa para el motor de base de datos, se documentó el código para encontrar fácilmente los objetos, relaciones y registros insertados en la base de datos.

Las estructuras XML, que son parte del núcleo del sistema de información SOPHIA fueron codificadas, utilizando la definición que creamos en el capítulo anterior, de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8" ?>
<Collection>
  <Category name="Sistema Oseo" subdirectory="oseo">
    <item id="1" name="Craneo" file="videoCrane.flv" >Este video muestra
la estructura de un craneo.</item>
    <item id="2" name="Rodilla desgastada"
file="videoRodilla_20100106.flv" >Este video muestra el desgaste natural
de la rodilla.</item>
    <item id="3" name="Femur Roto por Accidente"
file="femur_20100202.flv" >Este video muestra la anomalia de un femur
despues de un accidente de bicicleta.</item>
  </Category>
  <Category name="Sistema Neurologico" subdirectory="neurologico">
    <item id="1" name="Neurona con Citicercosis" file="citiNeurona.flv"
>Este video muestra la estructura de una enfermedad que atacá el cerebro
y es mortal.</item>
    <item id="2" name="Muerte Cerebral" file="muerteCerebral.flv" >Este
video muestra la anomalia de un cerebro que no muestra actividad después
de un accidente de auto.</item>
  </Category>
  <Category name="Sistema Muscular" subdirectory="muscular">
    <item id="1" name="Atrofia Muscular file="musculoAtrofia.flv" >Este
video muestra un musculo que ha sufrido daños por falta de
ejercicio.</item>
  </Category>
</Collection>
```

En esta muestra de una de las galerías se muestran 3 catálogos de videos, con la siguiente estructura:

1. Sistema Óseo
   1.1. Cráneo
   1.2. Rodilla desgastada
   1.3. Fémur roto por accidente
2. Sistema Neurológico
   2.1. Neurona con cisticercosis
   2.2. Muerte Cerebral
3. Sistema Muscular
   3.1. Atrofia Muscular

Existen 3 galerías con la misma estructura XML, esto para tener una administración centralizada y estandarizada, fácil de entender y de codificar para cualquier usuario. También existe dentro de la estructura un archivo con la codificación de los Menús, de acuerdo a cada perfil de Usuario. Esta estructura XML será descrita a continuación debido a su importancia como pilar de todo el sistema de información.

```
<?xml version="1.0" encoding="utf-8" ?>
<perfiles>
  <!--Perfil para todo el público-->
  <menu perfil="0">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
  <!--Perfil para el administrador del sitio-->
  <menu perfil="1">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Usuarios">
      <subopcion1 name="Control de Usuarios" page="sphaUsuarios.aspx" />
    </opcion>
    <opcion name="Evaluación">
      <subopcion1 name="Administrar Cuestionarios"
page="sphaCuestionarioAdmin.aspx" />
      <subopcion1 name="Administrar Encuestas"
page="sphaEncuestasAdmin.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="Captura Noticia"
page="sphaNoticiasAdmin.aspx?ref=createnotes" />
      <subopcion1 name="Ver mis Noticias"
page="sphaNoticiasAdmin.aspx?ref=seenotes" />
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
```

```
      <subopcion1 name="Autorización de Noticias"
page="sphaNoticiasAdmin.aspx?ref=authnotes" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Comentarios">
      <subopcion1 name="Ver comentarios" page="sphaComentariosAdmin.aspx"
/>
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Administrar Sitios de Interes"
page="sphaSitiosInteresAdmin.aspx" />
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
  <!--Perfil para todo el Investigador-->
  <menu perfil="2">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="Captura Noticia"
page="sphaNoticiasAdmin.aspx?ref=createnotes" />
      <subopcion1 name="Ver mis Noticias"
page="sphaNoticiasAdmin.aspx?ref=seenotes" />
      <subopcion1 name="Buscar Noticias"
page="sphaNoticia.aspx?ref=seenotes" />
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
  <!--Perfil para el personal de apoyo de Sistemas de Servicio Social-->
  <menu perfil="3">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Evaluación">
```

```
      <subopcion1 name="Administrar Cuestionarios"
page="sphaCuestionarioAdmin.aspx" />
      <subopcion1 name="Administrar Encuestas"
page="sphaEncuestasAdmin.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Administrar Sitios de Interes"
page="sphaSitiosInteresAdmin.aspx" />
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
  <!--Perfil para todo el Alumno-->
  <menu perfil="4">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Evaluación">
      <subopcion1 name="Resolver cuestionario"
page="sphaCuestionario.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
  <!--Perfil para todo el Docente-->
  <menu perfil="5">
    <opcion name="Inicio">
      <subopcion1 name="Volver al Inicio" page="Default.aspx" />
    </opcion>
    <opcion name="Evaluación">
```

```
      <subopcion1 name="Administrar Cuestionarios"
page="sphaCuestionarioAdmin.aspx" />
      <subopcion1 name="Administrar Encuestas"
page="sphaEncuestasAdmin.aspx" />
    </opcion>
    <opcion name="Noticias">
      <subopcion1 name="Captura Noticia"
page="sphaNoticiasAdmin.aspx?ref=createnotes" />
      <subopcion1 name="Ver mis Noticias"
page="sphaNoticiasAdmin.aspx?ref=seenotes" />
      <subopcion1 name="Buscar Noticias"
page="sphaNoticia.aspx?ref=seenotes" />
      <subopcion1 name="RSS" page="sphaRSS.aspx" />
    </opcion>
    <opcion name="Multimedia">
      <subopcion1 name="Ver Videos"
page="sphaMultimedia.aspx?ref=seevideos" />
      <subopcion1 name="Ver Imagenes"
page="sphaMultimedia.aspx?ref=seeimages" />
      <subopcion1 name="Ver Animación 3D"
page="sphaMultimedia.aspx?ref=engine3d" />
    </opcion>
    <opcion name="Sitios de Interes">
      <subopcion1 name="Ver Sitios de Interes"
page="sphaSitiosInteres.aspx" />
    </opcion>
  </menu>
</perfiles>
```

La ventaja de utilizar archivos XML para las galerías multimedia, es que se puede tener un control de versiones, por cada archivo que se genera y puede ser modificada sin necesidad de cambiar ningún elemento del sito, ni tener que recompilar el código.

Con la codificación de estos archivos XML se termina de diseñar el núcleo del modelo por capas, por lo que es momento de acceder a la siguiente capa, los Procedimientos Almacenados y Manejo de Archivos.

En esta capa vamos a codificar en lenguaje SQL todos los procedimientos que nos servirán para insertar, modificar y eliminar registros de las tablas en la base de datos.

Para la creación de estos procedimientos almacenados vamos a crearlos de acuerdo a las reglas que planteamos en el capítulo 2 en cuanto a nomenclatura,

y en caso que el flujo de datos, permita realizar varias acciones sobre una tabla, las colocaremos dentro del mismo procedimiento, mandando llamar cada una a través de diferentes parámetros de entrada.

A continuación vamos a mostrar el desarrollo de uno de los procedimientos, este tiene la función de proveer al sistema de las noticias bajo 3 esquemas:

- Todas las noticias.
- Todas las noticias de un usuario en específico.
- Una noticia específica.

En el caso de este procedimiento, al tratarse de una misma funcionalidad se consideraron reglas en los parámetros de entrada que nos permitieron realizar las 3 acciones en un mismo procedimiento almacenado, esto nos da un mejor control y un mantenimiento más sencillo.

También se consideró un control de versiones al realizar los cambios, los cuales se colocan en el encabezado del procedimiento.

La ventaja de la utilización de esta capa es que se pueden manejar la validación de los datos sin necesidad de modificar las librerías de la aplicación, lo que permite una administración centralizada y provee recursos íntegros.

Todos los procedimientos almacenados fueron creados en la base de datos de desarrollo y se irán modificando de acuerdo a las necesidades del sistema y con el proceso iterativo que nos marca RUP, sin embargo, el objetivo es que, mientras más iteraciones llevemos, menos modificaciones realizaremos a las capas inferiores.

A continuación vamos a ejemplificar un procedimiento donde desde está capa generamos código HTML, pues de acuerdo a las necesidades del sistema era más óptimo la concatenación de palabras en la búsqueda dentro de la base de datos, que analizar el conjunto de resultados y agregar el código correspondiente. Sin embargo, cabe mencionar que el resultado mostrado es una

versión final, después del proceso iterativo, donde ya se colocaron algunas etiquetas de estilo.

También vale la pena decir que, aunque ya venía código HTML, estas cadenas pasaron por las capas correspondientes hasta llegar a la capa que ve el usuario. Los controles de .NET no tuvieron acceso a la base de datos, sino al resultado del proceso.

El procedimiento es el siguiente:

```
USE [DBSOPHIA]
GO

-- =============================================
-- Autores:
-- Fernando A. Girón
-- Iván C. Sanchéz
-- Create date: 6/12/2009
-- Description:
-- 1.0  -> Obtener Noticias
-- 1.01 -> Agregar busqueda de noticias
-- 1.1  -> Agregar control de Usuarios
-- =============================================
CREATE PROCEDURE [dbo].[SphaSPObtenerNoticias](
      @BUSQUEDA AS VARCHAR(100) = '',
      @USUARIO AS INT = 0
)
AS
BEGIN
      SET NOCOUNT ON;
      IF @USUARIO = 0
      BEGIN
            IF @BUSQUEDA = ''
            BEGIN
                  SELECT
                        B.[NotID],
                        '<table width="100%">' +
                        '<tr><td colspan="2" style="text-align:left;
                        padding-left:10px;"><b>' +
                        B.[NotTitulo] +
                        '</b></td></tr>' +
                        '<tr><td style="text-align:right; font-
                        size:10px; padding-right:25px;"><i>Por: ' +
                        A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] +
                        ' ' + A.[UsuApellidoMaterno] +
                        '</i></td></tr><tr><td style="text-align:right;
                        font-size:10px; padding-right:25px;"><i>Fecha
                        de Creación: ' + CAST(B.[NotFechaCreacion] AS
                        VARCHAR)
                        +'</i></td></tr></table>' AS NotDescripcion,
```

```
                        C.[StaNotDescripcion],
                        'Modificar' AS NotComando
                FROM [dbo].[SphaUsuario] A INNER JOIN
                     [dbo].[SphaNoticia] B
                ON A.[UsuID] = B.[UsuID] INNER JOIN
                     [dbo].[SphaCATStatusNot] C
                ON C.[StaNotID] = B.[StaNotID]
                ORDER BY B.[NotFechaCreacion] DESC
        END;
        ELSE
        BEGIN
                SELECT
                       B.[NotID],
                        '<table width="100%">' +
                        '<tr><td colspan="2" style="text-align:left;
                        padding-left:10px;"><b>' +
                        B.[NotTitulo] +
                        '</b></td></tr>' +
                        '<tr><td style="text-align:right; font-
                        size:10px; padding-right:25px;"><i>Por: ' +
                        A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] +
                        ' ' + A.[UsuApellidoMaterno] +
                        '</i></td></tr><tr><td style="text-align:right;
                        font-size:10px; padding-right:25px;"><i>Fecha
                        de Creación: ' + CAST(B.[NotFechaCreacion] AS
                        VARCHAR)
                        +'</i></td></tr></table>' AS NotDescripcion,
                        C.[StaNotDescripcion],
                        'Modificar' AS NotComando
                FROM [dbo].[SphaUsuario] A INNER JOIN
                     [dbo].[SphaNoticia] B
                ON A.[UsuID] = B.[UsuID] INNER JOIN
                     [dbo].[SphaCATStatusNot] C
                ON C.[StaNotID] = B.[StaNotID]
                WHERE B.[NotTitulo] LIKE '%' + @BUSQUEDA + '%' OR
                A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] + ' ' +
                 A.[UsuApellidoMaterno] LIKE '%' + @BUSQUEDA + '%'
                ORDER BY B.[NotFechaCreacion] DESC
        END
END
ELSE
BEGIN
        IF @BUSQUEDA = ''
        BEGIN
                SELECT
                        B.[NotID],
                        '<table width="100%">' +
                        '<tr><td colspan="2" style="text-align:left;
                        padding-left:10px;"><b>' +
                        B.[NotTitulo] +
                        '</b></td></tr>' +
                        '<tr><td style="text-align:right; font-
                        size:10px; padding-right:25px;"><i>Por: ' +
                        A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] +
                        ' ' + A.[UsuApellidoMaterno] +
```

