UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA WEB DE GENERACIÓN DE HERRAMIENTAS PARA EL APRENDIZAJE DEL IDIOMA INGLÉS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

CÉSAR GARZÓN TRINIDAD

DIR. ING. JUAN CARREÓN GRANADOS

CIUDAD UNIVERSITARIA 2006

A mis padres, gracias por su continuo amor y apoyo.

Agradecimientos

A la Universidad Nacional Autónoma de México por la formación académica que me ha brindado.

A mi director de tesis el Ing. Juan Carreón Granados.

A mis amigos Jorge, Juan Carlos, Saúl y Gerardo.

Y a todas aquellas personas que han contribuido a mi formación profesional.

Contenido

Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés

El presente trabajo de tesis analiza la combinación de la enseñanza presencial junto con las tecnologías de la información y la comunicación (TIC). Aprovechando el contexto tecnológico actual propone el manejo de las herramientas tradicionales de la educación y las nuevas herramientas tecnológicas como una sola, que se complementan utilizando la Web como medio para facilitar el acceso a dichos instrumentos especializados que contribuyan tanto al aprendizaje como al reforzamiento del mismo.

Para tal fin, se propone un sistema Web el cual sea una herramienta complementaria a un curso presencial del idioma Inglés. El dominio de este idioma es un punto estratégico para poder competir en un mundo globalizado como el que vivimos hoy día, por lo cual resulta el vehículo fundamental para la comunicación internacional en todas las esferas.

De tal forma el objetivo de este trabajo es proporcionar el medio idóneo para un modelo de formación flexible al que se le pueda dar seguimiento “en cualquier lugar”, “a cualquier hora”, “para cualquiera” y a un ritmo personalizado, basado en el concepto de un aprendizaje mixto entre la educación presencial y el uso de las TIC.

1. Las computadoras en la educación hoy.

El desarrollo de nuevas tecnologías junto con las redes de comunicaciones digitales como Internet han proporcionado la infraestructura necesaria para la creación de nuevos escenarios educativos basados en la computadora.

1 Las Computadoras en la Educación.

El desarrollo de nuevas tecnologías junto con las redes de comunicaciones digitales como Internet han proporcionado la infraestructura necesaria para la creación de nuevos escenarios educativos basados en la computadora. Al extenderse de forma espectacular en los últimos años el acceso a los mismos, sobre todo en el ámbito universitario y profesional, se ha despertado un renovado interés por ofrecer una oferta de educación basada en el uso de las tecnologías actuales.

1.1 Teorías del Aprendizaje e Integración de la Tecnología en la Educación

El aprendizaje y las teorías que tratan los procesos de adquisición de conocimiento han tenido durante este último siglo un enorme desarrollo debido fundamentalmente a los avances de la psicología y de las teorías instruccionales, que han tratado de sistematizar los mecanismos asociados a los procesos mentales que hacen posible el aprendizaje. El propósito de las teorías educativas es el de comprender e identificar estos procesos y a partir de ellos, tratar de describir métodos para que la instrucción sea más efectiva.

Es en este último aspecto en el que principalmente se basa el diseño instruccional, que se fundamenta en identificar cuáles son los métodos que deben ser utilizados en el diseño del proceso de instrucción, y también en determinar en qué situaciones estos métodos deben ser usados.

De acuerdo con Reigeluth[1] de la combinación de estos elementos (métodos y situaciones) se determinan los principios y las teorías del aprendizaje. Un principio de aprendizaje describe el efecto de un único componente estratégico en el aprendizaje de forma que determina el resultado de dicho componente sobre el alumno bajo unas determinadas condiciones.

Desde el punto de vista prescriptivo, un principio determina cuándo debe este componente ser utilizado. Por otro lado, una teoría describe los efectos de un modelo completo de instrucción, entendido como un conjunto integrado de componentes estratégicos en lugar de los efectos de un componente estratégico aislado.

En un primer lugar, desde un punto de vista psicológico y pedagógico, se trata de identificar qué elementos de conocimiento intervienen en la enseñanza y cuáles

[1] Charles M. Reigeluth
Elaboration theory(ET)

son las condiciones bajo las que es posible el aprendizaje. Por otro lado, en el campo de la tecnología instruccional, se trata de sistematizar este proceso de aprendizaje mediante la identificación de los mecanismos y de los procesos mentales que intervienen en el mismo. Ambos campos van a servir de marco de referencia para el desarrollo de los sistemas de enseñanza basados en computadora.

1.2 Teorías de Aprendizaje

Las teorías de aprendizaje desde el punto de vista psicológico han estado asociadas a la realización del método pedagógico en la educación. El escenario en el que se lleva a cabo el proceso educativo determina los métodos y los estímulos con los que se lleva a cabo el aprendizaje. Desde un punto de vista histórico, a grandes rasgos son tres las tendencias educativas que han tenido vigencia a lo largo de la educación: La educación social, la educación liberal y la educación progresista.

- Educación Social

  En la educación social nos encontramos en una etapa anterior a la existencia de instituciones educativas. En este contexto la educación se puede considerar que es exclusivamente oral y responsabilidad de la familia y de la sociedad que la guarda y la transmite. En esta situación, el proceso de aprendizaje se lleva a cabo en el contexto social y como parte de la integración del individuo en el grupo, proceso éste que se realiza día a día a lo largo de su vida.

- Educación Liberal

El modelo clásico de educación se puede considerar el modelo liberal, basado en La República de Platón, donde ésta se plantea como un proceso disciplinado y exigente. El proceso de aprendizaje se basa en el seguimiento de un currículum estricto donde las materias se presentan en forma de una secuencia lógica que haga más coherente el aprendizaje.

- Educación Progresista

En contraposición a este se puede definir el modelo “progresista”, que trata de ayudar al alumno en su proceso educativo de forma que éste sea percibido como un proceso “natural”. Estas teorías tienen origen en el desarrollo de las ideas sociales de Rousseau y que han tenido un gran desarrollo en la segunda mitad del siglo de la mano de John Dewey en EE.UU. y de Jean Piaget en Europa.

Estas tres corrientes pedagógicas se han apoyado generalmente en varias teorías educativas y modelos cognitivos de la mente para la elaboración de las estrategias de aprendizaje. En muchos aspectos, el desarrollo de estas teorías y de otras derivadas de ellas está influido por el contexto tecnológico en el que se aplican, pero fundamentalmente tienen como consecuencia el desarrollo de elementos de diseño instruccional, como parte de un proceso modelador del aprendizaje, para lo cual se trata de investigar tanto los mecanismos mentales que intervienen en el aprendizaje como los que describen el conocimiento. Desde este punto de vista más orientado a la psicología se pueden distinguir principalmente dos enfoques: el enfoque conductista y el enfoque cognitivista.

1.2.1 El enfoque conductista

Conductivismo. Se denomina así a la teoría del aprendizaje animal y humano que se centra solo en conductas objetivas observables, descartando las actividades mentales que ocurren por estos procesos. Para el conductismo, el modelo de la mente se comporta como una "caja negra" donde el conocimiento se percibe a través de la conducta, como manifestación externa de los procesos mentales internos, aunque éstos últimos se manifiestan desconocidos.

En el siguiente cuadro se observan las técnicas para la adquisición, mantenimiento, retención de habilidades y conocimientos:

Reforzamiento	Consiste en presentar un estímulo reforzador, de manera seguida a una respuesta. El reforzador es el estímulo que aumenta la probabilidad de ocurrencia de una respuesta.
Moldeamiento por Aproximaciones sucesivas	Primero se identifica la tarea meta o terminal. Se inicia con el primer eslabón proporcionando reforzadores ante la emisión de respuestas adecuadas, una vez dada la respuesta correcta al primer eslabón se continúa con el siguiente, actuando de la misma forma hasta llegar a la respuesta terminal.
Generalización y discriminación	Ocurre cuando una persona, ante estímulos similares más no idénticos, emite una misma respuesta o bien, cuando ante un mismo estímulo se emiten respuestas similares. En la discriminación se responde de manera diferencial ante los estímulos.
Modelado	Consiste en modelar (exhibir) la conducta que se desea que alguien aprenda haciendo evidente la consecuencia que sigue a la conducta exhibida.

Tabla 1.1 Técnicas de adquisición, mantenimiento, retención de habilidades y conocimientos según el conductivismo.

Las técnicas para la eliminación de conductas son:

Extinción	Consiste en el retiro del reforzador que mantiene una conducta.
Castigo	Es un procedimiento por medio del cual se proporciona un estímulo negativo, adverso, después de la emisión de una respuesta.
Reforzamiento diferencial	Consiste en la selección de una conducta incompatible con la conducta que se desea eliminar.
Tiempo fuera	Esta técnica, consiste en suspender o retirar al sujeto por un tiempo "x" de la situación en la cual manifiesta conductas indeseables.

Tabla 1.2 Técnicas Conductivistas para la eliminación de conductas.

El conductismo aplicado a la educación es una tradición dentro de la psicología educativa un ejemplo son los conceptos substanciales del proceso instruccional. Cualquier conducta académica puede ser enseñada de manera oportuna, si se tiene una programación instruccional eficaz basada en el análisis detallado de las respuestas de los alumnos. Otra característica de este enfoque es el supuesto de que la enseñanza consiste en proporcionar contenidos o información al alumno el cual tendrá que adquirir básicamente en el arreglo adecuado de las contingencias de reforzamiento. De acuerdo con este enfoque, la participación del alumno en los procesos de enseñanza–aprendizaje está condicionada por las características prefijadas del programa por donde tiene que transitar para aprender, es decir es un sujeto cuyo desempeño y aprendizaje escolar pueden ser arreglados desde el exterior (la situación instruccional, los métodos, los contenidos), siempre y cuando se realicen los ajustes ambientales y curriculares necesarios. En esta perspectiva el trabajo de los maestros consiste en diseñar una adecuada serie de arreglos contingenciales de reforzamiento para enseñar. De acuerdo con esta aproximación el maestro debe verse como un ingeniero educacional y un administrador de contingencia. Un maestro eficaz debe ser capaz de manejar hábilmente, los recursos tecnológicos conductuales de este enfoque (principios, procedimientos, programas conductuales) para lograr con éxito niveles de eficiencia en su enseñanza y sobre todo en el aprendizaje de sus alumnos.

Algunos ejemplos de la aplicación de este enfoque son:

- La enseñanza programada: Durante principios de los setenta se desarrollo una gran cantidad de experiencias y aplicaciones de programas de enseñanza diseñados desde esta aproximación. En un inicio las protagonistas fueron las máquinas de enseñanza y posteriormente los textos programados.