```
                        '</i></td></tr><tr><td style="text-align:right;
                        font-size:10px; padding-right:25px;"><i>Fecha
                        de Creación: ' + CAST(B.[NotFechaCreacion] AS
                        VARCHAR)
                        +'</i></td></tr></table>' AS NotDescripcion,
                        C.[StaNotDescripcion],
                        'Modificar' AS NotComando
                FROM [dbo].[SphaUsuario] A INNER JOIN
                     [dbo].[SphaNoticia] B
                ON A.[UsuID] = B.[UsuID] INNER JOIN
                   [dbo].[SphaCATStatusNot] C
                ON C.[StaNotID] = B.[StaNotID]
                WHERE A.[UsuID] = @USUARIO
                ORDER BY B.[NotFechaCreacion] DESC
        END;
        ELSE
        BEGIN
                SELECT
                        B.[NotID],
                        '<table width="100%">' +
                        '<tr><td colspan="2" style="text-align:left;
                        padding-left:10px;"><b>' +
                        B.[NotTitulo] +
                        '</b></td></tr>' +
                        '<tr><td style="text-align:right; font-
                        size:10px; padding-right:25px;"><i>Por: ' +
                        A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] +
                        ' ' + A.[UsuApellidoMaterno] +
                        '</i></td></tr><tr><td style="text-align:right;
                        font-size:10px; padding-right:25px;"><i>Fecha
                        de Creación: ' + CAST(B.[NotFechaCreacion] AS
                        VARCHAR)
                        +'</i></td></tr></table>' AS NotDescripcion,
                        C.[StaNotDescripcion],
                        'Modificar' AS NotComando
                FROM [dbo].[SphaUsuario] A INNER JOIN
                [dbo].[SphaNoticia] B
                ON A.[UsuID] = B.[UsuID] AND A.[UsuID] = @USUARIO INNER
                JOIN [dbo].[SphaCATStatusNot] C
                ON C.[StaNotID] = B.[StaNotID]
                WHERE B.[NotTitulo] LIKE '%' + @BUSQUEDA + '%' OR
                A.[UsuNombre] + ' ' + A.[UsuApellidoPaterno] + ' ' +
                A.[UsuApellidoMaterno] LIKE '%' + @BUSQUEDA + '%'
                ORDER BY B.[NotFechaCreacion] DESC
        END
    END
END
```

En este ejemplo podemos apreciar cómo no requerimos crear varios procedimientos, sino que, con el manejo de las variables de entrada, podemos controlar el resultado que estamos buscando.

El manejo de errores se llevó a cabo con la generación de excepciones en los procedimientos cuando las variables o los datos no cumplían con la calidad requerida por la aplicación, por lo que, a lo largo del sistema, sin embargo, en las capas de adelante se verá que el manejo de excepciones será por el servidor de aplicación y éste no dará más información a los usuarios finales.

Se crearon 35 procedimientos almacenados que dan servicio a la aplicación de SOPHIA, los cuales no pueden ser incluidos en este documento en su totalidad, dado que el objetivo no es mostrar el desarrollo, sino los resultados obtenidos por la solución.

El manejo de archivos, que es el otro elemento que conforma está capa, se realizó utilizando las librerías nativas del framework; solamente se manejaron los archivos XML y al ser un estándar, se adaptaron perfectamente a las necesidades de esta solución. Estas librerías fueron llamadas por la tercera capa, que fue la encargada de controlar la comunicación entre la base de datos y/o el sistema de archivos con la capa de entidades.

La tercer capa está constituida por 2 clases principalmente, esta es la primera ocasión dentro de la solución donde iniciaremos con la codificación del diagrama de clases y que utilizaremos el lenguaje de C#.

Sin embargo, se tiene que definir la estructura del proyecto en la IDE de Visual Studio, con el fin de tener un control ordenado sobre lo que se está desarrollando.

Vamos a crear el sitio web y vamos a anexar las librerías dentro de un solo proyecto, para no tener que estar utilizando más de un repositorio de archivos, o tener que estar compilando en ambientes separados.

La ventaja de tener una única solución y todas las capas dentro de la misma es que, al momento de realizar la compilación del sitio para realizar las pruebas, todo cambio, tanto en el sitio como en las librerías impacta inmediatamente y nos permite encontrar errores más rápido. Sin embargo, también pueden ser compiladas por separado sin afectar a las otras aplicaciones.

En la IDE de Visual Studio vamos a crear un nuevo proyecto, y lo vamos a definir como se muestra en la siguiente imagen:

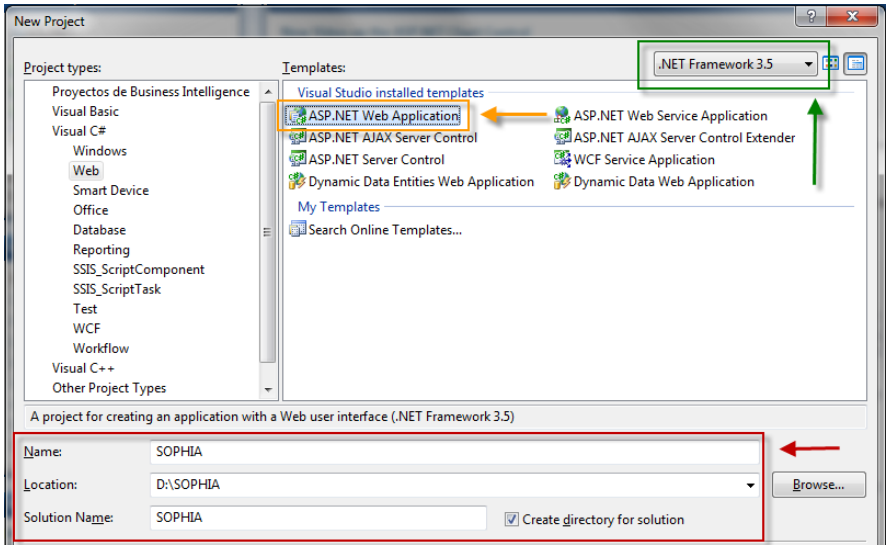

Figura 40. Visual Studio 2008 - Creación de proyecto

Se le indica a la IDE que se creará un proyecto de una aplicación Web de ASP.NET, con el framework 3.5 y el nombre de SOPHIA.

Una vez creada la solución, se le indicará que debe agregar un proyecto de librerías, donde se desarrollarán las capas. Esta solución tendrá las características que se muestran en la imagen:

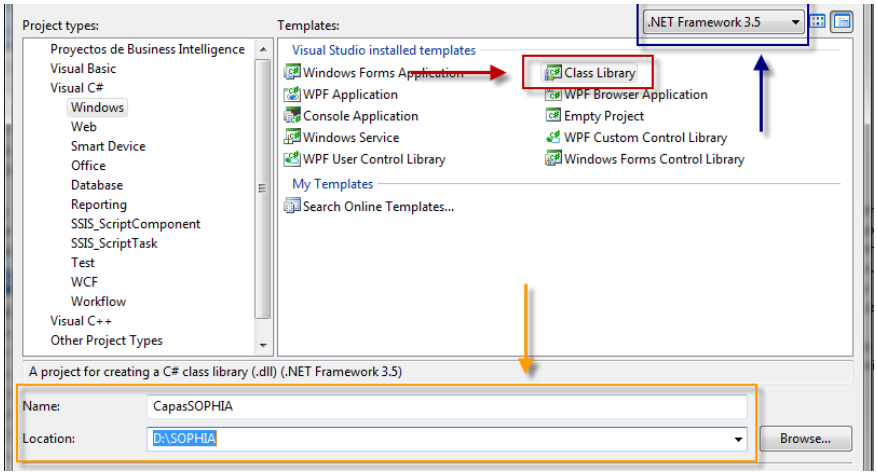

Figura 41. Visual Studio 2008 - Creación de proyecto

Cabe mencionar que la definición vista en las imágenes es solo para ejemplificar el proceso, debido a que, como SOPHIA será un sistema que se encontrará en internet es necesario proteger algunos puntos de acceso desde el exterior, en el caso de la base de datos no es tan necesario dado que estará en una red local protegida por el área de Sistemas de la Facultad de Medicina.

Una vez creada la solución las reglas serán las siguientes:

- Las capas 3, 4 y 5 (únicamente las clases auxiliares) serán parte de las librerías.
- Las capas 5 (modelos de negocio) y 6 serán parte de la aplicación web.
- La capa 3 tendrá acceso controlado, únicamente se podrá acceder a ellas desde la librería.
- Las capas 4 y 5 de las librerías tendrán acceso público a la aplicación.

Con estas reglas y el proyecto creado vamos a iniciar con el desarrollo de las 2 clases de la capa de Control de Acceso. Estas 2 clases tendrán como función la ejecución de los procedimientos almacenados y el procesamiento de los archivos XML.

Por lo que, primero nos evocaremos a comentar sobre el desarrollo de la clase que gestionará el acceso a la base de datos.

Esta clase permitirá que las entidades puedan comunicarse a la base de datos de forma fácil y permitirá que las conexiones a la base de datos sean óptimas, evitando que éstas queden abiertas y generen una negación de servicio.

El proceso será el siguiente:

1. Se crea un objeto, el constructor recibe la cadena de conexión a la base de datos.
2. Se crean los parámetros que requiere el procedimiento almacenado.
3. Se llama al método que ejecuta el procedimiento, este recibe como parámetros de entrada el nombre del procedimiento almacenado y una colección de parámetros necesarios.

4. El objeto crea un enlace con la base de datos
5. Ejecuta la instrucción
6. La base de datos retorna los resultados
7. El método puede regresar:
   a) Una estructura con los resultados divididos en múltiples tablas (de acuerdo a la o las consultas).
   b) Un objeto escalar, con un resultado único.

Esto nos indica que tenemos que generar una clase que controle los parámetros que requiere el procedimiento almacenado, está clase tendrá que tener los siguientes elementos:

- Nombre del parámetro
- Valor
- Tipo de dato

Como se requiere la utilización de más de un parámetro, vamos a utilizar elementos de C# para definir colecciones de objetos, en este caso, una colección de esta nueva clase que vamos a generar.

Podemos utilizar arreglos de objetos (que son permitidos en .NET), estos arreglos son finitos y al declararlos se define su tamaño. Sin embargo, también podemos utilizar listas dinámicas de objetos, que no se define un tamaño inicial, sino que se van agregando los necesarios. Para proporcionar una solución completa, vamos a implementar la solución dinámica que permita a través de la sobrecarga, que se ejecuten procedimientos sin parámetros, con un parámetro o con una colección dinámica de parámetros.

No vamos a colocar todo el código de esta clase, pero si al menos sus definiciones y la documentación con etiquetas XML que ya habíamos comentado al inicio de este capítulo.

```
    /// <summary>
    /// Tipos de Parametros
    /// </summary>
    internal enum TipoParametro
```

```
    {
      …
    }

    /// <summary>
    /// Clase que permite la comunicación con la base de datos
    /// con acceso internal para que solo pueda ser accedido dentro de la
libreria
    /// </summary>
    internal class ControlDataBase
    {
        /// <summary>
        /// Constructor que recibe la cadena de conexión y generá un
objeto de conexión
        /// </summary>
        /// <param name="CadenaConexion"></param>
        public ControlDataBase(string CadenaConexion)
        {
            …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
        /// <param name="Argumentos">Colección de Parametros
dinámico</param>
        /// <returns>Estructura de datos con resultados divididos por
tablas</returns>
        public DataSet EjecutaSP(string StoredProcedure,
List<ParametroDB> Argumentos)
        {
           …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
        /// <param name="Argumento">Parametro Unico</param>
        /// <returns>Estructura de datos con resultados divididos por
tablas</returns>
        public DataSet EjecutaSP(string StoredProcedure, ParametroDB
Argumento)
        {
         …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado sin parametros
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
```

```
        /// <returns>Estructura de datos con resultados divididos por
tablas</returns>
        public DataSet EjecutaSP(string StoredProcedure)
        {
         …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
        /// <param name="Argumentos">Colección de Parametros
dinámico</param>
        /// <returns>Objeto con resultado escalar</returns>
        public object EjecutaSPScalar(string StoredProcedure,
List<ParametroDB> Argumentos)
        {
            …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
        /// <param name="Argumento">Parametro Unico</param>
        /// <returns>Objeto con resultado escalar</returns>
        public object EjecutaSPScalar(string StoredProcedure, ParametroDB
Argumento)
        {
            …
        }

        /// <summary>
        /// Método que ejecuta un procedimiento almacenado sin parametros
        /// </summary>
        /// <param name="StoredProcedure">Nombre del procedimiento
almacenado</param>
        /// <returns>Objeto con resultado escalar</returns>
        public object EjecutaSPScalar(string StoredProcedure)
        {
         …
        }
    }

    /// <summary>
    /// Control de parametros
    /// </summary>
    internal class ParametroDB
    {
        …
        /// <summary>
        /// Tipo de Parametro
        /// </summary>
```

```
        public TipoParametro TipoParam {
            set { this.typeParameter = value; }
            get { return this.typeParameter; }
        }
        /// <summary>
        /// Longitud de la cadena en caso de ser finita
        /// </summary>
        public int Longitud
        {
            set { this.lenghtString = value; }
            get { return this.lenghtString; }
        }

        /// <summary>
        /// Valor del parametro
        /// </summary>
        public object Valor
        {
            set { this.Parameter = value; }
            get { return this.Parameter; }
        }

        /// <summary>
        /// Nombre del parametro
        /// </summary>
        public string Nombre
        {
            set { this.NameParameter = value; }
            get { return this.NameParameter; }
        }



.
        /// <summary>
        /// Constructor de la clase para parametros sin longitud de
cadena
        /// </summary>
        /// <param name="Nombre">Nombre del Parametro</param>
        /// <param name="Valor">Valor</param>
        /// <param name="Tipo">Tipo de Dato</param>
        public ParametroDB(string Nombre, object Valor, TipoParametro
Tipo)
        {
            …
        }

        /// <summary>
        /// Constructor de la clase para parametros con longitud de
cadena
        /// </summary>
        /// <param name="Nombre">Nombre del Parametro</param>
        /// <param name="Valor">Valor</param>
        /// <param name="Tipo">Tipo de Dato</param>
        /// <param name="Longitud">Longitud de la cadena</param>
```

```
        public ParametroDB(string Nombre, object Valor, TipoParametro
Tipo, int Longitud)
        {
            …
        }