Las características de dicha metodología son las siguientes:

a) Definición explícita de los objetivos del programa.
b) Presentación secuencial de la información según la lógica de dificultad creciente.
c) Participación del estudiante.
d) Reforzamiento inmediato de la información.
e) Individualización (avance de cada estudiante a su propio ritmo)
f) Registro de resultados y evaluación continua.

Los programas EAC (Programas de enseñanza asistida por computadora): constituyen software y courseware educativo con los mismos rasgos que la enseñanza programada (situaciones instruccionales demasiado estructurada y que dejan poca participación significativa al alumno) pero con las ventajas de la interactividad que proporciona la computadora.

Programas de enseñanza asistida por computadora	
Ventajas	**Desventajas**
Facilidad de uso; no se requieren conocimientos previos.	Alumno pasivo.
Existe interacción.	No es posible la participación del educador para el planteamiento de dudas, etc.
La secuencia de aprendizaje puede ser programada de acuerdo a las necesidades del alumno.	Excesiva rigidez en la secuencia de los contenidos, que impide el tratamiento de respuestas no previstas.
Retroalimentación de inmediato sobre cada respuesta.	No se sabe por qué un reactivo es correcto o Incorrecto.
Favorecen automatización de habilidades básicas para aprendizajes más complejos.	Fragmentación de contenidos excesivamente uniforme y reductora, sea cual sea la materia.
Proporciona enseñanza individualizada.	Individualización muy elemental; no tiene en cuenta el ritmo, no guía.

Tabla 1.3 Ventajas y desventajas de programas de enseñanza asistida por computadora.

Sin embargo la EAC ha continuado desarrollándose solventando algunos de los inconvenientes descritos. Pese a las muchas críticas recibidas, muchos programas actuales se basan en los las propuestas conductistas: "descomposición de la información en unidades, diseño de actividades que requieren una respuesta y planificación del refuerzo".

Desde el punto de vista de la aplicación de estas teorías en el diseño instruccional, fueron los trabajos desarrollados por B. F Skinner para la búsqueda de medidas de efectividad en la enseñanza el que primero lideró el movimiento de los objetivos conductistas.

De esta forma, el aprendizaje basado en este paradigma sugiere medir la efectividad en términos de resultados, es decir, del comportamiento final, por lo que ésta está condicionada por el estímulo inmediato ante un resultado del

alumno, con objeto de proporcionar una realimentación o refuerzo a cada una de las acciones del mismo.

Las críticas al conductismo están basadas en el hecho de que determinados tipos de aprendizaje solo proporcionan una descripción cuantitativa de la conducta y no permiten conocer el estado interno en el que se encuentra el individuo ni los procesos mentales que podrían facilitar o mejorar el aprendizaje.

1.2.2 El enfoque cognitivista.

La corriente cognoscitiva pone énfasis en el estudio de los procesos internos que conducen al aprendizaje, se interesa por los fenómenos y procesos internos que ocurren en el individuo cuando aprende, cómo ingresa la información a aprender, cómo se transforma en el individuo y cómo la información se encuentra lista para hacerse manifiesta, así mismo considera al aprendizaje como un proceso en el cual cambian las estructuras cognoscitivas (organización de esquemas, conocimientos y experiencias que posee un individuo), debido a su interacción con los factores del medio ambiente.

Las teorías cognitivas tienen su principal exponente en el constructivismo. El constructivismo es una teoría de aprendizaje. Uno de sus presupuestos básicos es que cuanto sabemos y creemos es fruto del lenguaje con que comprendemos y transmitimos nuestras percepciones y que, sobre una misma realidad, pueden darse diferentes puntos de vista, todos ellos igualmente válidos. Al hablar, vamos creando la realidad junto con nuestros interlocutores. Así es como creamos y modificamos nuestra identidad, que retocamos permanentemente en virtud del contexto, de las circunstancias de nuestra interacción, de las características y expectativas de nuestro interlocutor. El constructivismo ha tenido una gran influencia en la educación científica y matemática, pero mucho menos lo ha sido en la educación de las Ciencias de la Computación. El constructivismo en realidad cubre un espectro amplio de teorías acerca de la cognición que se fundamentan en que el conocimiento existe en la mente como representación interna de una realidad externa. El aprendizaje en el constructivismo tiene una dimensión individual, ya que al residir el conocimiento en la propia mente, el aprendizaje es visto como un proceso de construcción individual interna de dicho conocimiento.[2]
Jean Piaget, biólogo de formación con una especial preferencia por problemas de corte filosófico y principalmente sobre los referidos al tópico del conocimiento, considera que las estructuras del pensamiento se construyen, pues nada está dado al comienzo. Las estructuras se construyen por interacción entre las actividades del sujeto y las reacciones del objeto. Más bien recaen en las acciones mismas que el sujeto ha realizado sobre los objetos, y consiste en abstraer de

2 **Duffy and Jonassen, 1992**

Duffy, T. and Jonassen, D. (1992).

Constructivism and the Technology of Instruction.

Laurence Erlbaum Assocviates, Hillsdale, Ney Jersey.

esas acciones, por medio de un juego de "asimilaciones" y "acomodaciones", los elementos necesarios para su integración en estructuras nuevas y cada vez más complejas.

Otra de las teorías educativas cognitivistas es el conexionismo. Conjunto de teorías sobre la estructura de enlaces caóticos del cerebro y su modelado usando redes neuronales artificiales. El conexionismo involucra un esquema masivamente paralelo en el cual el aspecto sobresaliente es la estructura de enlaces con sus pesos (niveles de flujo eléctrico admisible por el enlace). Ambos son críticos. El conexionismo es fruto de la investigación en inteligencia artificial, neurología e informática para la creación de un modelo de los procesos neuronales. Para las teorías conexionistas la mente es una máquina natural con una estructura de red donde el conocimiento reside en forma de patrones y relaciones entre neuronas y que se construye mediante la experiencia.

En el conexionismo, el conocimiento externo y la representación mental interna no guardan relación directa, es decir, la red no modela o refleja la realidad externa porque la representación no es simbólica sino basada en un determinado reforzamiento de las conexiones debido a la experiencia en una determinada situación.

Por último, otra teoría derivada del cognitivismo y también en parte proveniente de las ciencias sociales es el postmodernismo. Para el postmodernismo, el pensamiento es una actividad interpretativa, por lo que más que la cuestión de crear una representación interna de la realidad o de representar el mundo externo lo que se postula es cómo se interpretan las interacciones con el mundo de forma que tengan significado. En este sentido la cognición es vista como una internalización de una interacción de dimensión social, en donde el individuo está sometido e inmerso en determinadas situaciones. De esta forma, para estos dos enfoques cognitivos, el postmoderno y el conexionista, la realidad no es modelable, sino interpretada. Tanto una teoría como la otra son no representacionales y ambos sugieren métodos instruccionales basados en las situaciones sociales o cooperativas.

Es en esta línea social donde los conexionistas y en mayor medida el postmodernismo se han alineado con el movimiento de la cognición situada que compromete el proceso de aprendizaje a la observancia del entorno cultural en el que se realiza, influido por el contexto social y material. Por último, podemos decir que la diferencia fundamental entre ambos enfoques está en su actitud ante la naturaleza de la inteligencia.

En tanto que el conexionismo presupone que sí es posible la creación artificial de inteligencia mediante la construcción de una red neuronal que sea inteligente, el postmodernismo argumenta que una computadora es incapaz de capturar la inteligencia humana.[3] Muchas de estas consideraciones han tenido importantes

3 **Winograd and Flores, 1986**

consecuencias en el desarrollo de paradigmas educativos basados en la enseñanza por computadora.

1.3 Breve Historia de las Computadoras en la Educación.

El origen de la instrucción automática, entendida como un proceso que no necesita de la intervención de un profesor, tiene sus raíces antes incluso de la aparición de las primeras computadoras hacia mediados de los años 40.

Ya en 1912, E. L. Thorndike apuntaba la idea de un material auto-guiado o de una enseñanza programada de forma automática, en lo que puede considerarse una visión precursora de lo que más tarde se entendió como instrucción asistida.

Posteriormente, no es hasta los años 50, cuando surge la enseñanza asistida por computadora, entendida como la aplicación de la tecnología informática para proporcionar enseñanza, y como la solución tecnológica al proceso de instrucción individualizada. En general, es comúnmente aceptado que el nacimiento de la disciplina de la "instrucción asistida por computadora" y de los primeros fundamentos instruccionales de la misma se realiza hacia mediados de los años 50 de la mano de las teorías conductistas de B. F. Skinner con la publicación del artículo "The Science of Learning and the Art of Teaching", quien primero apunta las deficiencias de las técnicas de instrucción tradicionales y estableciendo que éstas podían mejorarse con el uso de lo que entonces se denominaban teaching machines. El paradigma en el que se inspira para el desarrollo de la tecnología aplicada a la enseñanza es el que entonces se denomina "instrucción programada", de la que fue pionero el psicólogo norteamericano S. J. Pressey, y que se asienta sobre la base de que el material instruccional debe estar compuesto por una serie de pequeños "pasos", cada uno de los cuales precisa de la respuesta activa del estudiante, quien recibe una realimentación instantánea en el uso de los mismos.

Según estos principios de diseño, el estudiante debe conservar en todo momento capacidad para proceder de forma libre en el material y conservando lo que se definen como tres principios fundamentales de la instrucción programada:

1. El desarrollo del auto-estímulo en el uso de los sistemas.
2. La participación activa del estudiante
3. La realimentación durante el uso de los sistemas

En 1963, Patrick Suppes y Reichard Atkinson en la universidad de Stanford realizaron algunas investigaciones y desarrollos sobre la enseñanza asistida por

Winograd, T. and Flores, F. (1986).
Understanding computers and cognition.
Ablex, Norwood, NJ.

computadora (CAI) en lectura y matemáticas. El CAI involucraba estudiantes en una actividad educacional en una computadora. Suppes y Atkinson produjeron un entrenamiento matemático usando la computadora. La pantalla de la computadora exhibiría un problema, los estudiantes respondían, y la computadora proveía inmediatamente una retroalimentación.

En los años siguientes se siguen iniciativas como las realizadas por los investigadores de IBM para la creación de sistemas informáticos para la enseñanza, en lo que ya se empezó a conocer como Computer Assisted Instruction (CAI), término que ha sido utilizado hasta nuestros días. A lo largo de la década siguiente se desarrolla el uso de sistemas para el aprendizaje individual basados en el paradigma de la instrucción programada y se prolonga hasta mediados de los 70 con resultados a veces adversos. A partir de este momento también se desarrollan otros enfoques pedagógicos más orientados hacia el cognitivismo pero ahora basados en los sistemas CAI.