    }
```

No se muestran los procesos completos de validación para no sobrecargar este documento, sin embargo podemos apreciar algunas propiedades documentadas, los constructores y los métodos.

En este ejemplo podemos ver la utilización del concepto de sobrecarga en la programación Orientada a Objetos y podemos ver la documentación del código para poder consultarla en tiempo de desarrollo.

La forma de documentar en .NET es con la utilización de etiquetas XML, teniendo principalmente 3 tipos de etiquetas:

- summary. Provee al usuario la descripción sobre el elemento.
- param. Provee al usuario de una descripción sobre un parámetro de entrada para un método o un constructor.
- returns. Provee al usuario una descripción sobre el valor que regresa el método.

Las etiquetas XML siempre se colocan antes del elemento que se desea documentar, y está descripción es interpretada por la herramienta Intellisense para apoyar el desarrollo de la aplicación.

La ventaja de la implementación de nuestro modelo de capas del núcleo hacia el exterior, es que, nos podemos apoyar en todo momento de la documentación con las herramientas que nos provee la interfaz de desarrollo, haciendo este proceso de desarrollo más ágil y con el menor número de errores posibles.

La segunda clase que se implementó en esta capa es el control de acceso; la clase tiene por objetivo recibir el perfil de un usuario, barrer el archivo XML que

posee la descripción de las páginas y generar el código HTML que será el menú de cada usuario.

Esta clase, solo tiene por objetivo la generación del código y refiere su nombre a que crea la estructura para cada perfil, sin embargo, no lleva la seguridad del acceso, esta capa tiene por objetivo la comunicación entre la aplicación y los sistemas externos, la capa que manejará este control lógico será la siguiente.

La utilización de esta clase sigue el siguiente proceso:

1. Se crea el objeto para Generar el Menú
2. Se manda llamar el método que genera el código HTML del menú, se le dan como parámetros el Identificador del Perfil y la ruta donde se encuentra el archivo (PerfilMaster.XML).
3. Retorna el código.

Esta capa utilizará un constructor por default, en un lenguaje orientado a objetos, una clase a la que no se le ha definido un constructor, se le asigna uno predeterminado, que no requiere parámetros de entrada, su función es inicializar las propiedades de la clase.

En esta clase vamos a aplicar el concepto descrito anteriormente concepto, por lo que solo tendremos un método que nos regresará una cadena con el código del menú personalizado para cada perfil.

Al analizar esta capa, nos damos cuenta que es necesario definir la estructura del sitio web para la aplicación, esto debido a que, en el menú se requiere la utilización de código Javascript para que pueda ser un elemento muy visual y tenga estilos definidos (utilizando lenguaje CSS).

Por lo tanto, vamos a crear una estructura donde agrupemos en directorios por código los archivos, y en el directorio principal, vamos a colocar los archivos ASPX, que son aquellos a los que el usuario tiene acceso desde su navegador y a los cuales, la clase que genera el menú, hace referencia.

La estructura será la siguiente:

1. Code. En esta carpeta irán los archivos con código en C# que son la capa 4 que posee las reglas de negocio y el funcionamiento de las páginas que el usuario verá en su navegador. El nombre se coloca en inglés para no hacer una referencia ambigua a código que se puede confundir con cualquier lenguaje.

2. Estilos. En esta carpeta irán los archivos con código CSS y gobernarán los estilos de todas las páginas de la aplicación.

3. Images. En esta carpeta irán todas las imágenes que dan estilo al sitio web, se coloca en inglés para no ser confundida con las imágenes de las galerías multimedia de SOPHIA.

4. Javascript. En esta carpeta irán todos los archivos con código Javascript que se ejecutan en el cliente.

5. XML. Todos los archivos en lenguaje XML que definen estructuras y configuraciones.

A continuación se muestra la estructura de directorios en el proyecto de Visual Studio, la interfaz nos permite llevar el control desde la misma aplicación.

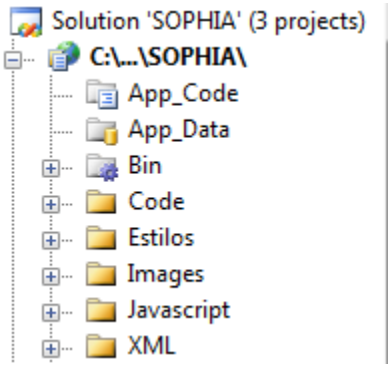

Figura 42. Estructura de sitio web en Visual Studio

Una vez que tenemos ya definida la estructura de archivos, es momento de continuar con la siguiente capa, donde se definen las entidades de SOPHIA.

Esta capa intermedia es la encargada de administrar los recursos que recibe de la base de datos y transformarla en información útil al usuario, y permite tomar los datos de los usuarios y enviarlos a la base de datos de forma correcta a través de las capas inferiores.

La definición de este elemento dentro de la estructura de nuestra aplicación permite que, el desarrollo en las capas superiores sea más sencillo, pues no es necesario conocer la estructura de la base de datos o de los procedimientos almacenados, simplemente se consultan las entidades, se crean los objetos necesarios y se consultan sus propiedades requeridas o se llaman a los métodos de transformación de la información, todas las operaciones se definen en esta capa y los desarrolladores las utilizan de acuerdo al flujo de negocios requerido.

En el caso de esta capa, se va a crear un elemento para realizar las pruebas necesarias, este elemento será un proyecto donde se puedan crear objetos de las clases que estemos probando y verificar su funcionamiento a través de impresiones en consola. Por lo que, se creará un proyecto que mostrará su ejecución en una consola.

La ventaja de desarrollar en la plataforma de .NET es que estas librerías pueden ser ejecutadas tanto en un ambiente web, como en un ambiente de consola, o en una aplicación para Windows. Sin embargo, existen algunas clases que no sucede de esta forma, sin embargo, SOPHIA será un sistema en el que sus capas que no estén casadas con las páginas ASPX, tendrán que funcionar en cualquier tipo de aplicación, no solo web.

Esta capacidad de nuestras librerías nos permitirá realizar pruebas exhaustivas en esta capa, y realizar seguimientos del código, para encontrar errores fácilmente y controlar las excepciones que no se han planeado.

También podremos realizar pruebas de relación entre las entidades y verificar su buen funcionamiento, lo que nos lleva a colocar en este punto un control de calidad y determinar que, hasta que esta capa no cumpla con todas las reglas

del modelo de negocio y con la transformación correcta de la información, no se podrá avanzar a la siguiente capa.

En este momento, se ha terminado la codificación y validación de esta capa, y ya solo resta describir en este documento la forma en que se desarrolló, las pruebas que se realizaron y como se llevó a cabo el control de calidad propuesto.

Cabe mencionar que, esta capa fue la que más tiempo de desarrollo nos llevó debido a que, era donde se concentraba la mayoría de las clases y donde se colocaron pruebas estrictas sobre las reglas de negocio, probadas desde una aplicación externa distinta al que verá el usuario final de SOPHIA.

Esta forma de evaluar la calidad del sistema antes de continuar con las capas superiores se pensó buscando crear entregables más pequeños de las clases, conforme se van desarrollando para cumplir una buena práctica de la metodología XP.

También nos permite explotar las habilidades de desarrollo del equipo, dado que podemos realizar pruebas y mejoras utilizando un ambiente libre, sin procesos formales, permitiendo explotar las formas de trabajar de cada uno y podemos ver un sistema simple, pero funcional.

Este control de calidad nos permite aplicar la mayoría de buenas prácticas de las metodologías ágiles, como lo vemos en el párrafo anterior, y nos deja implementar las siguientes prácticas, que pertenecen a XP:

- Entregas Pequeñas. Cada que se termina una clase y cumple con su proceso de integración en este ambiente de pruebas, se considera una entrega.
- Diseño Sencillo. El diseño de la aplicación es sencillo, dado que su núcleo para realizar las transacciones son objetos fácil de utilizar y actualizar.

- Desarrollo previamente probado. Este control de calidad nos garantiza que el desarrollo está probado previamente a inicializar la integración con el frente del sistema.
- Refactorización. Mientras se realizan pruebas de integración, se puede mejorar el código de las clases previamente liberadas.
- Programación en parejas. La refactorización en las pruebas de integración se realizará en pareja para mejorar el código, utilizando el conocimiento de los desarrolladores.
- Propiedad Colectiva. La integración del código permite que los desarrolladores conozcan la funcionalidad de cada elemento de SOPHIA y puedan realizar mejoras continuamente.
- Integración continúa. Toda clase desarrollada se integra con SOPHIA inmediatamente cuando se termina, lo que permite que el equipo de desarrollo pueda continuar con nuevas clases sin detenerse a recordar los pasos para integrar que realizo en clases que desarrollo mucho tiempo atrás.

Se inició el desarrollo de las clases con el Usuario, esta entidad es aquella que asocia la pertenencia de otras entidades a una persona, por esto es que debe ser la primera clase a desarrollar la cual será probada con las siguientes clases que se desarrollen.

El usuario es muy importante, no solo porque se le asocia la pertenencia de otros elementos dentro del sistema, sino que limita a los usuarios a través de su correo electrónico y una contraseña.

Afortunadamente, la utilización de RUP nos permitió retornar a la capa de los procedimientos almacenados y crear una función que tenga por entrada el correo electrónico y la *contraseña en claro*[39] del usuario, e internamente realiza lo siguiente:

[39] Se refiere a una contraseña que no ha sido cifrada con ningún método, viaja tal y como se tecleo.

- Busca al usuario con el correo electrónico (por lo que se debe modificar la tabla para que el correo sea único para cada usuario) y donde sus contraseñas son iguales, comparando el resultado de la función MD5[40] de la contraseña de entrada y el resultado de la función MD5 de la contraseña que pertenece al usuario.
- Si la búsqueda trae registros, la validación fue correcta.
- Si la búsqueda no trae registros, se genera una excepción en la base de datos que será atrapada por la aplicación.

Se utiliza la función MD5 para comprobar que la contraseña tecleada es idéntica a la almacenada en la base de datos, pues el motor de base de datos puede realizar la búsqueda omitiendo el uso de mayúsculas y minúsculas. Con la comprobación utilizando la función MD5 hacemos una función más robusta en cuanto a seguridad y control de acceso.

Vamos a ejemplificar el resultado de 2 palabras, utilizando la función MD5:

Texto	Función MD5
Password	dc647eb65e6711e155375218212b3964
password	5f4dcc3b5aa765d61d8327deb882cf99
PASSWORD	319f4d26e3c536b5dd871bb2c52e3178
passworD	a61f3f0aee2e87cf0571ca70afe289d2

Para un motor de base de datos, el texto mostrado es igual, pues solo existen cambios de las letras (mayúsculas por minúsculas), pero si vemos la función MD5 resultante de estas palabras, podemos ver que las cadenas resultantes son muy diferentes, lo que nos permite realizar una validación mucho más exacta.

[40] Acrónimo de Message-Digest Algorithm 5, es la representación en 128 bits de la aplicación de un algoritmo con éste nombre, que permite crear una firma digital en este caso de una cadena.

Una vez que se creó el procedimiento almacenado correcto, se inició con el desarrollo de la clase Usuario, el primer constructor que creamos (utilizando el concepto de sobrecarga) fue aquel que recibiera como parámetros de entrada el correo electrónico y la contraseña en claro. El constructor utilizando las capas inferiores solicita los datos a la base de datos, si está regresa resultados, inicializará el objeto con los datos del usuario o bien, si la base de datos genera una excepción, la clase mostrará una excepción y no permitirá la inicialización del objeto.

Más adelante, en la penúltima capa vamos a aprovechar las ventajas de la utilización del servidor de aplicación para almacenar al usuario como un objeto, lo que nos permitirá un mayor control de acceso a la aplicación.

Como se mencionó, utilizando el concepto de sobrecarga, los otros constructores serán los siguientes:

- Un constructor que reciba un identificador del usuario. Permitirá que al ingresar el identificador numérico, que es llave primaria en la base de datos, inicialice el objeto con los datos de este usuario. Si el identificador no existe, la clase generará una excepción y no se inicializará.
- Un constructor que reciba el correo electrónico del usuario. Permitirá que al ingresar el correo electrónico, que es un campo único en la base de datos, inicialice el objeto con los datos de este usuario. Si el correo no existe, la clase generará una excepción y no se inicializará.
- Un constructor sin parámetros. Permitirá inicializar un objeto sin datos, los cuales podrán ser llenados y generar un nuevo usuario para el sistema.
- El constructor ya mencionado, que recibe como parámetros el correo electrónico y la contraseña.

Esta clase tendrá como propiedades los siguientes elementos:

- El Identificador único del Usuario.
- El Correo Electrónico del Usuario.

- La contraseña del Usuario. Esté campo solo servirá para cambiar la contraseña del usuario, pero no permitirá ser accedida desde otras clases, esto se hace indicando en el código, que la propiedad es únicamente de tipo *set*[41].
- Nombres del Usuario
- Apellido Paterno del Usuario
- Apellido Materno del Usuario
- Fecha del último acceso.
- Fecha de Nacimiento.
- Identificador del estado del usuario.
- Descripción del estado del usuario.
- Identificador del Perfil del usuario.
- Descripción del Perfil del usuario.

Al desarrollar las propiedades de las clases de las entidades, se definió una regla que nos permitirá definir claramente las propiedades públicas y las propiedades privadas de la clase. Todas las propiedades están encapsuladas en propiedades con identificadores de acceso *get* o *set* e internamente son propiedades tipo privadas. La regla es la siguiente:

> *"Toda propiedad privada de una clase, será descriptiva en su contenido y a su clase que pertenece, pero su nombre será en inglés y toda propiedad pública de una clase, será descriptiva a su contenido y será en español"*

[41] Una propiedad de una clase puede definirse como *set* (únicamente escritura), como *get* (únicamente lectura) o set y *get* (de escritura y lectura).

Por ejemplo, para usuario haremos un comparativo de sus propiedades privadas y públicas:

Propiedades privadas	Propiedades Publicas
userID	ID
userMail	CorreoElectronico
userName	Nombres
userLastName	ApellidoPaterno
userLastName2	ApellidoMaterno
userProfileID	IDPerfil
userProfileDescription	DescripcionPerfil
userBirthday	Cumpleaños
userLastLogin	UltimoAcceso
userPassword	Contraseña
userStatusID	StatusID
userStatusDescription	StatusDescripcion

Aquí nos damos cuenta que la combinación de los 2 idiomas nos permite hacer clases que en su programación interna sean muy descriptivas, gracias a que utilizamos el idioma Inglés para sus variables y métodos privados, y para los usuarios de estas clases, pero desde el exterior son perfectamente entendibles gracias a que se encuentran en español, tanto las entidades, como sus propiedades y métodos.

Otra ventaja es que, gracias a esta regla es fácil identificar las propiedades al realizar alguna modificación (pudiendo identificar cuáles son públicas y privadas rápidamente) y no nos quita tiempo el tener que pensar en 2 nombres para un mismo concepto de variable al realizar el encapsulamiento.

Esta regla la definimos a través de la experiencia de las primeras, al darnos cuenta que, las entidades tenían muchas más propiedades y era importante estandarizar los nombres para evitar confusiones en una capa superior, o una vez que ya tuviéramos desarrolladas algunas clases y evitar atrasos en los tiempos de entrega.

Terminada de desarrollar la capa referida a las entidades, se realizaron pruebas en un ambiente controlado y se determinó que cada entidad y procesos dentro de ella cumplían con las especificaciones planteadas en la etapa de diseño.

A continuación viene el desarrollo de la capa que controla el modelo de negocio, pues aunque las entidades cumplen esas reglas, se requiere de una capa que administre el proceso y se comunique directamente con el frente de la aplicación, para evaluar y validar que los datos que proporcionan los usuarios son correctos y no generaran conflictos al ingresar al sistema de información.

Esta capa, también posee clases auxiliares que permiten la transformación de los objetos en código HTML, CSS y Javascript para mostrarse a los usuarios en la capa de presentación.

Esta capa tiene como fin la traducción de un modelo abstracto de datos a una vista que el usuario pueda entender perfectamente, o bien, que pueda interactuar con la información y ésta vuelva al modelo abstracto para su almacenamiento.

La utilización de ASP.NET nos permite que, por cada página web que se desarrolla, se crea una clase relacionada a esta página y este objeto hereda los métodos y propiedades de una clase llamada Page que contiene los elementos necesarios para poder enviar, controlar y recibir los parámetros desde el usuario, con esto podemos interactuar directamente con los controles propios de ASP.NET y con elementos HTML.

Estos controles son procesados por el servidor de aplicación y la función de la capa 5, es realizar la transformación a código HTML, CSS y Javascript que finalmente se mostrará al usuario en cualquier navegador web.

Esta capa constará de los siguientes elementos:

- Clases de Modelo de Negocio.
    - Administración de Comentarios.
    - Administración de Cuestionarios.
    - Resolución de Cuestionarios.
    - Administración de Encuestas.
    - Administración de Noticias.
    - Administración de Usuarios.
    - RSS.
    - Administración de Sitios de Interés.
    - Visor de Sitios de Interés.
    - Visor de Información Principal
    - Control de Sesión, Menú y Encuestas.
- Clases Auxiliares
    - Generales de SOPHIA
    - Generación de Menú Dinámico.

Cada una de las clases del modelo de negocio representa una página web en el sitio web de la aplicación, la ventaja de la utilización de esta tecnología es que no se requiere crear varios sitios por cada acción que se requiera, sino que se pueden agrupar en módulos de administración por cada tema o entidad que se requiera y dentro del diseño de la capa de aplicación y la capa de modelo de negocio se puede dar la funcionalidad requerida por cada acción del sistema.

El trabajar con la capa de negocio nos permite convivir directamente con la capa de aplicación, pues es necesario insertar dentro del código HTML las etiquetas de los controles de ASP.NET para poder dar presentación y vista al sitio web.

La experiencia del desarrollo de aplicaciones web utilizando está tecnología nos indica que se debe trabajar en paralelo estas 2 capas, aprovechando así las capacidades del equipo e implementando las buenas prácticas de la metodología XP, se trabajó en simultaneo el diseño visual y la transformación de la información.

Aprovechando las ventajas de la utilización de un sistema en ASP.NET corriendo en un servidor de aplicación con IIS, vamos a utilizar una configuración centralizada de 3 elementos:

- La cadena de configuración a la base de datos
- El directorio temporal donde se almacenaran las imágenes de las gráficas.
- La cadena de la versión que se está corriendo en ese momento.
-

Esto se hace en un archivo de configuración llamado web.config, el cual se administra con el servidor de aplicación y puede ser llamado desde cualquier elemento del sitio web y posee la centralización de los archivos de configuración para un fácil mantenimiento o bien para realizar cambios en la estructura de los servidores de una manera más controlada.

Vamos a ejemplificar 2 módulos de estas capa, el desarrollo del servicio RSS (utilizando únicamente la capa de modelo de negocio y etiquetas ASP.NET) y la implementación del menú (que utiliza la capa del modelo de negocio y un objeto auxiliar para generar el código HTML y se apoya de un control de ASP.NET para su implementación, así como de código Javascript y CSS para su presentación, considerando a este el objeto más completo para ejemplificar.

El moduló RSS consta de 2 elementos, el modelo de negocio, desarrollado en C# y la página web a la que se accederá para ver el contenido. A continuación se muestran los 2 elementos:

Código C# (sphaRSS.aspx.cs)

```
public partial class RSS : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        // Realizamos la configuración de la página indicando que será un
código XML el obtendrá
        Response.Clear();
        Response.ContentType = "text/xml";

        //Indicamos la estructura de canalr RSS
        XmlTextWriter feed = new XmlTextWriter(Response.OutputStream,
Encoding.UTF8);
        feed.WriteStartDocument();
        feed.WriteStartElement("rss");
        feed.WriteAttributeString("version", "2.0");
        feed.WriteStartElement("channel");
        feed.WriteElementString("title", "Centro de Enseñanza y
Adiestramiento Quirurgico");
        feed.WriteElementString("link", "http://www.unam.mx");
        feed.WriteElementString("description", "Sitio con recursos del
Centro de Enseñanza y Adiestramiento Quirurgico de la UNAM");
        feed.WriteElementString("copyright", "Copyright 2009
www.ingenieria.unam.mx. Derechos Reservados.");

        //Creamos un objeto de la entidad noticia
        Noticia Notes = new Noticia();

        //Obtenemos una colección de las 20 noticias mas Recientes
        //Aquí enviamos la cadena de conexión a la base de datos,
utilizando el web.config
        List<Noticia> Resultado =
Notes.ObtenerTop20Noticias(ConfigurationManager.ConnectionStrings["DBSOPH
IA"].ConnectionString);

        //Creamos cada elemento de las noticias, en el canal RSS
        foreach (Noticia post in Resultado)
        {
            feed.WriteStartElement("item");
            feed.WriteElementString("title", post.Titulo);
            feed.WriteElementString("description", post.Cuerpo);
            feed.WriteElementString("link",
"http://localhost:52719/SOPHIA/Default.aspx?udata=" + post.ID.ToString()
+ "&conf=" + GetMD5(DateTime.Now.AddDays(post.ID).ToLongDateString()));
```

```
feed.WriteElementString("pubDate",post.FechaCreacion.ToString());
            feed.WriteElementString("author",
post.Creador.CorreoElectronico + "(" + post.Creador.Nombres + " " +
post.Creador.ApellidoPaterno + ")");
            feed.WriteEndElement();
        }

        feed.WriteEndElement();
        feed.WriteEndElement();
        feed.WriteEndDocument();

        //Enviamos el contenido XML al Usuario
        feed.Flush();
        feed.Close();
        Response.End();

    }
}
```

Sin embargo el usuario deberá acceder a otro archivo, denominado sphaRSS.aspx, éste se muestra a continuación:

sphaRSS.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="~/Code/sphaRSS.aspx.cs" Inherits="RSS" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Podemos ver que al inicio de está página se le indica al servidor que es una página, desarrollada en lenguaje C#, que tiene asociado un archivo de código (mediante la propiedad CodeFile) llamado sphaRSS.aspx.cs (ya ejemplificado) y tiene una clase asociada llamada RSS (mediante la propiedad Inherits).

Esté es un ejemplo básico y al final del proceso de ejecución, el servidor arroja al cliente una cadena XML que es interpretada por un lector RSS.

El siguiente ejemplo lo vamos a ver por partes, de acuerdo al proceso que sigue.

Cuando un usuario accede a SOPHIA, requiere tener un menú que le muestre lo que puede realizar con el sistema, ese menú es personalizado y depende del perfil del usuario, para esto ya se definió una estructura XML que contiene la definición de las páginas que puede ver cada perfil, sin embargo, es necesario trasnformarlo a código HTML que pueda ser interpretado por el navegador web.

La aplicación posee un control de sesión, que en realidad es una página aspx (denominada MasterPage.master en una aplicación de ASP.NET) que aparece como encabezado en toda página de la aplicación de SOPHIA, en está página será donde colocaremos nuestro menú, para que puedar ser accedido a él en cualquier punto de la aplicación.

Dentro de está página denominada MasterPage.master se coloca la siguiente linea:

```
<asp:Literal ID="controlMenu" runat="server" />
```

Este es un control de ASP.NET que nos permite insertar en la página web cualquier cadena de texto (ya sea solo texto o bien, código HTML con Javascript y CSS). La línea será gestionada por la parte de la capa del modelo de negocio, que mandará llamar a una clase auxiliar, de la siguiente forma:

```
GeneraMenu Menu = new GeneraMenu();
controlMenu.Text = Menu.MuestraMenu(Server.MapPath(Path), 0);
```

Aquí le indicamos a la aplicación que asigne el valor que le arroja el método MuestraMenu al control de ASP.NET, esta cadena contiene los 3 códigos que se utilizan en la capa de la aplicación. Dicha cadena se generá en la clase auxiliar llamada GeneraMenu, esta se produce de la siguiente forma:

```
public class GeneraMenu
    {
        public string MuestraMenu(string Path, int idPerfil)
        {
            StringBuilder Salida = new StringBuilder();
            XmlDocument xmlfile = new XmlDocument();
            xmlfile.Load(Path + "/XML/PerfilMaster.xml");
            XmlNodeList Perfil =
xmlfile.GetElementsByTagName("perfiles");
            foreach (XmlElement menu in (XmlElement)Perfil[0])
            {
                if (Convert.ToInt32(menu.GetAttribute("perfil")) ==
idPerfil)
                {
                    Salida.AppendLine("<table class=\"menuTable\"
cellpadding=\"0\" cellspacing=\"0\">");
                    Salida.AppendLine("<tr>");
                    XmlNodeList opcion =
menu.GetElementsByTagName("opcion");
                    foreach (XmlNode opcionNodo in opcion)
                    {
                        Salida.AppendLine("<td
class=\"menuOpcionInactive\">");
                        Salida.AppendLine("<a class=\"vinculoMenu\"
onmouseover=\"showDiv('div_" +
opcionNodo.Attributes["name"].Value.Replace(" ", "_") + "')\"
href=\"javascript:void(0)\">");

Salida.AppendLine(opcionNodo.Attributes["name"].Value);
                        Salida.AppendLine("</a>");
                        XmlNodeList submenu = opcionNodo.ChildNodes;
                        Salida.AppendLine("<div id=\"div_" +
opcionNodo.Attributes["name"].Value.Replace(" ", "_") + "\"
class=\"divMenu\" style=\"display:none;\" >");
                        Salida.AppendLine("<table cellspacing=\"0\"
cellpadding=\"0\">");
                        Salida.AppendLine("<tr><td>");
                        Salida.AppendLine("<img
src=\"./Images/AppFrontMP/SubMenuTop.png\" alt=\"\" />");
                        Salida.AppendLine("</td></tr>");
                        foreach (XmlNode subMenuOpcion in submenu)
                        {
                            Salida.AppendLine("<tr><td align=\"left\"
class=\"rowSubMenu\">");
                            Salida.AppendLine("<a id=\"" +
subMenuOpcion.Attributes["page"].Value + "\" href=\"" +
subMenuOpcion.Attributes["page"].Value + "\" style=\"color:White;\"
onmouseover=\"opcionSubMenuSelected('" +
subMenuOpcion.Attributes["page"].Value + "')\"
onmouseout=\"opcionSubMenuNonSelected('" +
subMenuOpcion.Attributes["page"].Value + "')\" >");

Salida.AppendLine(subMenuOpcion.Attributes["name"].Value);
                            Salida.AppendLine("</a>");
                            Salida.AppendLine("</td></tr>");
```

```
                        }
                        Salida.AppendLine("<tr><td>");
                        Salida.AppendLine("<img
src=\"./Images/AppFrontMP/SubMenuBottom.png\" alt=\"\" />");
                        Salida.AppendLine("</td></tr>");
                        Salida.AppendLine("</table>");
                        Salida.AppendLine("</div>");
                        Salida.AppendLine("</td>");
                    }
                    Salida.AppendLine("<td>");
                    Salida.AppendLine("<img alt=\"Menu\"
src=\"Images/AppFrontMP/finMenu.png\" />");
                    Salida.AppendLine("</td>");
                    Salida.AppendLine("</tr>");
                    Salida.AppendLine("</table>");
                }
            }
            return Salida.ToString();
        }
    }
```

Aquí podemos ver como se van concatenando cadenas HTML que van generando la estructura básica del menú, y en lugar de inyectar directamente los códigos CSS y Javascript, se hace a través del mismo HTML.

Para los estilos con lenguaje CSS, se utiliza la propiedad class de las etiquetas HTML, que indican al navegador web que tiene que buscar dentro de las definiciones CSS algúna que contenga el nombre que se le indica en esa propiedad (por ejemplo class="rowSubMenu"). A continuación ejemplificamos el código CSS con ese nombre, que se encuentra dentro de los archivos de definición de CSS:

```
.rowSubMenu
{
    background-image:url('../Images/AppFrontMP/SubMenuMiddle.png');
    background-repeat:repeat-y;
    padding-left:10px;
    padding-top:5px;
    padding-bottom:5px;
}
```

Para el código Javascript que se ejecutará en el cliente, se indica dentro del código HTML la función que se llamará, a través de propiedades que indican el manejo de eventos por parte del navegador web (por ejemplo, onmouseover="opcionSubMenuSelected('pagina.aspx')", que le indica al

navegador que, cuando el puntero del dipositivo de navegación se coloque sobre el elemento se disparé esa función). La función se almacena en los archivos que posee el código Javascript, a continuación se ejemplifica ésta función:

```
function opcionSubMenuSelected(id) {
    clearTimeout(ControlTime);
    document.getElementById(id).parentNode.className =
"rowSubMenuSelected";
}

function showDiv(id) {
    clearTimeout(ControlTime);
    var divMenu =
document.getElementById('contextMenu').getElementsByTagName('div');
    var posX = findPosX(document.getElementById(id).parentNode);
    var posY = findPosY(document.getElementById(id).parentNode) + 30;
    for (var counter = 0; counter < divMenu.length; counter++) {
        if (divMenu.item(counter).id == id) {
            $("div#" + id).show("slow");
            divMenu.item(counter).parentNode.className =
"menuOpcionActive";
            divMenu.item(counter).style.left = posX + "px";
            divMenu.item(counter).style.top = posY + "px";
        }
        else {
            divMenu.item(counter).parentNode.className =
"menuOpcionInactive";
            $("div#" + divMenu.item(counter).id).hide("slow");
        }
    }
    ControlTime = setTimeout("closeAllDiv()", 3000);
}
```

Con esto creamos la funcionalidad total del menú y tan solo falta agregar a la página web la indicación que incluya los archivos con los códigos CSS y Javascript para ver un menú totalmente funcional creado a partir de una estructura XML. Para agregar los archivos con los códigos adicionales se agregan las siguientes líneas:

```
<link rel="Stylesheet" type="text/css" href="Estilos/estilos.css" />
<script language="javascript" type="text/javascript"
src="Javascript/javascript.js"></script>
```

Después de la codificación de este ejemplo podemos ver por primera vez un elemento del producto terminado, respetando las reglas de nuestro diseño por capas.

Figura 43. Menú de SOPHIA

A continuación iniciamos con el desarrollo de los elementos utilizando ActionScript y Flash, estos son para la parte multimedia de SOPHIA.

Galería de imágenes

Para la creación de la galería primero tenemos que leer el archivo XML que contiene la información de las imágenes como es: categorías, ruta de la imagen miniatura, ruta de la imagen en su tamaño completo y descripción de la imagen. Una vez que tenemos cargados estos datos en una estructura XML de Flash podemos navegar por cada nodo y sus respectivos hijos para ir formando nuestro catálogo de imágenes y a su vez se crean subcatálogos por categoría que nos muestra las imágenes de cada galería, con métodos de librerías de action script damos funcionalidad a nuestra galería creando eventos para cuando el usuario de clic a una categoría para que cargue la lista de imágenes y a su vez se crean los eventos para cuando el usuario da clic en la imagen y esta se despliegue en su tamaño original y también despliegue el icono de información sobre la imagen.

Se han colocado transiciones y animaciones en toda la galería para que sea más atractiva para el usuario al momento de navegar por ella.

Galería de videos

En la galería de videos se trabajó de manera similar a la galería de imágenes en lo que respecta a la lectura del archivo XML y la separación por categorías, pero en esta ocasión no se va a desplegar de la misma manera la información contenida de cada categoría en el XML, ya que en esta ocasión será puesta en una lista con el nombre del video, el usuario al dar clic sobre el nombre cargará el video.

También se tuvo que crear un reproductor de video que lea el streaming de video, esto realmente no resultó muy complicado puesto que ya se cuenta con las funciones que leen este tipo de datos y se despliegan en una pantalla de video definida. Se crearon botones y barra de volumen y de estado de la película para que el usuario interactué enriqueciendo la interactividad con los videos y también se agregó un panel en el que se describe el video.

Visor de modelos tridimensionales

Para la elaboración de este visor se tiene que trabajar con la librería Papervision 3D ya que contiene las funciones que nos permitirán cargar y manipular los modelos tridimensionales, estos al igual que las galerías anteriores se encuentra estructurado en un archivo XML el cual separa a los modelos por categoría, pero también contiene la información sobre la ruta del modelo y su textura, así como una breve descripción de lo que se está mostrando. El usuario podrá escoger el modelo a visualizar escogiéndolo por su nombre de una lista similar a la de videos y podrá manipular el ángulo del cual ve el modelo por medio del mouse.

Se necesitaran crear modelos especiales para web, ya que los actuales son muy grandes con respecto al número de polígonos lo cual hace muy pesado al modelo y difícil de manipular, más si se van a distribuir por web, por lo tanto se plantea un proceso por el cual se obtendrán modelos de baja resolución pero que contendrán, en lo posible, detalles de los modelos originales.

Con esta etapa damos por terminada la etapa de desarrollo, por lo que en este momento se genera la versión 1.0 Beta de SOPHIA.

Es una versión Beta dado que debe pasar por la etapa de pruebas y despliegue, para por fin generar el entregable que será funcional para el centro y cumplirá con todos los requisitos planteados en el documento visión, que recordemos que sería nuestra guía durante el proceso de diseño y desarrollo, y plantearía nuestros objetivos para poder entregar un producto funcional y que pueda ser explotado por el centro, para cambiar su modelo del negocio.

Este capítulo tuvo como objetivo la traducción de los distintos diagramas que seleccionamos de la metodología RUP a una aplicación funcional escrita en los diferentes lenguajes que seleccionamos, toda esta codificación se realizó utilizando las buenas prácticas de la metodología XP.

Tuvimos complicaciones en la etapa de desarrollo para poder hacer la combinación de todos los lenguajes sin que estos nos generaran errores. La principal de éstas fue poder realizar la manipulación de los controles de ASP.NET desde el cliente y que estos se reflejen en el servidor de aplicación, así como la manipulación de los eventos de los controles con tecnología AJAX, como son las llamadas asíncronas al servidor web.

Estas complicaciones las resolvimos creando ambientes controlados de pruebas, utilizando controles de ASP.NET y combinándolos con códigos Javascript, y mientras realizábamos las pruebas, hacíamos seguimiento línea por línea de la forma en que se manipulaba la información y se trasladaba nuevamente al servidor de aplicación.

También resolvimos la forma en que enviábamos los parámetros necesarios de configuración a los elementos desarrollados con Flash, utilizando etiquetas en el cliente que pueden generarse dinámicamente desde el servidor.

El diseño visual del sistema se realizó considerando el logo actual del sitio, debido a que este ya existe y se encuentra en muchos documentos oficiales, por lo que los colores y formas se basan en este elemento.

Ahora debemos continuar con el proceso planteado y proseguiremos con el capítulo IV, referido a las pruebas y el despliegue del sistema en el centro y a la comunidad de la Facultad de Medicina. Sin embargo, es necesario presentar el primer despliegue del sistema SOPHIA en un ambiente de desarrollo, previo a pasar a la siguiente etapa.

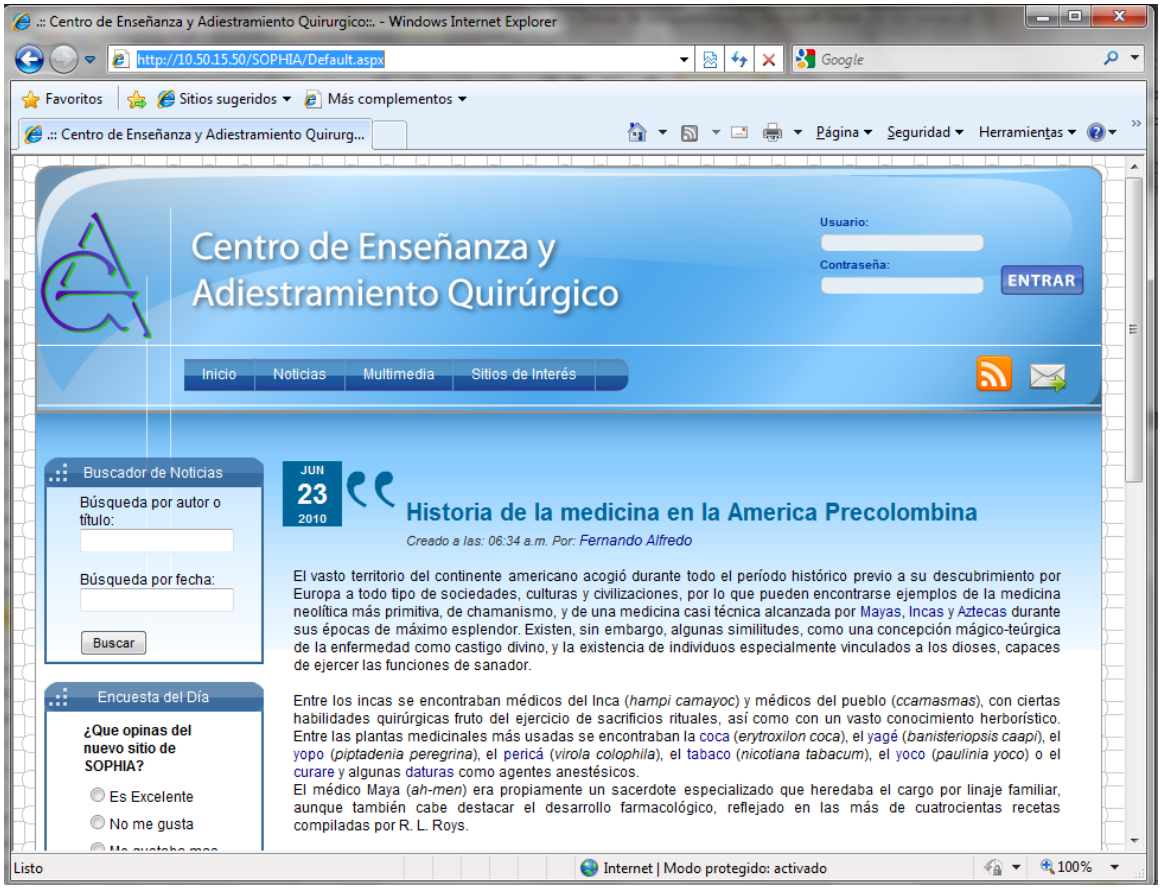

Figura 44. Imagen de página principal de aplicación de SOPHIA

Capítulo IV. Pruebas y Despliegue

Introducción

En esta etapa realizamos las acciones necesarias para verificar que todo lo descrito en el documento visión se encuentra en el sistema que se terminó de desarrollar, esto para validar que es un producto que cumple con las especificaciones del sistema y proporciona las herramientas necesarias para cumplir con el despliegue del nuevo modelo propuesto para el negocio.

Si el sistema cumple con las especificaciones, podemos decir que la solución está completa y es momento de evaluar si el proceso de reingeniería ha sido exitoso, buscando determinar en primer lugar si la información fluye correctamente del centro hacía sus usuarios y en segundo lugar, si el centro recibe retroalimentación por parte de sus usuarios para poder realizar un proceso de mejora.

Es muy importante plantear desde el inicio de esta etapa que para la validación del sistema y la realización de pruebas, se utilizará como base el documento visión generado durante el capítulo 2, en el que planteamos los objetivos de ésta primer versión además de contener las especificaciones del producto que vamos a entregar al centro.

En esta etapa tampoco vamos a realizar un análisis sobre nuevos requerimientos, pues aunque la estructura se pensó para poder tener expansiones más rápidas y un mantenimiento muy sencillo, el propósito de este proyecto es el análisis de la situación actual, la propuesta de una solución de reingeniería y la puesta en producción de una herramienta que se creará a partir de ésta.

Una vez que hayamos validado el producto y que garanticemos que cumple con todas las especificaciones propuestas al inicio, es momento de iniciar con el proceso de despliegue y la capacitación para el uso del mismo.

El despliegue del sistema consistirá en crear las librerías necesarias, el proyecto web, la base de datos, los usuarios y la configuración del servidor productivo, para poder iniciar con su operación por parte del centro y de los usuarios.

Más adelante, se creará un proceso de despliegue diferente al que se pensó al iniciar con la solución, y esto es debido a que, no depende únicamente del centro o de nosotros, sino que la implementación de un nuevo servidor requiere un proceso administrativo por parte del centro y de la Facultad de Medicina, el cual no puede ser gestionado por el equipo que participa en este proyecto además de existir otros factores que se deben considerar.

Sin embargo, el sistema de información SOPHIA iniciará operaciones para el personal del centro, en servidores virtualizados y posteriormente, ya que se tenga una base de información considerable, el producto será liberado a los usuarios finales del centro.

Esto tiene 3 fines, el primero es que, los investigadores, administradores de la aplicación y personal de apoyo del centro tengan un conocimiento total sobre el funcionamiento y utilización del sistema, el segundo es que, se analicen nuevos requerimientos para futuras versiones y que el personal a cargo de la aplicación pueda consultar con los creadores del proyecto sobre la utilización de la estructura creada y se faciliten las actualizaciones y por último, que se tenga suficiente material (tanto artículos, como material multimedia y avisos) que permitan que los usuarios finales puedan aprovechar desde un inicio el sitio web y no pierdan el interés dejando de proporcionar al centro de información útil, sin embargo este tema se tratará más adelante.

Evaluación del sistema

Para la evaluación del sistema tendremos que garantizar que pase por un proceso de Control de Calidad, que nos permitirá dictaminar que el producto de software cumple con los procesos, políticas y estándares especificados desde un inicio.

Antes de continuar cabe mencionar que el sistema de información SOPHIA y su aplicación siempre estarán en mejora continua, por lo que, tendremos que definir que se considerará un error aquello que no cumpla el proceso de evaluación y no se encuentre dentro de la planeación de la primer versión.

La calidad de un software para los negocios la podemos definir como: El grado en que un usuario percibe que el producto satisface sus necesidades o la especificación de sus requerimientos.

Sin embargo, la calidad de SOPHIA será evaluada por el equipo de desarrollo en pro de garantizar la funcionalidad total del producto, sin que posea errores de programación o bien, que el usuario no perciba leyendas o elementos que se agregan durante el desarrollo y que por descuido del equipo olvidara quitar.

La evaluación por parte del negocio será sobre los requerimientos planteados inicialmente, y cambiarán a lo largo de la vida del sistema, pues esta primer versión tiene por objetivo crear una herramienta que permita un flujo bidireccional entre el centro y sus usuarios, su navegabilidad será objeto de versiones posteriores, ya que el centro podrá evaluar el uso de la tecnología en pro de su trabajo y abrirá su mente a poder pedir requerimientos específicos visuales y de programación para interactuar de forma más natural con el sistema.

Se podría decir que, el producto que vamos a evaluar es el corazón de SOPHIA, su sistema de información, y la aplicación será el primer acercamiento de los usuarios hacia un software que les permite disminuir los tiempos de traslado su información entre el centro y la comunidad de alumnos de la Facultad de Medicina.

Podemos mencionar que por tratarse de la metodología RUP, durante la etapa de desarrollo, pasamos muchas veces por la etapa de pruebas (recordemos que la metodología es iterativa incremental), sin embargo es necesario realizar una etapa de pruebas totalmente dedicada al producto final.

En este proyecto, innovaremos considerando los resultados de las pruebas en 2 vertientes:

- Los errores de programación y mejor manejo de las excepciones serán corregidas durante el proceso iterativo de evaluación

- Las mejoras requeridas, serán pospuestas para la siguiente versión del sistema, buscando la intervención de los usuarios para realizar mejoras enfocadas en sus nuevas necesidades.

El proceso de evaluación lo podemos definir de la siguiente forma:

Figura 45. Proceso de validación

1. Análisis del requerimiento: Se tomará un flujo de los diagramas de estados y se analizará los resultados que se esperan al finalizar el proceso.
2. Creación de un escenario de pruebas: Se definirán las variables de entrada y los resultados esperados en función de esas variables.
3. Verificación de estándares y requerimientos: Se evaluará el flujo a través del sistema y se obtendrán los resultados especificados.
4. Evaluación de errores y fallas: Se comparará la salida del flujo con los resultados esperados.

5. Corrección de fallas de programación: Si el error detectado es debido a programación se corregirá y se evaluará nuevamente.
6. Documentación de mejoras: Si se refiere a una mejora de navegación o presentación, se documentará para una versión posterior.

Al utilizar la metodología RUP, nos permite llegar a esta etapa con el menor número de errores, esto tiene como ventaja que el costo de reparación de los mismos es mucho menor que en un ambiente productivo y se garantiza la calidad de datos capturados.

La forma de documentar las pruebas realizadas será a través de una matriz de pruebas, que es un documento que nos permite puntualizar las pruebas que vamos a realizar, los resultados esperados, los resultados obtenidos y un estado sobre el resultado de las pruebas.

En un futuro, ya que se encuentre en su versión productiva la aplicación se deberá implementar un sistema de seguimiento de errores, para poder llevar un control específico sobre los cambios y versiones liberadas.

En primer lugar vamos a definir los campos requeridos para la matriz de pruebas y posteriormente los procedimientos que evaluaremos en este documento, aunque en realidad se han evaluado todos los diagramas de estados.

Proceso	Función	Prueba	Resultado esperado	Resultado Obtenido	Comentarios

- Proceso: El diagrama de estados que se está evaluando.
- Función: La acción general que será probada.
- Prueba: La acción específica que será evaluada.
- Resultados esperados: De acuerdo con las variables de entrada, que se espera del sistema.
- Resultados obtenidos: El resultado arrojado por el sistema.
- Comentarios: El resultado de la evaluación.

Para este documento vamos a elegir 2 procesos completos, uno es el control de usuarios debido a su importancia para el flujo de la información que será descrito con pantallas de la ejecución de las pruebas y el resultado obtenido; otro será la creación y publicación de las noticias debido a que es necesario proteger el dominio de la Facultad de Medicina será descrito únicamente con el resultado de la matriz de pruebas y se detallaran las complicaciones que tuvimos para validarlo.