Paralelamente, a comienzos de los años 70 surge una propuesta para mejorar los sistemas CAI con la aplicación de las técnicas de Inteligencia Artificial, en completo auge en aquel momento. A este respecto fue Carbonell[4] con su artículo "AI in CAI: An Artificial Intelligence Approach to Computer Aided Instruction" y el desarrollo del SCHOLAR -un sistema tutor inteligente para la enseñanza de la geografía de América del Sur-, quien sentó las bases para el desarrollo de los llamados ICAI (Intelligent CAI) que se puede considerar como el punto de partida de los Sistemas Tutores Inteligentes (ITS), término acuñado por Sleeman and Brown[5] Carbonell propone a los Tutores Inteligentes como sustitutos de los sistemas CAI, como consecuencia a una serie de críticas que se realizan a éstos últimos y que son principalmente: el estudiante carece de iniciativa propia o ésta es muy limitada; no se puede utilizar el lenguaje natural en las respuestas; los sistemas CAI son demasiado rígidos y carentes de iniciativa propia ya que su comportamiento está preprogramado; y no poseen "conocimiento real".

En los años siguientes se proponen arquitecturas genéricas para estos sistemas que desarrollan la modelación de tres tipos de conocimiento:

1. El modelo del alumno,
2. El modelo de la estrategia docente
3. El modelo de conocimiento del dominio o de la materia.

4 **Carbonell, 1970**

Carbonell, J. (1970).

AI in CAI: An artificial intelligence approach to computer aided instruction.

Science, (167):190-202.

5 **Sleeman and Brown, 1982**

Sleeman, D. and Brown, J., editors (1982).

Intelligent Tutoring Systems.

Academic Press.

Arquitectura esta que sigue siendo válida en la actualidad. El marco de referencia de la Inteligencia Artificial en la educación ha marcado en parte el desarrollo de los sistemas de enseñanza asistida por computadora y ha establecido el desarrollo de los Tutores Inteligentes como el principal paradigma de los sistemas educativos basados en computadora hasta nuestros días. Sin embargo los ITS manifiestan una extrema dificultad en la práctica por lo complejo que resultan los modelos cognitivos que intervienen en su diseño, como apunta Terry Mayes[6]:

"The immense dificulty of modelling domain, learner and tutorial strategy in a computationally and pedagogically effective way, have raised many fundamental questions about the viability of this type of approach and led some to abandon ITS approaches altogether".

Por un lado los tutores están restringidos a un dominio particular, no siendo fácil adaptarlos y configurarlos para otros dominios. Además, implementan una determinada estrategia de enseñanza que depende del modelo del alumno para modificarla o personalizarla. Son sistemas de una enorme complejidad en la que se destacan tanto aspectos puramente informáticos como las limitaciones actuales de la Inteligencia Artificial o la psicología educativa, cuyos fundamentos no se han llegado a comprender completamente.

De esta forma, se ha diversificado la búsqueda de soluciones prácticas en algunos casos y en el planteamiento de nuevos paradigmas educativos menos centrados en el conductismo y que se contraponen a la metáfora de la "computadora como tutor" que se lleva a cabo en los ITS. Por un lado aparecen las propuestas basadas en la creación de escenarios para la realización de actividades en grupo, donde poner en práctica las teorías cognitivistas del constructivismo social, que se han traducido en el desarrollo de sistemas basados en el trabajo cooperativo (CSCW) y más concretamente en el ámbito educativo, el aprendizaje cooperativo asistido por computadora (CSCL).

6 **Mayes and Neilson, 1995**

Mayes, T. and Neilson, I. (1995).

Innovate Adult Learning with Innovate Technologies, chapter Learning from other people dialogues: questions about computer based answers, pages 31-48.

Number A61 in IFIP Series. Elsevier Science B.V (North Holland).

1.4 Los entornos integrados de enseñanza

La extensión de las comunicaciones y de la World Wide Web (WWW) ha llevado también aparejado el desarrollo de entornos genéricos para la creación de material educativo para la formación a distancia. Las características fundamentales de los llamados IDLE (Integrated Distributed Learning Environments) están basadas fundamentalmente en el aprovechamiento de las características de accesibilidad y cooperación entre los usuarios de la red. Constan por lo general de una serie de herramientas de gestión y creación de contenido educativo y proporcionan un entorno de desarrollo de material que posteriormente es accesible a través de la red mediante el uso de un cliente o navegador estándar. Estos entornos se basan en la creación integrada de políticas de acceso a servicios conocidos en el ámbito de la red, como son los foros de debate, o sistemas de conferencia electrónica, servicios de compartimiento de archivos, aplicaciones de comunicación síncrona como los llamados "Chat", entre otros. Son muy numerosos los ejemplos que podemos encontrar en este ámbito y tienen una serie de características comunes que son las siguientes:

- Se orientan fundamentalmente hacia el soporte de trabajo en grupo, generalmente para dar servicio de intercambio de archivos o de material entre los alumnos.
- Proporcionan facilidades para el diseño de páginas Web, pero este soporte se limita a crear plantillas a un nivel de la estructura física de las páginas.
- Pueden gestionar grupos para la creación de foros entre grupos cerrados de alumnos o de alumnos supervisados por un tutor.

Hay muchos ejemplos en el mercado de este tipo de entornos. Los más conocidos son los siguientes:

IDLE (Integrated Distributed Learning Environments)	
Forum	Este sistema desarrolla un entorno multiusuario para el compartimiento de archivos y de mensajes, con lo que básicamente se trata de un sistema de conferencia electrónica.
WebCT	Esta herramienta permite la creación de cursos para la realización de una actividad docente. WebCT proporciona herramientas tanto para la construcción de cursos on-line como para la distribución de material en foros organizados en grupo.
Learning Space	Se trata de una herramienta desarrollada sobre el sistema LotusNotes que proporciona cursos en formato multimedia con servidor de agenda para la planificación de la actividad desarrollada sobre el material
First Class	Este sistema proporciona la capacidad de organizar grupos de trabajo para compartir archivos y organizar los accesos de forma selectiva a determinados foros.
Top Class	Este sistema ofrece un servicio de clase virtual sobre una red privada tipo LAN o sobre Internet a través de un navegador o browser estándar

Tabla 1.4 Integrated Distributed Learning Environments.

En general, la descripción de estas herramientas proporciona una visión global de algunos de los productos con los que en la actualidad se ofrece el material electrónico para la docencia a distancia. Los inconvenientes que proporcionan están relacionados con el bajo nivel al que se realiza la autoría de los entornos y contenido estático.

En general, está implementado de forma más completa el aspecto relativo a la gestión de los accesos y de la organización de los contenidos de forma compartida, pero no ofrecen en este sentido toda la flexibilidad que sería menester. En el ámbito de los componentes de contenido, no hay una integración de los mismos en el material en sí, sino que cada uno de ellos (tests, problemas, contenido educativo, foros de discusión, preguntas sobre conceptos, etc.) está organizado en secciones diferentes, lo que hace que se pierda cierta asociación entre ellos. Al mismo tiempo, básicamente, se orientan más hacia un público que demanda una facilidad para la creación de contenidos Web, pero no hacia la creación de material reutilizable.

A finales de la década de los 90 y con el desarrollo de Internet se plantea la idea de crear un marco de referencia para la creación de los sistemas educativos desarrollados en la llamada “sociedad de la información”. En este contexto, las tecnologías educativas (TE) se van a adaptar a la creación de herramientas con el objetivo de facilitar el acceso de los ciudadanos a la educación en el marco del desarrollo tecnológico de la informática y de las telecomunicaciones.

De esta forma, más que desde el punto de vista de los paradigmas educativos, se enfocan los sistemas de enseñanza desde la ingeniería informática aplicada y el diseño de herramientas de aprendizaje, como una necesidad de proporcionar soluciones realistas, pero avanzadas desde el punto de vista de la investigación, a las demandas de formación en un ámbito más social. El objetivo es el de mejorar el proceso de creación, de diseño y de producción de software educativo. En este sentido la importancia de la educación en todos sus ámbitos (universitaria, formación continua, consulta, etc.) dentro del contexto tecnológico es creciente, y la demanda va a ser también creciente en esta área.

1.5 e-learning

La terminología tecnológica está en continuo desarrollo, al igual que el entorno en el que se genera y sobre todo los términos “e-“ que abarcan aspectos muy amplios y muy diversos dependiendo de las organizaciones y de los grupos de usuarios.

El término “e-learning” hizo sus primeras apariciones a finales de 1997, se utiliza actualmente para cubrir casi cualquier tipo de aprendizaje basado en las tecnologías de la información y la comunicación (TIC) en su significado más amplio. Según Elliot Marie, uno de los pioneros y gurús del e-learning, “el e-learning no es un curso puesto en una computadora personal sino una nueva mezcla de recursos, interactividad, rendimiento. Una nueva estructura para el

aprendizaje, una combinación de servicios de enseñanza proporcionados a través del uso de herramientas tecnológicas que proporciona un alto valor añadido: a cualquier hora y en cualquier lugar (anytime, anywhere)”

Definición El e-learning se refiere tanto al entorno como a los procesos de aprendizaje, siendo los contenidos electrónicos solamente una parte del sistema. El término e-learning se entiende como un método de enseñanza-aprendizaje que hace uso de herramientas tecnológicas, recogiendo un amplio abanico de aplicaciones y procesos entre los que se incluye el aprendizaje a través de una computadora personal (PC), el aprendizaje basado en tecnologías Web, clases virtuales, colaboraciones digitales, etc.

El e-learning implica la entrega de contenidos por medios electrónicos como Internet, televisión interactiva, CD ROMs, etc. De manera que el e-learning incluye aquellas sesiones presénciales que utilicen herramientas digitales o electrónicas como medio para la difusión y la práctica de los contenidos de un curso. La industria del e-learning responde a las características de un mercado todavía muy joven: alta fragmentación y baja transparencia y con las líneas entre segmentos muy difusas. La gran mayoría de las compañías de la industria están en manos privadas. Estos altos niveles de fragmentación junto con los largos ciclos de desarrollo y otras deficiencias, hacen que el mercado de e-learning esté preparado para comenzar con las fusiones y las adquisiciones.

A pesar de que se pueden distinguir más de 50 segmentos distintos dentro de la industria, los más importantes y dentro de los cuales se podrían englobar a todos los demás son tres: tecnología, contenidos y servicios. El sector de contenidos es actualmente el que posee una mayor cuota de mercado pero, dada las tasas de crecimiento acumulativo, el sector de servicios superará al de contenidos en un futuro no muy lejano. Las empresas de contenidos están comenzando a incluir valor añadido a sus servicios a través de las evaluaciones, el diseño personalizado de los contenidos, los análisis sobre la eficacia de la formación, etc.