Primero vamos a analizar los pasos que debemos realizar para poder ser usuario de SOPHIA:

1. En primer lugar necesitamos dar a de alta a un Usuario a través del módulo de administración de Usuarios.
2. El sistema validará que el correo registrado no exista, si ya existe, lo notificará al administrador.
3. El administrador de usuarios, podrá modificar los datos de cualquier usuario del sistema.
4. Una vez dado de alta el nuevo Usuario, vamos a ingresar al sistema con el correo electrónico tanto en el campo de usuario como en la contraseña
5. El administrador podrá reiniciar la contraseña de un usuario o darlo de baja en el sistema.
6. El sistema validará que somos un usuario nuevo y nos deberá solicitar una contraseña, para validar los datos, requerirá la fecha de nacimiento.
7. Capturada la contraseña, el sistema pedirá al nuevo usuario que ingrese nuevamente, ahora con su nueva clave.
8. Si los datos son correctos, el sistema mostrará el menú personalizado, de lo contrario, enviará un mensaje de error.
9. El nuevo usuario podrá cambiar su contraseña cuando así lo desee.

Una vez que sabemos los pasos que debemos seguir, podemos crear una matriz de prueba específica para este proceso. En la que mostramos aquí, omitiremos la columna del proceso debido a que será específica a esta evaluación, sin

embargo el la documentación completa, si se encuentra, debido a que se creó una matriz para todo el sistema.

Función	Prueba	Resultado esperado
Alta de Usuario	Validar que no exista un correo electrónico	Si el correo se repite, no permitir enviar los datos. Si el correo no existe, permitir enviar los datos.
Alta de Usuario	El administrador podrá dar de alta los datos de un nuevo usuario	Se almacenaran correctamente los datos del usuario y lo notificará al administrador
Alta de Usuarios	El administrador podrá modificar los datos de un usuario	Se almacenaran correctamente los datos del usuario y lo notificará al administrador
Alta de Usuarios	El Administrador podrá bloquear a un usuario	Cuando el usuario trate de entrar al sistema, se le negará el acceso argumentando que sus datos están mal
Alta de Usuarios	El Administrador podrá reiniciar la contraseña de un usuario	Cuando el usuario ingrese al sistema, le pedirá su nueva contraseña, como si fuese su primera vez.
Inicio de Sesión	Primer inicio de un usuario o primer inicio una vez que la contraseña fue reiniciada	Al usuario se le permitirá capturar su nueva contraseña y se le notificará
Inicio de Sesión	Inicio regular de un usuario	Si los datos son correctos, se le permitirá el acceso y se le mostrará su menú personalizado. Si los datos son incorrectos, se le indicará al usuario.

Las pruebas que son realizadas en el sistema serán registradas de la siguiente forma:

En primer lugar iniciamos una sesión con un usuario ya registrado y que, en este caso, es administrador del Sistema, los datos serán incorrectos en esta primera prueba y el resultado obtenido es:

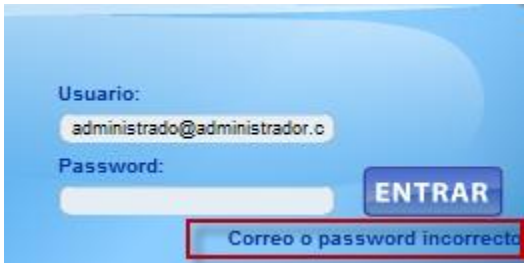

Figura 46. Inicio de Sesión Fallido

Ahora iniciaremos sesión con datos correctos:

Figura 47. Inicio de información correcto

Como podemos ver el sistema reconoció los datos y nos mostró el menú personalizado para un usuario administrativo.

Ahora con esta sesión, vamos a ingresar un nuevo usuario, en primer lugar vamos a ingresar un correo que ya existe dentro del sistema SOPHIA, para verificar que se cumpla esta regla de negocio.

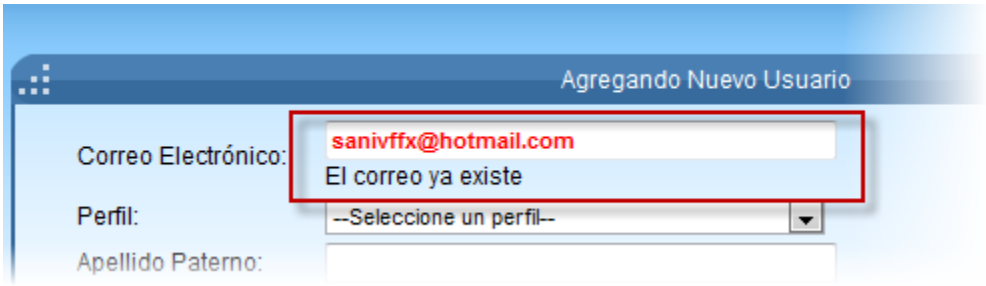

Figura 48. Nuevo usuario, cuando ya existe el correo electrónico

Como podemos verificar, el sistema le notifica al usuario que ya existe este correo electrónico registrado y no permite continuar con el registro del usuario.

A continuación colocamos un nuevo correo electrónico y el resultado de la validación fue el siguiente:

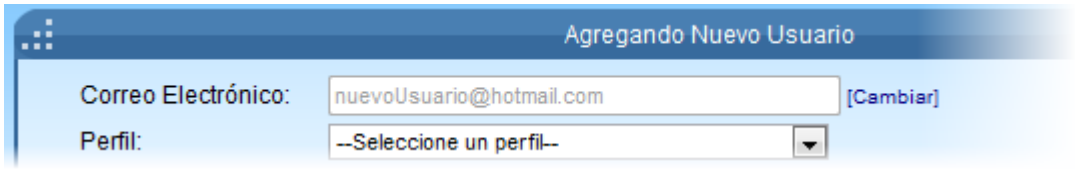

Figura 49. Nuevo Usuario, con correo electrónico correcto

El sistema valido que el correo no existe y bloqueo el campo para que no pueda ser modificado, pero si fuera necesaria alguna modificación, coloca un botón para poder cambiarlo y volver a realizar la validación.

Una vez que capturamos los datos correctos guardamos los datos y el sistema muestra lo siguiente:

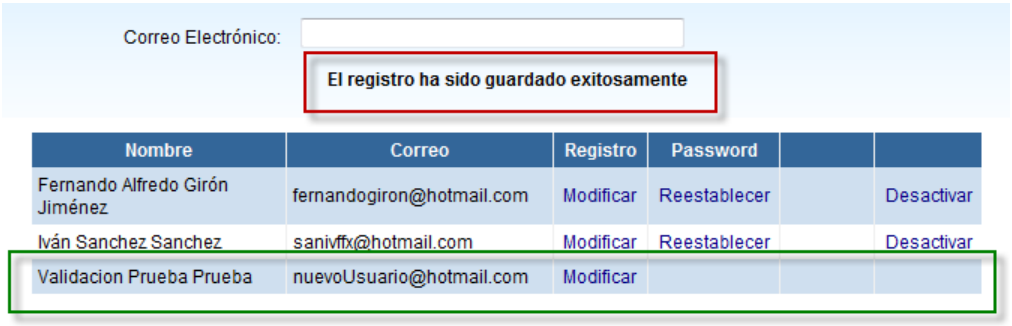

Figura 50. Nuevo Usuario registrado

Ahora realizaremos una modificación al apellido paterno del usuario, cambiará de Prueba a Modificar, y validaremos que el sistema lo permita, una vez realizado, obtenemos la siguiente pantalla:

Figura 51. Modificación de Usuarios

Aquí podemos apreciar que el apellido fue cambiado exitosamente, por lo que consideramos que la prueba fue exitosa.

Ahora que hemos dado de alta al nuevo usuario, vamos a ingresar al sistema con este correo, para validar que en el primer inicio de sesión nos permita el cambio de la contraseña.

Aquí encontramos el primer error de programación, pues ya en la aplicación, la clase que controla a la entidad Usuario, no envía correctamente los datos a la capa de los procedimientos almacenados, por lo que se requirió modificar un

elemento de esa capa y volver a realizar la prueba, de aquí se obtiene lo siguiente:

Figura 52. Primer inicio de sesión

Una vez que se logró el primer inicio y que se estableció la contraseña, realizamos la prueba para iniciar nuevamente al sistema, pero ahora con el nuevo usuario, esto queda de la siguiente forma:

Figura 53. Inicio de sesión correcto para nuevo usuario

Ahora, ya que el usuario puede ingresar a la aplicación de SOPHIA, puede realizar un cambio de contraseña cuando lo desee, presionando la leyenda [Cambiar Contraseña], realizamos la prueba, y el sistema nos arrojó la siguiente leyenda:

Cambio de contraseña por el usuario

Figura 54. Cambio de contraseña por el usuario

En este punto encontramos un error de diseño, pues el letrero se pierde con los elementos visuales de la aplicación, sin embargo, esta mejora será discutida para la siguiente versión, buscando personalizar esos elementos para los usuarios finales.

La siguiente prueba a realizar es el reinicio de contraseña, por parte del administrador del sistema, le damos restablecer y tratamos de iniciar sesión nuevamente, el resultado obtenido es el siguiente:

Figura 55. Restableciendo una contraseña olvidada

Por último, vamos a desactivar a este usuario de prueba, para que no pueda acceder a la aplicación de SOPHIA, una vez que lo hemos desactivado, intentamos entrar y el resultado es el siguiente:

Figura 56. Usuario inactivo

Con esta última prueba, damos por validado este proceso de administración y control de usuarios de SOPHIA, por lo tanto continuaremos mostrando la documentación del proceso de validación de la captura, modificación y publicación de las noticias.

Los pasos a realizar son los siguientes:

1. El usuario escribirá un artículo.
2. El usuario elegirá guardarlo como borrador o bien, iniciar el proceso de publicación.
3. El administrador del sistema verificará el artículo
4. El administrador del sistema autorizará su publicación o revocará el artículo.
5. El artículo se publicará en todos los medios disponibles.

Una vez que sabemos los puntos que debemos evaluar es necesario crear la matriz de pruebas y resolverla con la comparación del sistema, por lo que el resultado de la matriz es el siguiente:

Función	Prueba	Resultado esperado
Creación de Noticia	Escribir Noticia	Mostrar la estructura necesaria para capturar una noticia
Creación de Noticia	Guardado automático de noticias como borrador	Mientras se escribe una noticia, debe guardarse automáticamente y notificarse al usuario.
Creación de Noticia	Guardado de noticias con botón de borrador	Cuando se presiona el botón de borrador, la noticia será almacenada con ese estado.
Creación de Noticia	Continuar con el proceso de publicación	Cuando se presiona el botón de publicar, la noticia continuará con el proceso de publicación.
Publicación de Noticia	Notificación en bandeja de pendientes por publicar.	El administrador del sitio podrá leer las noticias en estado de pendientes por autorizar, revisando su bandeja de pendientes.
Publicación de Noticia	Autorización de publicación por el administrador	El administrador del sitio leerá y autorizará la publicación de la noticia.
Modificación de Noticias	El usuario podrá modificar las noticias que crea.	Si la noticia es un borrador, el usuario podrá modificarla libremente. Si la noticia ha sido publicada, al modificarse saldrá de publicación.

Una vez realizadas las pruebas correspondientes y las correcciones respectivas en la programación, el resultado de la matriz de pruebas es el siguiente:

Resultado esperado	Resultado obtenido	Comentarios
Mostrar la estructura necesaria para capturar una noticia	Al entrar a la sección de captura de noticia, se despliega correctamente el control que permite la captura de noticias y la modificación de los estilos	OK
Mientras se escribe una noticia, debe guardarse automáticamente y notificarse al usuario.	Se dejó la aplicación sin datos y al realizarse el proceso de guardado automático, omitió hacerlo, esperando nuevos datos. Se capturó una noticia de prueba, y a los 5 minutos se realizó el guardado automático, quitando por ese momento el control al usuario	OK
Cuando se presiona el botón de borrador, la noticia será almacenada con ese estado.	Se presionó el botón de guardar borrador y se realizó el proceso correctamente	OK
Cuando se presiona el botón de publicar, la noticia continuará con el proceso de publicación.	Se presionó el botón de publicar y se realizó el proceso correctamente, la noticia cambio de estado	OK
El administrador del sitio podrá leer las noticias en estado de pendientes por autorizar, revisando su bandeja de pendientes.	Ingresamos al sistema con un perfil de administrador y en la bandeja apareció la noticia para continuar con el flujo	OK

El administrador del sitio leerá y autorizará la publicación de la noticia.	Desde la bandeja de pendientes del administrador, ingresamos a la noticia y la autorizamos. Verificamos que se encontraba en todos los medios disponibles por SOPHIA. Después revocamos la noticia y desapareció de todos los medios	OK
Si la noticia es un borrador, el usuario podrá modificarla libremente. Si la noticia ha sido publicada, al modificarse saldrá de publicación.	Ingresamos una nueva noticia y la guardamos como borrador, la modificamos múltiples veces y se realizó de forma correcta. Publicamos una noticia y entramos a modificarla. El sistema la quitó de todos los medios y la puso como borrador.	OK

Esta matriz de prueba nos permite indicar que este proceso se encuentra correcto y cumple con los objetivos planteados en el documento visión.

Estos 2 procesos de validación fueron aplicados de igual forma para todo el sistema, y una vez que fueron corregidos los errores de programación y validados con sus respectivas matrices de pruebas, podemos decir que el sistema SOPHIA tiene la calidad suficiente para ser puesta en producción y quedará en el usuario final dar seguimiento a nuevas necesidades para futuras versiones.

Comparación entre situación anterior y nueva

Es difícil realizar esta comparación en términos del servicio que proveen, pues la solución será publicada una vez que se garantice que cumple con las nuevas expectativas que tiene el centro.

La versión 1.0 de SOPHIA será utilizada por el centro en una intranet, bajo 2 objetivos, el primero es que el personal aprenda a utilizar la nueva herramienta y proporcione una retroalimentación sobre nuevas necesidades y el segundo es que el personal que se encargará del mantenimiento del sistema, aprenda a manipularlo y a crear nuevos requerimientos, aprovechando el sistema que ya se tiene y el diseño de la base de datos.

Sin embargo, podemos describir que, con la implementación de este sistema se tiene un mejor control sobre las creaciones que realiza cada investigador, y permite que el propio personal interno evalué la parte multimedia y proponga mejoras sobre el trabajo que lleva actualmente el centro en la parte del diseño tridimensional.

Podemos tomar el modelo actual y el modelo propuesto y ver que en realidad si existe una retroalimentación para las mejoras del centro.

La liberación del visor tridimensional nos permite hacer evaluaciones técnicas sobre los modelos que se crean, en baja resolución, para poder mejorarlos y proporcionar información más útil a los alumnos, modelos que puedan ser manipulados rápidamente y que tengan los suficientes detalles para su estudio.

La idea de este proyecto nació de buscar una forma de llevar a la comunidad el trabajo que realiza el centro, y una de las mejoras más importantes que se llevó a cabo fue la modificación del proceso de creación de modelos tridimensionales.

En la parte de la creación de la estructura tridimensional se encontraron formas para modificar los modelos de alta definición y colocarlos en modelos con mucho menos geometría pero que simulan perfectamente a lo que se trata de llegar.

En cuanto a las texturas se incorporaron nuevas herramientas para su creación y manipulación que logran efectos más realistas una vez que se incorporan a la geometría tridimensional, permitiendo que, aunque se sacrifique la calidad de estas geometrías y la resolución de las texturas, sean muy realistas y permitan su estudio.

Para el proceso de creación y publicación de los artículos, el sistema permite que los investigadores capturen y almacenen en un servidor de información los documentos que crean y manipulan, y que éstos sean publicados, siempre con la seguridad de pasar por un proceso de validación para proteger el dominio de la Facultad de Medicina y de la UNAM.

Los elementos de retroalimentación ahora existen en el sistema y proporcionan al centro una clara visión sobre lo que sus usuarios necesitan, por eso mismo se planteó dentro del proyecto que era necesario un equipo que se dedicará al mantenimiento y creación de nuevos requerimientos.

Ahora que se ha liberado a un ambiente productivo el sistema podemos decir que nuestra visión ha cambiado y nuestra propuesta para futuras versiones de la aplicación será:

- La implementación de SOPHIA para dispositivos móviles.
- Un sistema de administración del conocimiento, para el almacenamiento y publicación de archivos.
- La creación de nuevas tablas relacionales en la base de datos, para obtener y almacenar más datos y ligarlo a otros sistemas de la Facultad de Medicina.
- La creación de sistemas de análisis de información para la toma de decisiones.
- Realizar visores de modelos tridimensionales utilizando motores de gráficos tridimensionales que permitan desplegar modelos con más detalle desde el número de polígonos hasta carga de múltiples mapas de textura con el fin de dar mayor realismo al modelo.