Por otra parte, el sector de servicios ofrece una serie de herramientas para la formación que se pueden dividir en tres grandes grupos: portales, proveedores de servicios de aprendizaje (LSP) y otros servicios profesionales. Los portales y los LSP proporcionan el acceso al aprendizaje y a la formación, agregando, acogiendo y distribuyendo contenidos. Ofrecen servicios de aprendizaje y gestión de contenidos, herramientas de evaluación y pruebas, tutorías online, servicios de colaboración en red y producción, entrega de servicios multimedia entre otros.

2. Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés.

En el mundo actual el manejo del idioma Inglés se ha convertido en algo imprescindible, el inglés es actualmente el lenguaje de la globalización, por lo tanto la carencia de éste dentro de la enseñanza pone en desventaja a cualquier estudiante de cualquier profesional. Por lo tanto es necesario contar con herramientas que sirvan como apoyo dentro del aprendizaje de éste idioma.

1 Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés

En el mundo actual el manejo del idioma Inglés se ha convertido en algo imprescindible, el inglés es actualmente el lenguaje de la globalización, por lo tanto la carencia de éste dentro de la enseñanza pone en desventaja a cualquier estudiante de cualquier profesión. Hoy día el idioma Inglés es fundamental en la comunicación, la transmisión del saber, la construcción del conocimiento, el estudio y en la investigación.

Por lo tanto es necesario contar con herramientas que sirvan como apoyo dentro del aprendizaje de éste idioma. Por otra parte el creciente uso masivo del Internet, así como las tecnologías aunadas a éste medio de comunicación han multiplicado y extendido las posibilidades mismas del aprendizaje. Llevando el concepto de educación más allá de las aulas y al concepto mismo de escuela.

El Internet se ha convertido en un medio para transmitir información y ser una herramienta dentro del aprendizaje. Generando una demanda de software especializado en ello. En este sentido la importancia de la educación en todos sus ámbitos (universitaria, formación continua, consulta, etc.) dentro del contexto tecnológico es creciente.

Gobiernos, organismos públicos nacionales e internacionales y empresas se han dado cuenta de la importancia que tiene el rápido desarrollo de estas tecnologías educativas para la competencia en servicios y para el aprovechamiento de las infraestructuras existentes y las capacidades de comunicación.

1.1 Definición del sistema.

En este contexto el objetivo de esta tesis es el planteamiento del **“Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés” (SWGHAE)**, refiriéndonos también a éste durante este trabajo simplemente como “el sistema” para fines prácticos.

Siendo éste un sistema Web que proporcione un conjunto de herramientas que sirvan de apoyo en la enseñanza del idioma Inglés. Con lo cual contribuir a las crecientes necesidades del aprovechamiento de la tecnología aplicadas a la educación. Éste sistema no pretende ser un sustituto de un curso presencial formal del idioma Inglés sino una herramienta interactiva que contribuya al aprendizaje del mismo. Dotando al usuario del sistema un ambiente donde ejercitar y mejorar sus conocimientos del idioma Inglés. Creando un escenario complementario fuera del aula basado en un modelo de enseñanza individual, con

materiales elaborados para soportar el auto-estudio, y en donde la interactividad alumno-profesor no juega un papel relevante.

1.2 Características del sistema

Un sistema como el propuesto en este trabajo debe crear las condiciones para que se produzca un acceso masivo de la sociedad a las tecnologías educativas, y para ello es preciso que el sistema sea eficiente, escalable y accesible. Partiendo de la primicia de que el uso de la tecnología debe redundar en la mejora de la calidad en la educación. En segundo lugar la solución debe adaptarse a una amplia difusión y por último la diversidad de los usuarios no debe ser un obstáculo para la difusión de estas tecnologías en la sociedad.

El sistema Web aquí planteado propone un escenario educativo donde se distinguen dos tipos de perfiles, un administrador y el usuario del sitio Web.

Por una parte el administrador del sitio, tiene las funciones de generar contenidos estáticos y dinámicos, los cuales serán el material de trabajo de los usuarios finales del sitio. El sistema debe mantener una organización coherente de la información, organizando ésta en niveles y categorías, con la finalidad de ofrecer material adecuado al nivel del usuario.

El usuario del sitio Web es el actor que acudirá al sitio para encontrar herramientas que le ayuden al reforzamiento de sus estudios del idioma Inglés.

1.3 Organización del sistema

En cuanto a la organización sistema, éste esta dividido en dos secciones:

- Sección de administración.
- Sección pública.

En la sección de administración el sistema esta orientada a la generación, publicación y administración de contenidos. A dicha sección solo tendrá acceso el administrador del sistema.

La sección pública es donde el usuario final encontrará las herramientas creadas previamente por el administrador del sistema con las cuales trabajará. Los elementos principales de propuesta están orientados hacia el aprendizaje y ejercitación del idioma Inglés. Para cumplir con este objetivo, se plantean los siguientes módulos de trabajo.

Organización del Sistema basándose en el perfil del usuario	
Módulos Públicos q Materiales q Ejercicios q Artículos q Juegos q Conversación	Módulo de Administración

Tabla 2.1 Módulos que integran el SWGHAE.

2.3.1 Módulo de Materiales

Este módulo esta enfocado hacia la creación de elementos que posteriormente serán utilizados en otros módulos. Estos elementos se organizan en imágenes, audio y archivos.

Este sistema debe facilitar la reusabilidad y la interoperatividad de los componentes. De tal manera que la administración de contenidos pueda ser realizada por cualquier persona. Abarcando la facilidad de intercambio y actualización de contenidos, con el objetivo de proveer un escenario donde el proceso de creación, diseño y producción de material sea un proceso sencillo, rápido y que no requiera de conocimientos técnicos para la elaboración del mismo.

2.3.2 Módulo de Ejercicios

Esta parte del sistema esta dirigida hacia la publicación de ejercicios de diferentes tipos.

- Preguntas con respuestas de opción múltiple.
- Preguntas abiertas
- Ejercicios de audio
- Ejercicios de compresión de lectura
- Estructuras de Idioma

2.3.3 Módulo de Artículos

En esta sección del sistema el usuario puede leer artículos, traducirlos, conocer palabras clave dentro del contexto de la lectura, así como responder preguntas acerca de la lectura con el objetivo complementar su comprensión de la lectura.

2.3.4 Módulo de Juegos

Módulo interactivo formado por varios juegos de destreza, los cuales se deben completar bajo un tiempo determinado para obtener la mayor puntación posible. Esta sección se compone de cuatro juegos de habilidad:

- Ordenar letras para formar palabras en inglés
- Relacionar conjuntos palabras en inglés pertenecientes a un conjunto determinado.
- Relacionar significados de pares de palabras en inglés.
- Completar oraciones con palabras o frases faltantes.

2.3.5 Módulo de Conversación

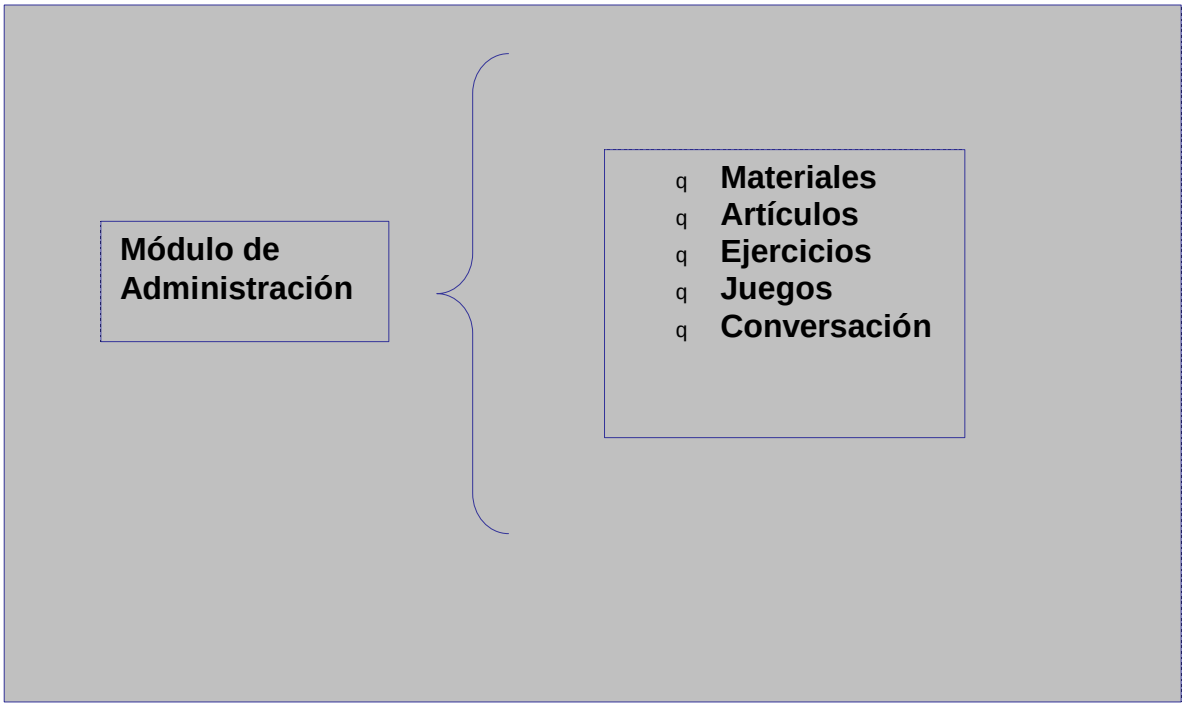
Existen nuevas tecnologías Web que explotan los avances de la inteligencia artificial para poder ofrecer interfaces de robots virtuales(chat-bots) con los cuales se pueden interactuar a través de conversaciones por escrito. Actualmente podemos encontrar este tipo de servicios como www.pandorabots.com que ofrece de manera gratuita la creación de personalidades virtuales a través del uso de AIML (Artificial Intelligence Mark-up Language), un lenguaje que permite introducir conocimientos a un ente denominado “chat-bot” basado en la tecnología A.L.I.C.E (Artificial Linguistic Internet Computer Entity).

Este tipo de herramientas resultan útiles como complemento de práctica para los estudiantes del idioma inglés.

2.3.6 Módulo de Administración

El modulo de Administración permitirá la edición de nuevos contenidos así como la administración de los mismos. Este módulo debe estar protegido bajo la cuenta de administrador. Así también debe estar organizado en cada uno de los módulos públicos que requieran una administración de contenidos, el siguiente diagrama muestra a los submódulos de administración requeridos.

Submódulos de Administración



1.4 Metodología de un Proyecto

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Combina el empleo de modelos o representaciones gráficas junto con el empleo de procedimientos detallados.

Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software. Además se debe tener mucho cuidado cuando se quiere cambiar una técnica por otra.