Estas propuestas nacen debido al análisis del modelo de negocio y de la investigación realizada para llegar al final de este proyecto, y no fueron incluidas porque recordemos que, desde un principio se plantearon los objetivos y la propuesta inicial fue mejorar el flujo de información del centro hacia su

comunidad, ahora depende del nuevo equipo de sistemas la implementación de nuevas tecnologías y el aprovechamiento de los datos almacenados.

En este capítulo queda terminado el diseño de la solución, habiendo cumplido con todo el proceso creado inicialmente, apoyándonos en 2 metodologías de diseño de software y utilizando los recursos tecnológicos que se tienen actualmente, por esto mismo ha llegado el tiempo de concluir este documento como equipo de trabajo y las experiencias obtenidas de forma personal para cada uno de los integrantes del equipo.

Capítulo V. Conclusiones

El proyecto inicio como la necesidad de crear una solución para el Centro de Adiestramiento Quirúrgico que permitiera mostrar a la comunidad de la Facultad de Medicina sus trabajos en cuanto a publicaciones y al material tridimensional que generan en apoyo de los alumnos de la Facultad de Ingeniería.

Este proyecto fue plasmado por primera vez, pensando en el desarrollo de un nuevo sitio web que permitiera generar contenidos de manera dinámica, sin embargo, comenzamos a enfrentar retos, como la necesidad de retomar lo que ya tenía la institución desarrollado.

En esta ocasión tuvimos que tocar el primero de muchos temas que para nosotros eran desconocidos, tuvimos que pensar en un concepto que permitiera tomar lo que existía en ese momento y transformarlo en una nueva solución que llenara las necesidades del CEAQ, el concepto fue la reingeniería, y de ahí, nació el título de este proyecto: "*Reingeniería del sitio web del CEAQ de la Facultad de Medicina*".

La propuesta era interesante al inicio del proyecto, sin embargo, teníamos que crear un proceso para aplicar este concepto, y durante un tiempo nos dedicamos a entender como atacar un proyecto de reingeniería y decidimos crear nuestro propio método.

Mientras analizamos el sitio web, nos dimos cuenta que, la creación del sitio web no era la solución que requería el CEAQ, sino que teníamos que entrar a entender el modelo del negocio, entender a los usuarios internos y usuarios finales y poder realizar el proceso de reingeniería directamente sobre su estructura como institución.

Durante un tiempo, comprendimos que teníamos que entender propiamente el modelo de la distribución de información y encontramos que, mucha información que creaba el centro no era conocida por su comunidad y a la vez, el centro no tenía una línea directa de retroalimentación para poder mejorar su producto llamado información.

Ya que habíamos entendido esto, logramos crear gráficamente el método que utilizaba el centro y como fluían los datos por todos los implicados y sin modificar sus procesos elementales, creamos nuevas líneas de flujo que permitían colocar todo lo creado dentro de un repositorio de datos.

Este concepto lo idealizamos en un diagrama de flujo y nos dimos cuenta que la solución realmente se encontraba en el Sistema de Información, que denominamos SOPHIA por su relación con el conocimiento.

El proyecto que estábamos a punto de desarrollar era un sistema de información que iba a ser alimentado y explotado, en su primera versión, por una aplicación web y que posteriormente, podrían aplicarse nuevas tecnologías de explotación de datos a este almacén de información.

En este momento inicio el desarrollo de un modelo por capas que nos permitiera encapsular el repositorio de datos de tal forma que fuese más segura su explotación desde la aplicación y a su vez, permita un mantenimiento más sencillo en un futuro (considerando actualizaciones de versiones o cambio de motor de base de datos).

En un principio, no se había considerado este modelo, sin embargo nos dimos cuenta de la necesidad de hacerlo de esta forma para reciclar código y aprovechar al máximo un lenguaje orientado a objetos.

Aquí fue cuando empezamos a utilizar metodologías para el desarrollo de software y optamos por experimentar 2 estrategias. La metodología RUP nos permitiría durante la etapa de análisis y diseño de la aplicación obtener la documentación suficiente para que en un futuro la aplicación quede a cargo del personal del CEAQ y su proceso iterativo incremental nos ayudará a disminuir los errores de diseño y proporcionar un producto con una calidad superior.

La primera estrategia nos generó confianza debido a que es la más utilizada en el desarrollo de los sistemas actuales sin embargo, experimentamos utilizando una metodología de desarrollo ágil denominada Programación Extrema (XP), ésta nos

permitiría adoptar únicamente las buenas prácticas en el desarrollo de la aplicación, para tener un producto de software creado a partir de la fusión de las mejores ideas de programación y mejorar los elementos de las capas en el proceso iterativo.

Ya que teníamos identificada la solución y las metodologías utilizadas comenzamos con el diseño de los elementos, siendo está la parte donde consumimos la mayor parte del proyecto, pues sabíamos que si teníamos diagramas que dieran solución a todos los objetivos planteados para el proyecto, la etapa de desarrollo sería mucho más sencilla.

Una vez que definimos los diagramas de todas las capas comenzamos con la implementación, la cual fue muy sencilla, pues solo era codificar los diagramas y realizar las buenas prácticas de XP, lo cual nos ayudó mucho a optimizar las líneas de código y utilizar todas las virtudes de la utilización del paradigma orientado a objetos, la utilización del estándar XML, el motor de base de datos elegido y los framework de desarrollo.

La utilización de las diferentes herramientas de software para el desarrollo de la aplicación y el motor tridimensional nos permitió probar los elementos antes de lanzarlos a su ambiente final de pruebas.

En un momento, tuvimos que separarnos en el desarrollo, en su primera iteración de algunos elementos (motor tridimensional y estructura general de SOPHIA), para aplicar de manera independiente nuestras experiencias en estas áreas de desarrollo, sin embargo, en la etapa de pruebas, volvimos a revisar en conjunto estos códigos y logramos optimizar algunos elementos, para reciclar líneas de código y funciones para generar dinámicamente algunos elementos web.

La implementación de la versión beta de la aplicación fue un proceso muy ágil, debido a que fuimos realizando pruebas en conjunto, durante cada etapa, por lo que la implementación se realizó en la última capa y por primera vez vio la luz, la versión beta de la aplicación de SOPHIA.

Una vez concluida la etapa de desarrollo y colocado en un ambiente de pruebas, se reiniciaron las pruebas en la base de datos, se cargaron los catálogos y se inició la etapa de pruebas. Para ello se utilizaron matrices de prueba que nos permitieron validar cada uno de los objetivos del sistema y se corrigieron algunos errores de incorporación, logrando con esto un producto de software de alta calidad.

Sin embargo, la solución propuesta requería de un servidor dedicado (por la transmisión de video y por los lenguajes utilizados de video), debido a eso, la implementación se iba a realizar en varias etapas.

Al no contar el centro con un sistema de información de éste tipo, los primeros requerimientos fueron propuestos por el equipo de desarrollo, sin embargo, una vez que se iniciaron las pruebas, surgieron nuevas necesidades y actualizaciones para la mejora de la navegabilidad, por lo que en la primera etapa, los usuarios del centro iban a iniciar con la publicación de información y elementos, mientras que su equipo de apoyo de la Facultad de Ingeniería, desarrollan las nuevas necesidades, con el fin de lanzar un producto más completo, concebido gracias a esta solución y con suficiente información, para no perder a sus primeros usuarios.

La propuesta es que, al finalizar la segunda etapa, se tenga la infraestructura de cómputo necesaria para ponerse en un ambiente productivo, tras el dominio de la Facultad de Medicina, y en una tercera etapa, se ha propuesto la utilización de tecnologías para obtener reportes ejecutivos para la toma de decisiones gracias al almacén principal de datos.

Después de los meses que nos llevó el análisis y desarrollo de la solución nos dimos cuenta que, la solución de un problema no se encuentra únicamente en un sistema, sino en el análisis del modelo de negocio y la implementación de un sistema de apoyo que permita a los usuarios, manipular la información que poseen en pro de una mejor toma de decisiones para ofrecer un mejor producto a sus clientes.

También entendimos que, las soluciones se encuentran en la realización de una buena propuesta para un sistema de información y una vez que los usuarios, tienen su primer acercamiento con un software de apoyo, ellos mismos podrán generar nuevas funcionalidades de acuerdo a como quieren que se manipule su información y la presentación que requieren de ella.

SOPHIA, el resultado de nuestro tema de tesis es un almacén de datos, creado a partir de metodologías de diseño de base de datos, que proporcionan al centro un repositorio centralizado de imágenes, videos, modelos tridimensionales e información que el centro genera todos los días y lo expone a la comunidad de la Facultad de Medicina.

Pero, SOPHIA también es un repositorio centralizado de información de retroalimentación que puede ser explotada por el centro para generar nuevos requerimientos de información y obtener informes para una mejor toma de decisiones.

SOPHIA es el núcleo bidireccional de información del CEAQ, y con su creación, paso a ser parte de su modelo de negocio, por lo cual se enfocó la mayor parte del tiempo en su diseño modular, que le permitirá crecer dinámicamente, como lo requiera la institución.

Al final de nuestro tema de tesis, la reingeniería del sitio web del CEAQ se convirtió en la reingeniería de su modelo de negocio, la creación del repositorio central de información llamado SOPHIA y la creación de una aplicación web para la manipulación de estos datos.

En conclusión, los objetivos del proyecto fueron cubiertos, aunque el análisis nos llevó a un nivel más profundo del planeado en el título de nuestra tesis, pero que cubrió el 100% de las necesidades generadas a un inicio, al final, el centro obtuvo un nuevo sitio web, reforzando sus procesos fundamentales con SOPHIA.

La experiencia obtenida por este proyecto está en haber entendido que, como profesionales en el campo de la ingeniería, las soluciones no se encuentran

únicamente en la propuesta de un sistema o de un desarrollo, las soluciones se encuentran en el núcleo del modelo de negocio, y todo en el análisis debe estar enfocado en la revisión del proceso el cual se piensa apoyar del negocio, o del área, para poder generar un producto que cubra realmente sus necesidades internas.

Cada vez que realizamos un proyecto, mejoramos de alguna forma debido a la experiencia obtenida, y SOPHIA es el resultado de muchos proyectos que realizamos durante la carrera y de otros tantos realizados profesionalmente, por eso consideramos que, la solución posee un diseño especial que permitirá a la siguiente generación de alumnos aprender de lo que creamos y mejorarlo, siempre en pro de dar al CEAQ una herramienta más compleja, que cubra sus necesidades actuales y sus futuros requerimientos, siendo una herramienta tan dinámica como el modelo de negocio al cual pertenece.

Bibliografía

1. PEÑA, Adriana. 2008. Hábitos de los usuarios de Internet en 2007.Havas Entertainment. Distrito Federal, México.

2. ROMERO, Diana. 2010. Hábitos de los usuarios de Internet en 2009. Vicepresidencia Investigación de Mercados AMIPCI. Distrito Federal, México.

3. SOMMERVILLE, Ian. 2005. Ingeniería de Software. Séptima Edición. Prentice Hall. Madrid, España.

4. GOODMAN, Danny. 2004. Javascript Bible. 5th Edition. Wiley Publishing, Inc. Indianapolis, Indiana. Estados Unidos.

5. SWEDBERG, Karl. 2010. jQuery 1.4 Reference Guide. First Edition. Packt Publishing Ltd. Olton, Reino Unido.

6. STANEK, William. 2010. R. SQL Server 2008 Administrator's Pocket Consultant. Second Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

7. BEN-GAN, Itzilk. 2009. Inside Microsoft SQL Server 2008: T-SQL Querying. First Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

8. BEN-GAN, Itzilk. 2009. Inside Microsoft SQL Server 2008: T-SQL Programming. First Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

9. SHEPHERD, George. 2005. Microsoft ASP.NET 2.0 Step by Step. Second Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

10. SHEPHERD, George. 2008. Microsoft ASP.NET 3.5 Step by Step. First Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

11. STANEK, William R. 2007. Windows Server 2008 Administrator's Pocket Consultant. First Edition. Microsoft Press. Redmon, Washington. Estados Unidos.

12. WELLS, Don. 2010. Programación Extrema. Sitio web oficial: http://www.extremeprogramming.org/

13. MENDOZA, Maria. 2004. Proceso Unificado de Rational. Sitio Web: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

14. POLLICE, Gary. 2001. RUP and XP. Sitio Web: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/apr01/RUPandXPPartIIApr01.pdf

15. HAMMER, Michael. 1994. Reingeniería. Primera Edición. Harper Collins. Londres, Inglaterra.

16. VIJOI, Daniel. 2008. Creating Next-Gen Enviroment Textures. AMC Studio. Estados Unidos.

17. AHEARN, Luke. 2006. 3D Game Textures. Focal Press. Estados Unidos.

Apéndice A. Casos de Uso

Artículo

Nombre:	Artículo
Descripción:	Permite publicar un artículo en la aplicación del sistema de información de SOPHIA.

Actores:

- Profesor
- Investigador
- Administrador del Sitio
- Alumno
- Público General

Precondiciones:

El autor debe estar registrador en el sistema.

Flujo Normal:

1. El profesor o investigador acceden al sistema con tu correo y contraseña.
2. El usuario escribe un artículo y puede guardarlo como borrador mientras lo sigue escribiendo.
3. Finalizado el artículo, el usuario podrá enviarlo al proceso de publicación.
4. El artículo se pública en el autorizador del administrador del sitio.
5. El administrador del sitio puede autorizar el artículo.
6. El artículo es publicado en el sitio web y en la fuente RSS.
7. Los demás usuarios pueden leer el contenido del artículo o mensaje.

Flujo Alternativo:

1. En el paso 1, el sistema valida que el usuario sea correcto, en caso que no, finaliza el caso de uso y lo coloca como público general.
2. En el paso 2, el sistema almacenará cada 5 minutos el avance del investigador como un borrador.

3. En el paso 5, el Administrador del sitio puede rechazar el artículo, por lo que terminará el flujo, y el autor conocerá el nuevo estado de su documento.
4. En el paso 7, si el artículo es modificado por el creador, cambia de estado a borrador y debe pasar por el proceso de publicación nuevamente.

Poscondiciones:

El articulo puede ser visualizado en el sistema web

Diagrama:

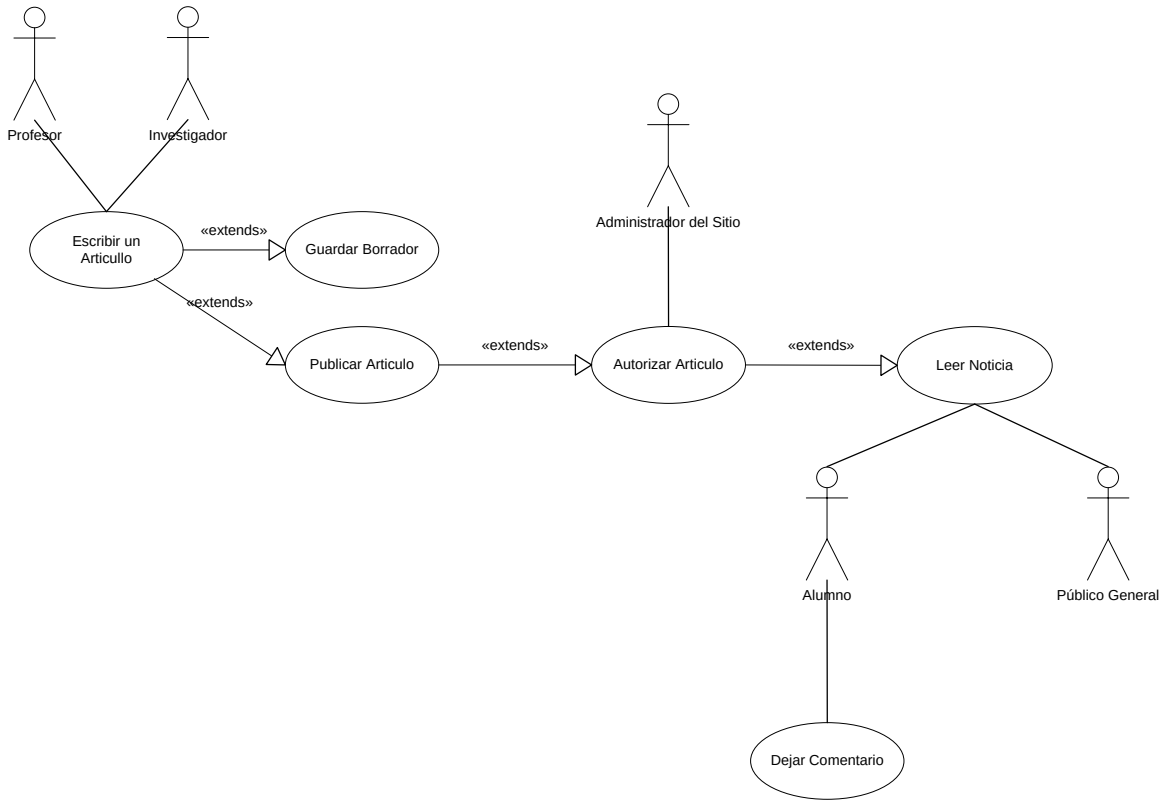

Figura 57. Articulo

Encuesta

Nombre:	Encuesta
Descripción:	Permite publicar una encuesta en la aplicación del sistema de información de SOPHIA e interactuar con ella.

Actores:

- Profesor
- Apoyo de Servicio Social de Sistemas
- Administrador del Sitio
- Alumno

Precondiciones:

El creador debe estar registrador en el sistema.

Flujo Normal:

1. El profesor, administrador del sitio o apoyo de servicio social de sistemas pueden agregar una encuesta.
2. Una vez capturada, se publica con una fecha de finalización.
3. Cuando los usuarios registrados inicien sesión en el sistema, podrán resolver las encuestas que se encuentre activas y ver el resultado parcial de los votos.
4. Concluido el tiempo, la encuesta pasa a estado de finalizada.
5. El profesor, administrador del sitio o el apoyo del servicio social de sistemas pueden ver los resultados finales de la encuesta.
6.

Flujo Alternativo:

1. En el paso 2, mientras no se publique una encuesta puede ser modificada, una vez publicada, no se podrán hacer modificaciones.
2. En el paso 3, el sistema elige aleatoriamente una encuesta de las activas en el sistema.

Poscondiciones:

Se pueden visualizar los resultados de las encuestas en el sitio web.

Diagrama:

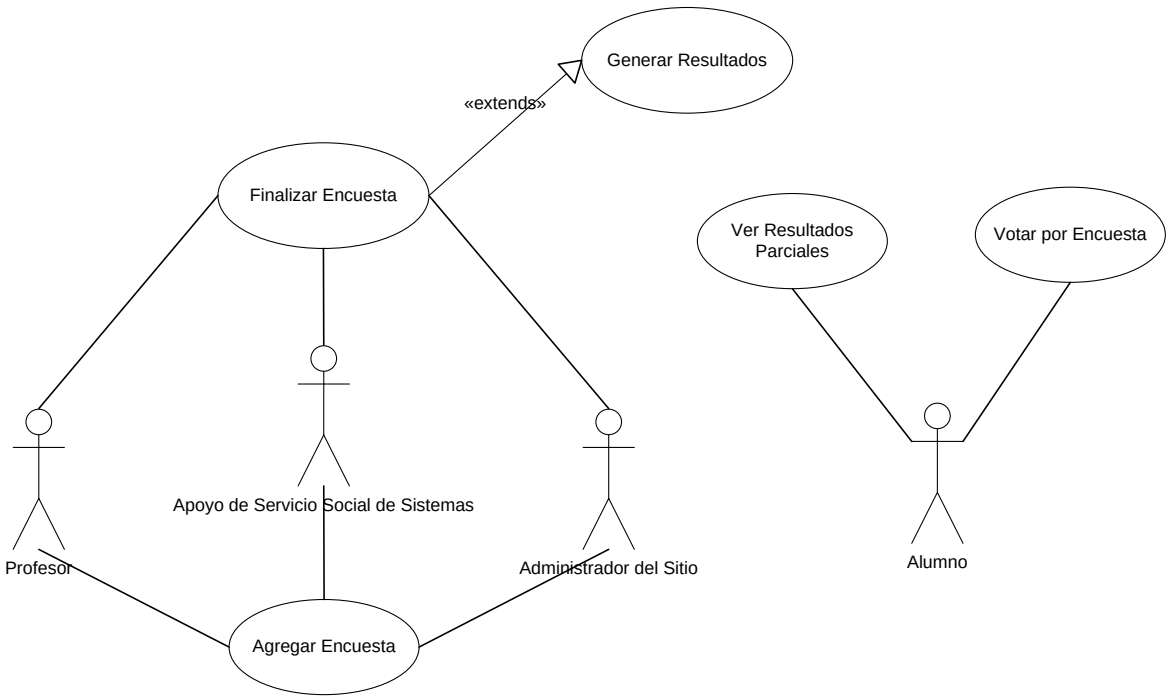

Figura 58. Encuesta

Cuestionario

Nombre:	Cuestionario
Descripción:	Permite publicar un cuestionario en la aplicación del sistema de información de SOPHIA e interactuar con ella.

Actores:

- Profesor
- Apoyo de Servicio Social de Sistemas
- Administrador del Sitio
- Alumno

Precondiciones:

El creador debe estar registrador en el sistema.

Flujo Normal:

1. El profesor, administrador del sitio o apoyo de servicio social de sistemas pueden agregar un cuestionario.
2. Una vez capturado, se publica en el sistema.
3. Cuando un alumno entra al sistema, en el área de cuestionario se le indica al alumno si existe algún cuestionario que pueda resolver.
4. El alumno resuelve el cuestionario.
5. Una vez terminado, puede ver su resultado inmediatamente.
6. El alumno puede entrar a ver su resultado, pero ya no puede volver a resolver el mismo cuestionario.
7. El profesor, administrador del sitio o apoyo de servicio social de sistemas pueden finalizar un cuestionario.
8. El sistema entrega una lista con los resultados de los cuestionarios finalizados.

Flujo Alternativo:

1. En el paso 2, se puede ir almacenando el cuestionario sin necesidad de publicarlo.
2. En el paso 4, el sistema coloca de forma aleatoria las preguntas y respuestas del cuestionario.

Poscondiciones:

Se pueden ver los resultados de los cuestionarios en el sitio web.

Diagrama:

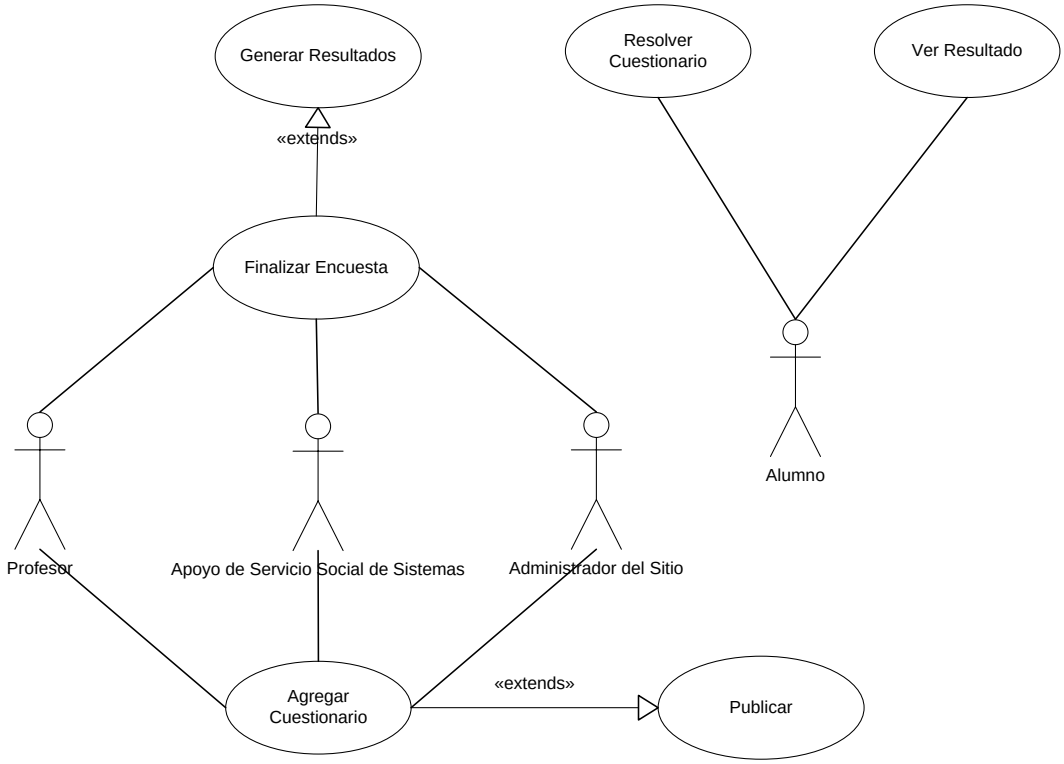

Figura 59. Cuestionario

Multimedia

Nombre:	Multimedia
Descripción:	Permite publicar contenido multimedia (imágenes, videos o modelos tridimensionales) en la aplicación del sistema de información de SOPHIA.

Actores:

- Apoyo de Servicio Social de Sistemas
- Administrador del Sitio
- Alumno
- Público General

Precondiciones:

Ninguna

Flujo Normal:

1. El Administrador del Sitio o personas de apoyo de servicio social de sistemas subirán al servidor web los nuevos archivos multimedia.
2. Estos 2 actores, modificaran las estructuras XML del sistema y los publicaran en el servidor web.
3. Los usuarios del sistema podrán ver el contenido tridimensional, videos e imágenes.

Flujo Alternativo:

Ninguno

Poscondiciones:

El artículo puede ser visualizado en el sistema web

Diagrama:

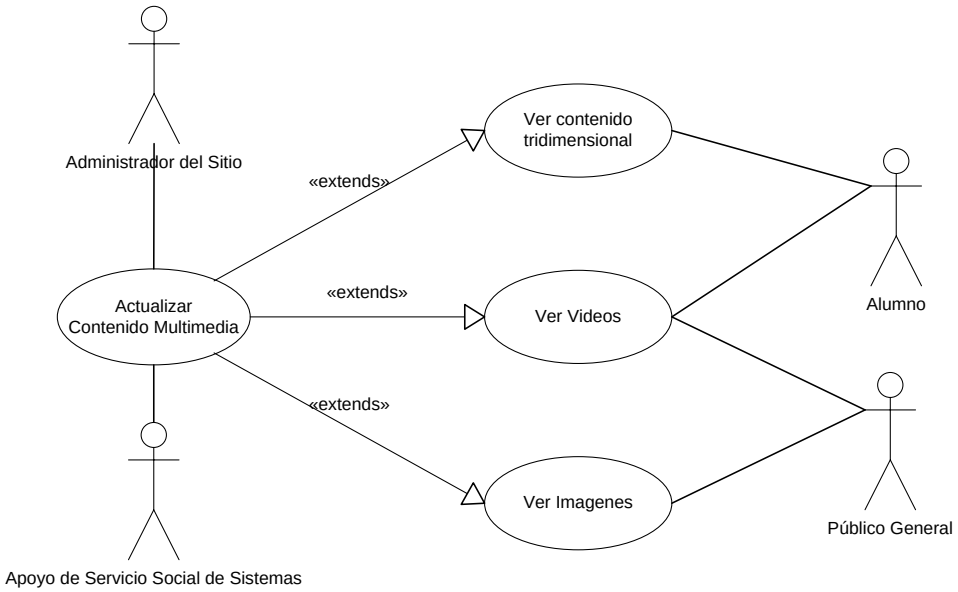

Figura 60. Multimedia

Sitios de interés

Nombre:	Sitios de Interés
Descripción:	Permite publicar y administrar sitios de interés para la comunidad en la aplicación del sistema de información de SOPHIA.

Actores:

- Apoyo de Servicio Social de Sistemas.
- Administrador del Sitio.
- Alumno.
- Investigador.
- Profesor.
- Público General.

Precondiciones:

El administrador del sitio y el personal de apoyo de servicio social deben haber iniciado su sesión en la aplicación.

Flujo Normal:

1. El administrador del sitio y el personal de apoyo de servicio social de sistemas podrán publicar, modificar o eliminar sitios de interés en la aplicación.
2. Todos los usuarios del sistema podrán ver los sitios de interés en la aplicación, ver su descripción y acceder a ellos.

Flujo Alternativo:

Ninguno

Poscondiciones:

Los sitios de interés podrán ser visualizados en la aplicación.

Diagrama:

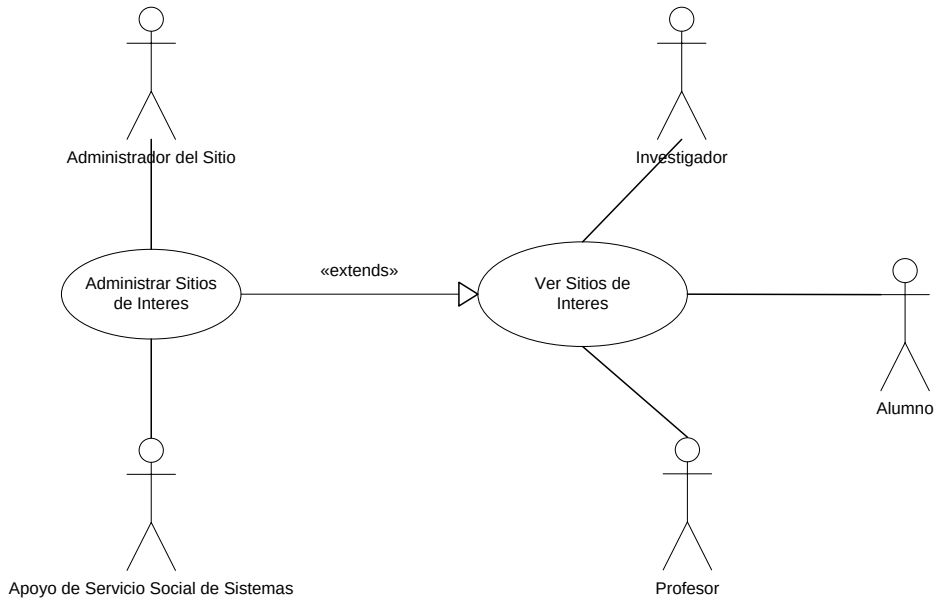

Figura 61. Sitios de interés

Control de Usuarios

Nombre:	Control de Usuarios
Descripción:	Permite administrar a los usuarios del sistema, para controlar su acceso a los distintos módulos del sistema de información de SOPHIA.

Actores:

- Administrador del Sitio.
- Apoyo de Servicio Social de Sistemas.
- Alumno.
- Investigador.
- Profesor.

Precondiciones:

Presentar una solicitud de acceso al sistema, con correo electrónico y datos generales.

Flujo Normal:

1. El centro valida la solicitud y pasa la petición al administrador del sitio web.
2. El administrador da de alta los datos del nuevo usuario.
3. El administrador notifica al nuevo usuario que ha sido dado de alta.
4. El nuevo usuario ingresará al portal con su correo electrónico como contraseña.
5. El sistema pedirá al nuevo usuario que coloque su contraseña y la confirme, y también pedirá la fecha de nacimiento del usuario.
6. Una vez registrada la contraseña, el sistema pedirá que vuelva a iniciar sesión y mostrará el menú de acuerdo a su perfil.

Flujo Alternativo:

1. Para el paso 2, el sistema validará que el correo electrónico aún no está registrado, en caso que lo esté no podrá llevarse a cabo el registro.
2. Para el paso 4, el sistema validará que exista y sea un nuevo usuario, en caso de que no, y la contraseña no sea correcta, termina el flujo.
3. Para el paso 5, el sistema validará la fecha de nacimiento, si no es correcta, termina el flujo.
4. Para el paso 6, el sistema validará los datos, si no es correcta, termina el flujo e indica el error.

Poscondiciones:

EL usuario podrá operar con el sistema de acuerdo a su perfil.

Diagrama:

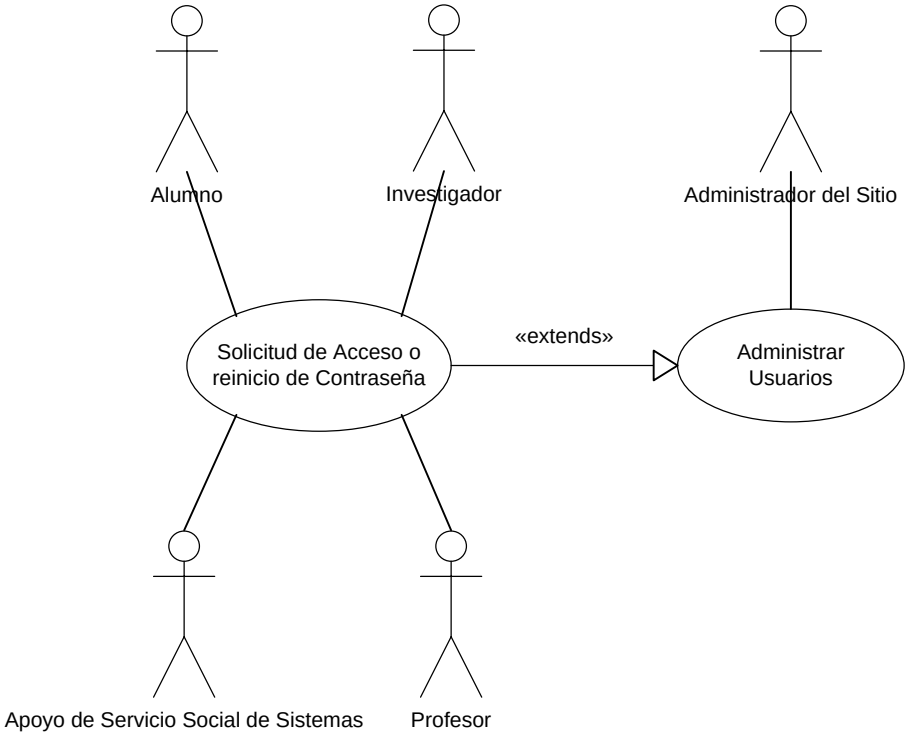

Figura 62. Control de Usuarios

Apéndice B. Índice de Crecimiento de la Base de datos

Para una base de datos SQL Server 2008, el proveedor pone a nuestra disposición una serie de reglas y fórmulas que nos permiten conocer el estado actual de la base de datos[42] y esto nos permite predecir el crecimiento a lo largo del tiempo.

Los pasos, tal y como lo indica el proveedor son los siguientes:

1. Especifique el número de filas que habrá en la tabla:

   Num_Renglones = número de filas en la tabla.

2. Especifique el número de columnas de longitud fija y de longitud variable, y calcule el espacio necesario para su almacenamiento. Calcule el espacio que ocupa cada uno de estos grupos de columnas en la fila de datos. El tamaño de una columna depende del tipo y la longitud especificados para los datos.

   Num_Cols = número total de columnas (de longitud fija y variable).
   Tamaño_Fijo_Datos = tamaño total en bytes de todas las columnas de longitud fija.

3. Una parte de la fila, conocida como el mapa de bits NULL, se reserva para administrar la nulabilidad en las columnas. Calcule el tamaño:

   Bitmap_Null = 2 + ((Num_Cols + 7) / 8)

4. Sólo debe utilizarse la parte entera de la expresión anterior. Descarte el resto.

5. Calcule el tamaño de los datos de longitud variable. Que en el caso de SOPHIA, no aplica, pues no se utilizaron estructuras variables.

6. Calcule el tamaño total de la fila:

   Tamaño_renglon = Tamaño_Fijo_Datos + Bitmap_null + 4

7. Calcule el número de filas por página (8096 bytes libres por página):

[42] Microsoft MSDN. Libros de Pantalla de SQL Server 2008 R2. Noviembre de 2009

Renglones_por_pagina = 8096 / (Tamaño_renglon + 2)

8. Calcule el número de páginas necesarias para almacenar todas las filas:

   Num_Paginas = Num_Renglones / Renglones_por_pagina

   Nota: El número de páginas estimado debe redondearse hacia arriba a la página completa más cercana.

9. Calcule el espacio necesario para almacenar los datos en el montón (8192 bytes en total por página):

   Tamaño de almacenamiento en bytes = 8192 x Num_Paginas

Para el cálculo del crecimiento de la base de datos se desarrollaron las fórmulas mencionadas en una hoja de cálculo y se consideró lo siguiente:

- Los catálogos no cambian en el tiempo.
- Cada cuestionario tiene un promedio de 10 preguntas.
- Cada pregunta tiene 4 respuestas.
- Considerando que el 50% de los usuarios de SOPHIA serán alumnos, será el número de cuestionarios respondidos.
- Cada Encuesta tiene un promedio de 8 opciones

La carga inicial tuvo como resultado la siguiente tabla:

Tabla	Registros	Columnas	Tamaño Datos	Bitmap Null	Tamaño Renglón	Renglón por Página	Número Páginas	Tamaño Tablas (bytes)
SphaCatStatusCuest	3	2	26	3	33	231.31	0.01	106
SphaCatStatusNot	5	2	21	3	28	269.87	0.02	152
SphaCatStatus	3	2	21	3	28	269.87	0.01	91
SphaCatPerfil	5	2	21	3	28	269.87	0.02	152
SphaCuestionario	1	4	64	3	71	110.90	0.01	74
SphaNoticia	1	7	64072	4	64080	0.13	7.92	64842
SphaPregunta	10	4	114	3	121	65.82	0.15	1245
SphaRespuesta	40	3	110	3	117	68.03	0.59	4816
SphaControlRespuesta	10	4	16	3	23	323.84	0.03	253
SphaSitiosInteres	1	5	64108	4	64116	0.13	7.92	64878
SphaUsuario	1	11	189	4	197	40.68	0.02	201
SphaEncuesta	1	4	64	3	71	110.90	0.01	74
SphaOpcionEncuesta	5	4	39	3	46	168.67	0.03	243
SphaComentario	1	4	64012	3	64019	0.13	7.91	64780

A los 5 años se espera una carga de:

Tabla	Registros	Columnas	Tamaño Datos	Bitmap Null	Tamaño Renglón	Renglón por Página	Número Páginas	Tamaño Tablas (bytes)
SphaCatStatusCuest	3	2	26	3	33	231.31	0.01	106
SphaCatStatusNot	5	2	21	3	28	269.87	0.02	152
SphaCatStatus	3	2	21	3	28	269.87	0.01	91
SphaCatPerfil	5	2	21	3	28	269.87	0.02	152
SphaCuestionario	60	4	64	3	71	110.90	0.54	4432
SphaNoticia	750	7	64072	4	64080	0.13	5,936.45	48631399
SphaPregunta	600	4	114	3	121	65.82	9.12	74675
SphaRespuesta	2400	3	110	3	117	68.03	35.28	288987
SphaControlRespuesta	75000	4	16	3	23	323.84	231.60	1897233
SphaSitiosInteres	25	5	64108	4	64116	0.13	197.99	1621957
SphaUsuario	250	11	189	4	197	40.68	6.15	50340
SphaEncuesta	60	4	64	3	71	110.90	0.54	4432
SphaOpcionEncuesta	300	4	39	3	46	168.67	1.78	14571
SphaComentario	1000	4	64012	3	64019	0.13	7,907.73	64780142

Se desarrolló el escenario del crecimiento por año, para obtener una gráfica donde se pudiera ver la tendencia de crecimiento. Debemos mencionar que el estimado está basado en la versión actual, sin embargo el sistema deberá crecer de una forma más dinámica mientras sigan existiendo nuevos requerimientos.

Año	Tamaño bytes	Tamaño en megabyte
Inicial	201,907.00	0.19
Primer Año	16,754,499.00	15.98
Segundo Año	43,250,704.00	41.25
Tercer Año	67,048,363.00	63.94
Cuarto Año	93,591,479.00	89.26
Quinto Año	117,368,669.00	111.93

A continuación poder visualizar gráficamente la tendencia de crecimiento de la base de datos:

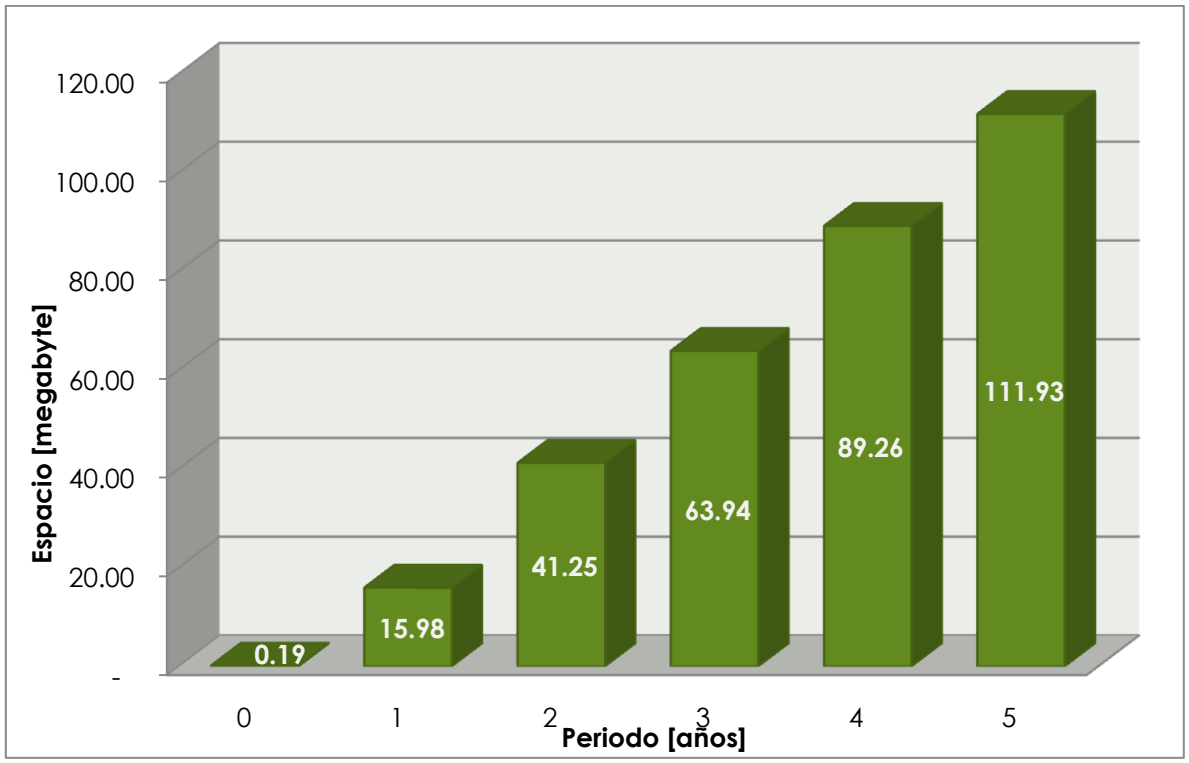

Figura 63. Gráfica de tendencia de crecimiento de la base de datos de SOPHIA

Apéndice C. Glosario

- **UNAM**. Universidad Nacional Autónoma de México.
- **CEAQ**. Centro de Enseñanza y Adiestramiento Quirúrgico
- **SOPHIA**. Sistema de administración de artículos y multimedia.
- **RUP**. Proceso unificado de Rational. Metodología de desarrollo.
- **XP**. Programación Extrema. Metodología de desarrollo.
- **RSS**. Really Simple Syndication. Lenguaje para la publicación de canales de información.
- **UML**. Lenguaje Unificado de Modelado.
- **Stakeholders**. Personas a quienes afecta la implementación o los cambios en un sistema.
- **RDF**. Marco de Descripción de Recursos. Lenguaje para la generación de canales de noticia.
- **Atom**. Formato de difusión de contenido. Alternativa a RSS.
- **NURBS.** Modelo matemático para generar y representar curvas y superficies.
- **Streaming**. Distribución de audio y video por internet, se refiere a corriente continua de información sin interrupción.
- **Buffer**. Ubicación de la memoria reservada para el almacenamiento temporal de información en espera de ser procesada.
- **Codecs**. Especificación desarrollada en software que transforma un flujo de datos o señal en elementos que pueden ser entendidos.
- **UDP**. Protocolo de comunicación en la capa de transporte que envía información sin establecer previamente una conexión.
- **TCP**. Protocolo de comunicación de la capa de transporte más utilizado en las redes actuales.
- **HD**. Alta Definición. Formatos de video superiores a los 1280 x 720 pixeles.
- **AJAX**. Acrónimo de Javascript asíncrono y XML. Es una técnica de desarrollo web para la creación de aplicaciones interactivas o RIA.
- **RIA**. Aplicaciones Enriquecidas de Internet

- **XHTML**. Lenguaje extensible de marcado de hipertexto. Es el lenguaje pensado en sustituir a HTML.
- **CSS**. Hojas de Estilo en Cascada.
- **DOM.** Modelo de objetos de documento. Permite a los programas acceder al contenido, la estructura y el estilo de los documentos HTML y XML.
- **XML**. Lenguaje de marcas extensible. Es un lenguaje que nos permite definir diferentes lenguajes, adaptables a las necesidades particulares de los usuarios.
- **XSLT**. Lenguaje de desarrollo que nos permite transformar documentos XML en otra especificación. Es muy utilizado para generar código XHTML a partir de consultas en XML.
- **JSON**. Es un formato ligero que permite la creación de objetos a partir de cadenas, es muy utilizado con AJAX cuando el servidor no retorna un XML.
- **XMLHttpRequest**. Es una interfaz empleada para realizar peticiones HTTP o HTTPS a servidores.
- **CLR**. Lenguaje común en tiempo de ejecución. Componente de la máquina virtual de .NET que transforma el código CIL en código nativo para el sistema operativo.
- **CIL**. Lenguaje de programación resultado de la compilación de los lenguajes de .NET. Código intermedio.
- **OLAP**. Procesamiento analítico en línea. Tecnología utilizada en el campo de la inteligencia empresarial cuyo objetivo es agilizar grandes cantidades de datos.
- **HNF**. La más alta forma normal.
- **FNBC**. Forma Normal de Boyce-Codd, es una forma normal más fuerte a la Tercer Forma normal.
- **#FN**. Forma Normal. Aplicación de reglas a las relaciones obtenidas tras el paso del modelo de CHEN al modelo relacional.

Apéndice D. Comportamiento de Software