2.4.1 Marco Contextual.

La Ingeniería de Software (SE del inglés "Software Engineering") es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software. Su origen se debe a que el entorno actual de desarrollo de sistemas software viene adoleciendo de:

- Retrasos considerables en la planificación
- Poca productividad
- Elevadas cargas de mantenimiento
- Demandas cada vez más desfasadas con las ofertas

- Baja calidad y fiabilidad del producto
- Dependencia de los realizadores

Esto es lo que se ha denominado comúnmente "crisis del software", que se ha originado históricamente en los siguientes pasos:

Cuadro descriptivo de las fases pertenecientes a la "Crisis del Software"	
Fases	***Características***
Primera Fase. Los albores (1945-1955)	• Uso de lenguaje máquina y ensamblador. • Programar no es una tarea diferenciada del diseño de una máquina.
Segunda Fase. El florecimiento (1955-1965)	• Aparecen multitud de lenguajes. • Era posible hacer de todo.
Tercera Fase. La crisis (1965-1970)	• Desarrollo inacabable de grandes programas. • Ineficiencia, errores, coste impredecible
Cuarta Fase. Innovación conceptual (1970-1980)	• Fundamentos de programación. • Verificación de programas. • Metodologías de diseño.
Quinta Fase. El diseño es el problema (1980-199?)	• Entornos de programación. • Especificación formal. • Programación automática.

Tabla 2.2 Fases de la llamada Crisis del Software.

2.4.2 Ciclo de Desarrollo de un Sistema.

La formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del software o ciclo de vida del desarrollo del software o ciclo de vida del desarrollo. Se puede describir como, "el período de tiempo que comienza con la decisión de desarrollar un producto software y finaliza cuando se ha entregado éste". Este ciclo, por lo general incluye, una fase de requisitos, fase de diseño, fase de implantación, fase de prueba, y a veces, fase de instalación y aceptación.

Actualmente está surgiendo una gran expectativa ante la evolución de la Ingeniería del Software, al ir apareciendo nuevos métodos y herramientas formales que van a permitir en el futuro un planteamiento de ingeniería en el

proceso de elaboración de software. Dicho planteamiento permitirá dar respuesta a los problemas de:

- Administración
- Calidad
- Productividad
- Fácil mantenimiento

Las nuevas metodologías suponen un enfoque integral del problema, abarcando todas las fases, que en su mayoría no se consideraba en los desarrollos tradicionales. En particular son fundamentales la reducción de costes y plazos, así como la calidad del producto final. Estas tecnologías constituyen la denominada "Ingeniería del Software", que se puede definir como "el tratamiento sistemático de todas las fases del ciclo de vida del software". Hay otras definiciones, pero todas inciden en la importancia de una disciplina de ingeniería para el desarrollo de software. El ciclo de desarrollo software se utiliza para estructurar las actividades que se llevan a cabo en el desarrollo de un producto software. A pesar de que no hay acuerdo acerca del uso y la forma del modelo, este sigue siendo útil para la comprensión y el control del proceso.

2.4.3 Consideraciones para Adoptar un Estándar o Construir una Metodología

En el momento de adoptar un estándar o construir una metodología, se han de considerar unos requisitos deseables, por lo que se proponen una serie de criterios de evaluación de dichos requisitos.

- Consideraciones para la elección de una metodología de desarrollo.

- La metodología debe ajustarse a los objetivos.
- La metodología debe cubrir el ciclo entero de desarrollo de software
- La metodología debe integrar las distintas fases del ciclo de desarrollo.
  - Es importante poder referirse a otras fases de un proyecto y fusionarlo con las fases previas. Es importante poder moverse no sólo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones.
- Fácil interacción entre etapas del ciclo de desarrollo.
  - Es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un impacto en el sistema resultante.
- La metodología debe detectar y corregir los errores cuanto antes.
  - Uno de los problemas más frecuentes y costosos es el aplazamiento de la detección y corrección de problemas en las etapas finales del proyecto. Cuanto más tarde sea detectado el error más caro será corregirlo. Por lo tanto cada fase del proceso

de desarrollo de software deberá incluir una actividad de validación explícita.

- Ø La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo.
  - o La exactitud del sistema implica muchos asuntos, incluyendo la correspondencia entre el sistema y sus especificaciones, así como que el sistema cumpla con las necesidades del usuario. Por ejemplo, los métodos usados para análisis y especificación del sistema deberían colaborar a terminar con el problema del entendimiento entre los informáticos, los usuarios, y otras posibles partes implicadas.
- Ø La metodología debe ser la base de una comunicación efectiva.
  - o Ha de haber una comunicación efectiva entre analistas, programadores, usuarios y gestores, con pasos bien definidos para realizar progresos visibles durante la actividad del desarrollo.
- Ø La metodología debe funcionar en un entorno dinámico orientado al usuario
  - o A lo largo de todo el ciclo de vida del desarrollo se debe producir una transferencia de conocimientos hacia el usuario. La clave del éxito es que todas las partes implicadas han de intercambiar información libremente. La participación del usuario es de importancia vital debido a que sus necesidades evolucionan constantemente. Por otra parte la adquisición de conocimientos del usuario la permitirá la toma de decisiones correctas. Es esencial contar con una buena técnica de diagramación.
- Ø La metodología debe especificar claramente los responsables de resultados
  - o Debe especificar claramente quienes son los participantes de cada tarea a desarrollar, debe detallar de una manera clara los resultados de los que serán responsables.
- Ø La metodología debe poder emplearse en un entorno amplio de proyectos software
  - o Variedad. Una empresa deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías en una misma empresa.
- Ø Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida.
- Ø La metodología debe servir para sistemas de distinta complejidad.
- Ø La metodología debe servir con independencia de la tecnología disponible.
- Ø La metodología se debe de poder enseñar.
  - o Incluso en una organización sencilla, serán muchas las personas que la van a utilizar, incluso los que se incorporen posteriormente a la empresa. Cada persona debe entender las técnicas específicas de la metodología, los procedimientos organizativos y de gestión que la hacen efectiva, las herramientas automatizadas que soportan la metodología y las motivaciones que subyacen en ella.
- Ø La metodología debe estar soportada por herramientas CASE
  - o La metodología debe estar soportada por herramientas automatizadas que mejoren la productividad, tanto del ingeniero de software en particular, como la del desarrollo en general.

- Ø La metodología debe soportar la eventual evolución del sistema
  - o Normalmente durante su tiempo de vida los sistemas tienen muchas versiones. Existen herramientas CASE para la gestión de la configuración y otras denominadas "Ingeniería inversa" para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.
- Ø La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software.
  - o Para mejorar el proceso es básico disponer de datos numéricos que evidencian la efectividad de la aplicación del proceso con respecto a cualquier producto software resultante del proceso. Para disponer de estos datos, la metodología debe contener un conjunto de mediciones de proceso para identificar la calidad y coste asociado a cada etapa del proceso. Sería ideal el uso de herramientas CASE.

Tabla 2.3 Consideraciones para la elección de una metodología de desarrollo.

2.4.4 Importancia de las Herramientas CASE.

Con la aparición de las herramientas CASE junto con los generadores de código, el ciclo de desarrollo software en cascada ha cambiado a un ciclo de vida basado en transformaciones. CASE (Computer Aided Software Engineering), en español "Ingeniería de software Asistida por Computadora", es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información.

La utilización de herramientas CASE afecta a todas las fases del ciclo de vida del software. Este ciclo de vida se puede considerar como una serie de transformaciones. Primero se definen los requisitos del sistema, seguido de un proceso de transformación que hace que la especificación se convierta en un diseño lógico del sistema. Posteriormente, este sufre otro proceso de transformación para lograr un diseño físico, es decir que responda a la tecnología destino.

La tecnología CASE propone que estos procesos de transformación sean lo más automatizados posible. Sus ventajas son:

- Posibilidad de comprobación de errores en etapas iniciales de desarrollo.
- Posibilidad de realizar el mantenimiento en el ámbito de especificación.
- Soporte de seguimiento de los requisitos.
- Soporte de reusabilidad.
- Potencia la especificación orientada al problema.

1.5 Metodología del Proyecto

Una vez revisados los puntos previos, considerados para tener presentes los problemas típicos al enfrentarse al desarrollo de un software, así como los puntos mínimos tanto para la elección y la justificación de una metodología de desarrollo para el "Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés" (SWGHAE) es momento de tomar la decisión de la metodología de desarrollo.

Definición Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc. Un paradigma de programación es una colección de modelos conceptuales que juntos modelan el proceso de diseño y determinan, al final, la estructura de un programa.

Esa estructura conceptual de modelos está pensada de forma que esos modelos determinan la forma correcta de los programas y controlan el modo en que pensamos y formulamos soluciones, y al llegar a la solución, ésta se debe de expresar mediante un lenguaje de programación. Para que este proceso sea efectivo las características del lenguaje deben reflejar adecuadamente los modelos conceptuales de ese paradigma.

2.5.1 Metodología y Diseño Orientado a Objetos.

La creciente tendencia de crear programas cada vez más grandes y complejos llevó a los desarrolladores a crear una nueva forma de programar que les permita crear sistemas de niveles empresariales y con reglas de negocios muy complejas. Para estas necesidades ya no bastaba la programación estructurada ni mucho menos la programación lineal. Es así como aparece la programación orientada a objetos (POO).

La POO viene de la evolución de la programación estructurada; básicamente la POO simplifica la programación con la nueva filosofía y nuevos conceptos que tiene. La POO se basa en la dividir el programa en pequeñas unidades lógicas de código. A estas pequeñas unidades lógicas de código se les llama objetos. Los objetos son unidades independientes que se comunican entre ellos mediante mensajes.

Para el desarrollo de esta tesis se eligió el paradigma orientado a objetos, sintetizando dicha decisión en los siguientes puntos:

- El paradigma orientado a objetos pone énfasis en el aspecto de modelado del sistema, examinando el domino del problema como un conjunto de objetos que se comunican entre sí mediante mensajes. De este modo, la estructura de los programas refleja directamente la del problema que se desea simular.

- Utilizando las metodologías de análisis y diseño orientado a objetos (AOO y DOO, respectivamente) y sus lenguajes (LOO), el software se construye a partir de objetos con un comportamiento específico. Los propios objetos se pueden construir a partir de otros, que a su vez pueden estar formados por otros objetos. Este proceso ofrece un beneficio muy importante, y que constituye la razón fundamental por la cual la ingeniería del software se ha abocado a este nuevo paradigma: la reusabilidad.

- Cuando se construye un nuevo programa se obtienen piezas para futuros programas, lo cual conduce a que el software se elabore por ensamblaje de objetos desarrollados por uno mismo o por otros para otras aplicaciones. Estos objetos pueden ser cada vez más complejos desde el punto de vista interno, pero más sencillos en cuanto a la interacción con ellos, ya que se pueden conceptuar como “cajas negras” susceptibles de ser utilizadas sin mirar en su interior. Lo que se persigue es disponer de depósitos con una gran cantidad de objetos, y de herramientas que permitan encontrar el tipo necesario de objeto, reutilizarlo y ampliar su comportamiento de la forma más conveniente.

Para el desarrollo de software orientado a objetos no basta usar un lenguaje orientado a objetos. También se necesitará realizar un análisis y diseño orientado a objetos. El modelo visual es la clave para realizar el análisis OO.

Desde los inicios del desarrollo de software orientado a objetos han existido diferentes metodologías para hacer dicho modelado, pero sin lugar a duda, el Lenguaje de Modelado Unificado (UML) puso fin a la guerra de metodologías. UML consta de todos los elementos y diagramas que permiten modelar los sistemas basándose en el paradigma orientado a objetos. Es por las anteriores razones que el desarrollo de este sistema será llevado apoyado de UML.

En el proceso de modelado orientado a objetos (OO) podemos distinguir las tres fases tradicionales de elaboración de un sistema informático:

1. Análisis: se ocupa del qué, de entender el dominio del problema.
2. Diseño: responde al cómo, y se centra en el espacio de la solución.
3. Implementación: fase en la que se adapta la solución a un entorno de programación específico.

1.6 Análisis del Sistema

Definición Análisis es el proceso de clasificación e interpretación de hechos, diagnostico de problemas y empleo de la información para recomendar mejoras al sistema.

El primer paso en el desarrollo de nuestra aplicación es el proceso de documentación de requerimientos informalmente en un pequeño conjunto de casos de uso.

El objetivo principal del modelo de casos de uso es capturar aquellas acciones que producen un resultado de valor para el usuario. Esto es especialmente útil cuando el modelador es ajeno al sistema a modelar. En caso contrario, los casos de uso pueden resultar un artefacto que proporciona un valor informativo general relativamente bajo. Sin embargo, aun siendo un conocedor o incluso un experto en el sistema a modelar, puede resultar aclarador el hecho de redactar ciertos casos de uso ya sea por su especial relevancia o por su particular complejidad.

Con el afán de ejemplificar de manera limitada se muestra el caso de uso del alta de un artículo del submódulo Artículos de la parte de Administración del Sistema. A continuación se muestra la redacción del caso de uso.

Nombre:	Alta de artículo
Objetivo:	Ingresar en el sistema un artículo
Actores	El administrador del Sistema.
Precondiciones:	El administrador debe estar autentificado en el sistema.
Flujo Principal:	A través de una interfaz Web el administrador llenará los campos correspondientes a la alta de un artículo para ser guardados en la base de datos: q Titulo q Autor q Resumen q Cuerpo del Articulo
Flujo Alternativo	El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija
Poscondiciones	El artículo ha sido almacenado en la base de datos del sistema.

Tabla 2.4 Redacción del caso de uso referente a dar de alta un nuevo articulo en el sistema SWGHAE.

Actores son aquellos que interactúan con el sistema. Las precondiciones son los hechos que se han de cumplir para que el flujo de evento se pueda llevar a cabo. Luego tenemos el flujo de eventos, que corresponde a la ejecución normal y exitosa del caso de uso. Los flujos alternativos son los que nos permiten indicar qué es lo que hace el sistema en los casos menos frecuentes e inesperados. Por último, las poscondiciones son los hechos que se ha de cumplir si el flujo de eventos normal se ha ejecutado correctamente.

Adicionalmente a un documento de caso de uso como el mostrado previamente también podemos agregar un diagrama de caso de uso. Un diagrama de caso de uso es una técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos.

Es siguiente diagrama muestra la interacción de un administrador con la parte de administración del sistema para dar de alta un artículo.

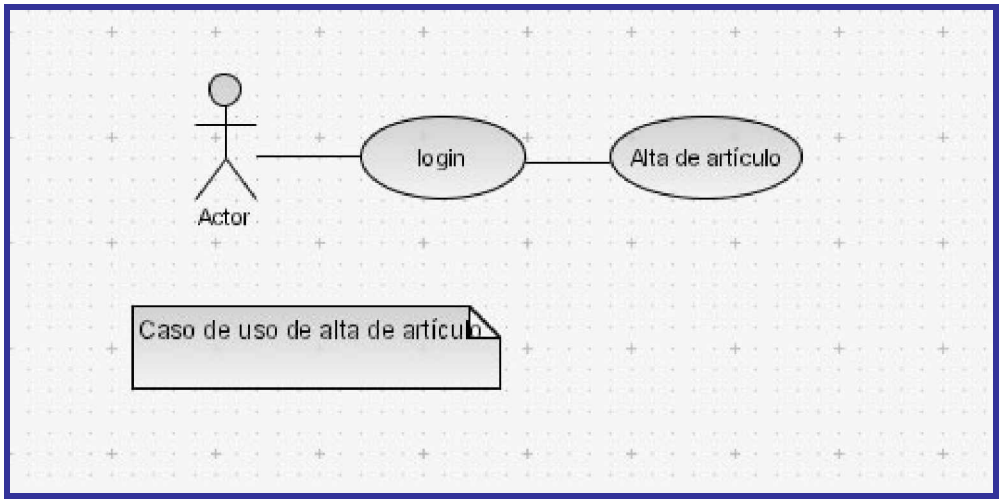


Figura 2.1 Caso de uso referente a dar de alta un nuevo articulo en el sistema SWGHAE.

Como nota podemos decir que los diagramas de caso de uso cumplen con las siguientes características:

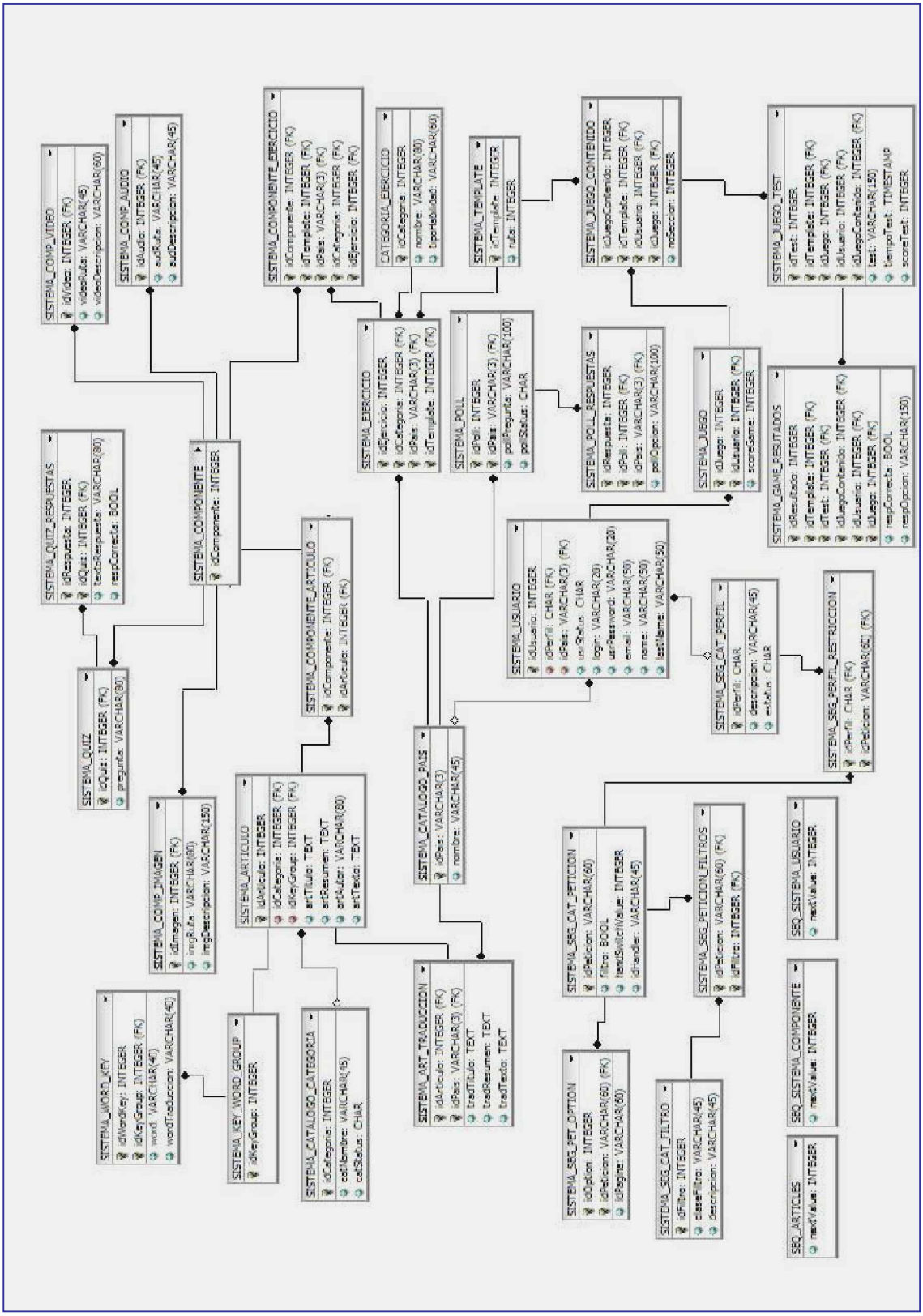
- Los Casos de Uso (Ivar Jacobson) describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario.
- Permiten definir los límites del sistema y las relaciones entre el éste y el entorno.
- Los Casos de Uso son descripciones de la funcionalidad del sistema independientes de la implementación.
- Los Casos de Uso cubren la carencia existente en métodos previos (OMT, Booch) en cuanto a la determinación de requisitos.

- Los Casos de Uso dividen el conjunto de necesidades atendiendo a la categoría de usuarios que participan en el mismo.
- Están basados en el lenguaje natural, es decir, es accesible para todos usuarios.

Después de presentar de manera sintetizada la descripción de cada uno de los módulos que integran el sistema Web de generación de herramientas para el aprendizaje del idioma Inglés, así como el análisis de uno de sus módulos, se presenta a continuación el diagrama entidad relación resultado de del análisis previo presentado.

Diagrama Entidad Relación del Sistema

Figura 2.2



3. Plataforma de Desarrollo para el Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés.

Hoy en día la elección de la plataforma tecnológica no es solo una decisión técnica, es sin duda una decisión estratégica. La clave es elegir sistemas de probada fiabilidad y aceptación, que proporcionen opciones de progreso que sean, sobre todo rentables, fiables y eficaces.

3 Plataforma de Desarrollo

El estado actual de las comunicaciones electrónicas, el incremento en el poder de hardware, nuevas tecnologías en el desarrollo de software, nuevos manejadores de bases de datos y diferentes arquitecturas de las aplicaciones en red, han incrementado la complejidad de elegir una plataforma de desarrollo adecuada a las necesidades de un proyecto.

Es la evolución de la tecnología ha hecho que las expectativas con respecto al software cambien y sean más exigentes. Los grandes hitos tecnológicos como la aparición de las computadoras personales y más recientemente Internet y las tecnologías móviles, han hecho que el software se convierta en algo que ha influido en todas las actividades.

En la época actual la integración de la tecnología se debe hacer desde las primeras etapas de diseño, y obliga al ingeniero de software a ampliar su dominio de competencias. El ingeniero de software debe manejar un modelo claro de las tecnologías, para poder tomar las decisiones adecuadas e integrarlas con el resto de elementos del programa. A partir de la selección de los elementos que van a definir la estructura de una aplicación, hay que definir el proceso de construcción, la metodología, la tecnología y las herramientas.

3.1 Evolución de Arquitecturas

3.1.1 Sistemas Multiusuario

Durante los años 60 y 70, las empresas que requerían tener gran poder de cómputo utilizaban una arquitectura centralizada. Consistía en una computadora única o mainframe a la cual, bajo el control de un sistema operativo multiusuario, se conectaban terminales “tontas” (llamadas así por carecer de un procesador).

La única función de las terminales era transmitir a la computadora central. Todos los programas o lógica de la aplicación, junto con sus datos, residía en el mainframe, el que generalmente con un solo procesador central (CPU) atendía todos los requerimientos introducidos a través de las terminales.

3.1.2 Servidor de Archivos

A finales de la década de los 70, aparecieron las computadoras personales y con ellas, ya en la década de los 80, se hicieron evidentes sus grandes ventajas y desventajas. Por un lado su facilidad de uso que propició que las personas con pocos conocimientos técnicos las pudieran aprovechar. También importante fue la competencia que se desencadenó entre las empresas de cómputo, misma que

trajo consigo una explosión en el poder de cómputo y en la complejidad de los programas. Estos desarrollos acercaron aún más las computadoras a los usuarios finales.

Bajo estas circunstancias nació el concepto de “Servidor de Archivos”, esta arquitectura consiste en conectar varias computadoras personales o Workstations a una computadora central (o varias) la cual proporciona acceso a recursos de cómputo como impresoras y discos duros. La lógica de la aplicación reside ahora en los “clientes” en lugar del servidor. El servidor se encarga de enviar archivos o grandes bloques de datos conforme le son requeridos.

Entre las ventajas de esta arquitectura se encuentra un costo inicial bajo y acomodo flexible. Entre las desventajas tenemos que el cliente ejecuta casi toda la aplicación: interfase con el usuario y reglas del negocio, por lo cual se requiere que tenga suficiente poder de cómputo. Si es una aplicación muy demandante de recursos quizá sea imposible hacer crecer al cliente a un nivel satisfactorio. Adicionalmente, a largo plazo, el costo de la arquitectura se incrementa considerablemente. Esta forma de operación genera una sobrecarga de la red. Los modelos de servidor de archivos todavía están en uso y son ideales en aplicaciones donde se comparten archivos, por ejemplo para crear depósitos compartidos de documentos, imágenes o planos de ingeniería.

3.1.3 Cliente - Servidor

Una manera de eliminar las desventajas de la arquitectura de servidor de archivos, es la arquitectura cliente - servidor. Uno de los objetivos de este modelo es el de distribuir lo mas eficientemente posible las cargas del proceso entre todas las computadoras. El modelo cliente - servidor es un paradigma en el cual los elementos de un sistema de computación requieren servicios desde otro sistema para completar una tarea. El que requiere el servicio se denomina cliente y el que lo proporciona es el servidor.

Es una relación de servicio entre procesos. El proceso servidor es un proveedor de servicios y el cliente es un consumidor de servicios. La arquitectura cliente – servidor tiene las siguientes características:

- Los servidores pueden conectarse a uno o a más de un cliente
- Los clientes pueden comunicarse con múltiples servidores
- Es posible una comunicación cliente a cliente y una comunicación servidor a servidor
- Los clientes siempre inician el diálogo al requerir un servicio. Los servidores esperan pasivamente requerimientos de sus clientes.
- La interacción se logra mediante un intercambio de mensajes
- Los elementos cliente - servidor pueden existir en diferentes computadoras a través de la red o en una sola computadora.

Los sistemas Cliente-Servidor se pueden escalar (hacer crecer) horizontal y verticalmente. El escalamiento horizontal se da cuando aumentamos mas clientes quizá disminuyendo ligeramente el desempeño del sistema. El escalamiento vertical quiere decir cambiar el servidor o distribuir la carga en varios servidores.

3.1.4 Cliente – Servidor de N Capas

El concepto de n capas ha cambiado con el tiempo, puede referirse tanto a hardware como a software. La más clara definición se refiere al hardware, donde una arquitectura será de n capas si esta repartida en n computadoras. Desde el punto de vista de software, la arquitectura Cliente-Servidor de tres capas está dada por los tres elementos funcionales básicos que generalmente tiene una aplicación: presentación, lógica de negocio y datos. Esto es cada aplicación que interactúa con un usuario necesita una sección de software que se encargue de establecer la interfase hombre – máquina. Así mismo, si la aplicación requiere mantener resultados a través del tiempo, es necesaria una sección que maneje los datos y por último, el procesar esos resultados (lógica de negocio).

Entre las ventajas de la arquitectura de tres capas mencionamos las siguientes:

- Administración menos compleja ya que en los servidores de cada capa se descarga el grueso de la administración.
- La seguridad se incrementa porque se pueden establecer controles finos en cada capa y, además por ejemplo, en vez de actuar con los datos directamente, el cliente tiene contacto con las reglas del negocio y nada más. Los datos y las reglas quedan encapsulados.
- Facilidad de escalamiento (crecimiento), intrínseca en el modelo.
- Soporte para bases de datos heterogéneas.
- Mayor flexibilidad de distribución de las funciones de la aplicación.

3.1.5 Soluciones Por Componentes

Un nuevo modelo está cambiando la forma de organizar las aplicaciones y está revolucionando el desarrollo de las mismas: las aplicaciones Cliente Servidor con Objetos Distribuidos. Esta nueva tecnología promueve la construcción de aplicaciones eficientes, robustas, “escalables” y “mantenibles”. Brindando la posibilidad de instalar sistemas de información complejos simplemente ensamblando y/o extendiendo las capacidades de componentes de software reutilizables que trabajan juntos, independientemente del tipo de computadoras, redes, lenguajes y sistemas operativos. Cualquiera de estos componentes de software puede ser modificado o reemplazado sin afectar al resto de los componentes en el sistema y sin modificar la forma en que interactúan.

3.1.5.1 Componentes

Los componentes son una extensión de los objetos descritos en la programación orientada a objetos. Como tales son una fusión de código y datos que funcionan como una unidad y funcionan también como objetos en el sentido de que tienen herencia por interfase, polimorfismo y encapsulación. Por el contrario, a diferencia de los objetos tradicionales, los componentes pueden interactuar entre ellos independientemente de lenguajes, herramientas, sistemas operativos y redes y pueden extenderse utilizando herencia por implementación.

Un componente (objeto distribuido), es un programa inteligente que puede vivir en cualquier lugar en la red. Se empacan como piezas independientes e inteligentes de código, a los cuales pueden acceder clientes remotos vía invocaciones de sus métodos. El lenguaje y el compilador que se utilizaron para crearlos (los componentes servidores) no son importantes para los componentes clientes. Adicionalmente los componentes clientes no necesitan conocer en donde se encuentran los componentes servidores o que sistema operativo los controla para poder tener acceso a ellos.

3.1.5.2 Object Request Brokers (ORB)

Para que el esquema de componentes distribuidos trabaje, debe existir algún mecanismo para conocer el lugar en que se encuentran y para encaminar los requerimientos de los componentes cliente y las respuestas de los componentes servidor. Estos mecanismos son conocidos como Object Request Brokers (ORB) los cuales funcionan como un intermediario o gestor de componentes.

Un object request broker es el software que reside en la parte media o middleware del esquema Cliente - Servidor y que se encarga de comunicar componentes cliente con componentes servidor. El cliente invoca un método de un componente remoto, el ORB localiza un ejemplar de la clase de ese componente, invoca el método solicitado que hace que el componente trabaje y regresa el resultado al objeto cliente.

Obviamente, si cada fabricante se dedicara a elaborar componentes sin el seguimiento de algún estándar, la comunicación entre los mismos sería imposible. En el presente existen dos tecnologías o estándares importantes para el manejo de los componentes: CORBA y COM. Estos dos estándares, junto con otras infraestructuras tales como los Object Transaction Monitors (OTM), y metodologías unificadas de análisis y diseño, han permitido que una tecnología que tiene mas de 20 años de existir logre consolidarse y hacerse útil. A la fecha existen varios ORB’s comerciales que siguen el estándar CORBA. Como ejemplo están: Orbix, VisiBroker, JavaIDL, ObjectBorker y PowerBroker entre otros. Por el otro estándar que es patrocinado por Microsoft, solo existe un ORB llamado DCOM.

3.1.5.3 El estándar CORBA

CORBA son las siglas de Common Object Request Broker Arquitecture. Estas siglas designan a un conjunto de estándares que forman un marco de referencia para establecer la interacción de los componentes. Fue elaborado por el OMG (Object Management Group), una institución formada por la mayoría de las compañías más importantes de hardware y software del mundo, que tiene por objetivo promover asuntos de orientación a objetos y la promoción de estándares abiertos para sistemas orientados a objetos. CORBA no es un producto sino solo un estándar. Las especificaciones se escriben en un lenguaje de definición neutral (Interfase Definition Language - IDL) que define la frontera de un componente, es decir, su interfase con clientes potenciales. CORBA fue diseñado desde su origen para resolver el problema de intercomunicar máquinas, es entonces una arquitectura “ad hoc” para manejar componentes remotos o distribuidos. La especificación CORBA establece una infraestructura para que exista una comunicación entre procesos dentro de la misma máquina o entre diferentes máquinas. El empaquetamiento de componentes no fue el primer objetivo durante las primeras fases del desarrollo de CORBA; sin embargo, provee una interacción entre lenguajes, plataformas y vendedores. Debido a que se ha instrumentado en un rango amplio de plataformas y vendedores (mainframe, Unix, Windows), CORBA es la arquitectura dominante en la distribución de objetos remotos.

3.1.5.4 El Estándar DCOM

A principios de los años 90, Microsoft realizó un gran esfuerzo en promover OLE (Object Linking and Embedding) que se podría traducir como incrustación y vinculación de objetos. La necesidad de hacer esto nació del deseo de los usuarios de computadoras personales de elaborar documentos que incluyeran tanto texto como gráficas y tablas. Dado que cada programa es especialista en alguna tarea, por ejemplo un procesador de texto sirve para manejar texto, los programas de hojas de cálculo sirven para el manejo de tablas y números y los de diseño para gráficas, etc. De tal forma que resultaba difícil incrustar los resultados de uno en otro. Rápidamente Microsoft reconoció la necesidad de un mecanismo estándar de empaquetamiento de componentes si se quería que OLE fuera exitoso. Era crucial que estos componentes fueran independientes de lenguajes para que se pudieran desarrollar con cualquier lenguaje e incrustarse en otro componente posiblemente escrito en otro lenguaje.

Microsoft creó COM (Component Object Model) modelo de objetos componentes, que es un estándar para la creación y comunicación entre componentes pero dentro de la misma máquina. Al final de 1996, Microsoft introdujo DCOM (La D es de distribuido) un conjunto de extensiones a COM que permite distribuir a los objetos COM en varias computadoras en la red. Ha existido una lentitud en que aparezca COM en plataformas diferentes a Windows, debido a esto, regularmente se identifica a COM como una arquitectura de componentes en lugar de una

arquitectura de componentes remotos o distribuidos; sin embargo, COM tiene mucho que ofrecer en este sentido, especialmente cuando se trabaja en ambientes Windows.

3.1.5.5 Object Transaction Monitors (OTM)

Se mencionó ya, que el ORB (Object Request Broker) es el mecanismo encargado de realizar la comunicación entre los componentes. Encuentra al componente requerido por el cliente, acciona los métodos de dicho componente y devuelve al cliente el resultado; es simplemente un bus de comunicación entre componentes. Sin embargo, un ORB no facilita del todo las cosas. Queda pendiente una administración efectiva y eficiente de los componentes a los que los clientes pueden acceder. Ejemplos de aspectos pendientes serían el de seguridad, permisos, ciclo de vida de los componentes, cuando y como llamar a todos los servicios del ORB, etc. Una alternativa de solución es escribir código para que ellos mismos realicen estas funciones. Esta solución convertiría a los componentes en piezas difíciles de manejar y requeriría programadores con mucha experiencia en un ORB en particular, adicionalmente, los componentes servirían o estarían optimizados para un ORB pero para otros no, lo cual dificultaría la comunicación entre brokers.

La solución que se ha encontrado es emplear un OTM (Object Transaction Manager) o gestionador de las transacciones de los componentes. Un OTM es el software que se encarga de crear un medio ambiente organizado para manejar y controlar los componentes del lado del servidor. Permite definir y administrar los componentes típicamente en un medio ambiente gráfico. El OTM establece un pool de componentes, distribuye sus cargas y coordina las transacciones entre componentes. El usuario escribe la lógica y a tiempo de ejecución, el OTM intercepta todas las llamadas a componentes, invoca los componentes requeridos y le pasa una conexión del componente al cliente haciendo las veces de un orquestador. El resultado es un manejo automático de los componentes del lado del servidor haciéndolos robustos, persistentes, seguros y eficientes. Cuando se utiliza un OTM se logra que los componentes adquieran características adicionales en su funcionamiento como las que se mencionan a continuación:

- Seguridad: Un componente puede protegerse así mismo y a sus recursos, desde el medio ambiente. Puede identificar a sus clientes y viceversa, provee controles de acceso y conserva estadísticas de su utilización.
- Permisos: Puede establecer políticas de permisos.
- Versiones: El componente provee una forma de control de versiones para asegurar que sus clientes utilizan la versión correcta.
- Manejo del ciclo de vida: Se puede manejar la creación, destrucción, almacenamiento, publicación del contenido y el traslado de un lugar a otro de los componentes.
- Persistencia: El componente puede salvar su estado y posteriormente restablecerlo.

- Creación de colecciones (suits) de componentes dentro de contenedores que facilitan una relación más estrecha, dinámica y eficiente entre componentes propios de un tema o dominio (Por ejemplo hotelería, las finanzas, etc.).
- Auto Instalación: El componente se puede auto instalar y automáticamente también registrarse en el sistema operativo.

Varias compañías han desarrollado OTM's. Entre las que siguen el estándar de la OMG están los siguientes: Application Server 4.0 de Oracle, PowerTier de Persistence, NetDynamics de Sun, Jaguar CTS de Sybase, Etc. Por otra parte, el OTM de Microsoft se llama Microsoft Transaction Server (MTS).

3.1.5.6 Objetos de negocio

Solo falta una cosa para hacer completa una aplicación Cliente Servidor moderna: Los componentes de negocio. La última meta es crear componentes que se comporten como objetos de negocio del mundo real, son objetos que modelan el comportamiento del mundo en algún tema o dominio, realizan funciones características del objeto que representan. Los objetos de negocio son perfectos para crear aplicaciones de tres y más capas. En la primera capa se colocan los componentes visuales que tienen que ver con el usuario para el manejo de la aplicación. Por lo general estarán en el cliente. En la(s) capa(s) de en medio se coloca los componentes servidores del negocio que contienen las reglas del negocio y por último, en la tercera capa se encontrarán típicamente las bases de datos. Los componentes de las capas de en medio interactúan con sus clientes. Estos a su vez, pueden obtener sus datos desde múltiples fuentes, por ejemplo bases de datos, archivos planos, etc. Estos componentes servidores presentan a sus clientes una visión integrada y uniforme, de tal manera que el cliente no se entera ni se preocupa de donde están los datos, a los cuales nunca tiene acceso directamente lo que hace mayor la seguridad. Estas fuentes de datos son totalmente ocultadas a los clientes pudiéndose inclusive, repartir en varios equipos.

3.2 Elección de arquitectura para el sistema

Como se ha descrito en las secciones anteriores el estado actual de las comunicaciones electrónicas, el incremento en el poder de hardware, nuevas tecnologías en el desarrollo de software, nuevos manejadores de bases de datos y diferentes arquitecturas en el "acomodo" de las aplicaciones dentro de la red, han hecho posible crear sistemas que se acerquen cada día más, a tener esas características deseadas.

Típicamente las aplicaciones que proveen estos servicios deben combinar los existentes sistemas de información empresarial (EIS por sus siglas en inglés) con

nuevas funcionalidades de negocio que permiten ofrecer un conjunto de servicios a un ancho rango de usuarios (clientes, socios, empleados y proveedores) Estos servicios necesitan contar con las siguientes características:

- Disponibilidad: Conocer las necesidades actuales del ambiente global comercial.
- Seguridad: Proteger la privacidad de los usuarios y la integridad de la información.
- Confiable y Escalable: Asegurar que la transacción de negocios se procesen de forma puntual y confiable.

Por una variedad de razones, estos servicios son generalmente organizados como aplicaciones distribuidas consistentes en varias capas en clientes (front end), recursos de información (back end), y una o más en la capa intermedia. La capa intermedia implementa nuevos servicios que integran los sistemas de información empresarial con las funciones de negocio y la información de nuevos servicios. La capa intermedia (capa de negocios) protege a la capa cliente de la posible complejidad de ésta, con el objetivo de que el cliente tome ventaja de las tecnologías de Internet.

Este es el esquema de funcionamiento de las tecnologías de desarrollo empresarial. Actualmente existen dos tecnologías dominantes para desarrollar aplicaciones empresariales el framework .NET de Microsoft y la plataforma J2EE de Sun Microsystems.

3.3 Comparativa entre las tecnologías .NET Y J2EE

3.3.1 El Framework .NET

En esta sección se estudiarán las principales características que ofrecen el framework .NET y J2EE.

.NET es una plataforma de software desarrollada con base en los estándares de Servicios Web XML que conecta información, sistemas y dispositivos. La plataforma .NET conecta una grande variedad de tecnologías de uso personal y de negocios, de teléfonos celulares a servidores corporativos, permitiendo el acceso a información importante, donde y cuando se necesiten.

.NET es un componente presente en toda la línea de productos Microsoft, ofreciendo la capacidad de desarrollar, implementar, administrar y utilizar soluciones conectadas a través de Servicios Web XML. Estas soluciones permiten una integración más rápida y ágil entre las empresas y el acceso a información a cualquier hora, en cualquier lugar y a través de cualquier dispositivo.

La idea fundamental de Microsoft .NET es un cambio de enfoque en lo que es la informática, pasando de un mundo de aplicaciones, sitios Web y dispositivos aislados a una infinidad de computadoras, dispositivos, transacciones y servicios que se conectan directamente y trabajan en conjunto para ofrecer soluciones más amplias y ricas en contenido. Las personas tendrán el control sobre cómo, cuándo y qué información desean. Las computadoras, sistemas y servicios estarán en capacidad de colaborar e interactuar mutuamente para beneficiar al usuario, mientras que las empresas podrán ofrecer sus productos y servicios a los clientes apropiados, en el momento correcto y de la forma precisa, combinando procesos de manera mucho más granular de lo que es posible hoy.

.NET Framework está integrado directamente en la familia de Windows Server 2003. Esto elimina la necesidad de una implementación o una administración adicionales. Como resultado, Windows Server 2003 se beneficia de las ventajas de .NET Framework: Un entorno de ejecución de aplicaciones y un modelo de programación completamente administrado, protegido y con muchas características. Implementación y desarrollo simplificados. Integración sin problemas con una gran variedad de lenguajes de programación.

El Framework .NET de Microsoft como el término en inglés dice (Framework = Armazón) es un marco en donde nuestras aplicaciones se ejecutarán. Las aplicaciones ya no corren directamente bajo el sistema operativo si no que corren bajo este armazón o marco.

Los elementos principales del .NET Framework son:

- CLR (Common Language Runtime)
- El conjunto de clases del .NET Framework
- ASP.NET
- Los servicios Web
- Remoting
- Windows Forms

El CLR es el motor de ejecución de las aplicaciones .NET (lo que en Java sería la máquina virtual), este motor se encarga de ejecutar todo el código .NET. El CLR es el encargado de convertir este lenguaje intermedio en leguaje máquina del procesador, esto normalmente se hace en tiempo real por un compilador JIT (Just-In-Time) que lleva incorporado el CLR.

El conjunto de clases del .NET Framework es un rico conjunto de clases, interfaces, tipos que simplifican y optimizan el desarrollo de aplicaciones .NET además de proporcionar acceso a la funcionalidad del sistema. Como desarrolladores el dominio de este conjunto de clases es vital para un buen desarrollo en .NET.

ASP.NET es la parte del .NET Framework dedicada al desarrollo Web. A través del servidor Web (IIS) las aplicaciones ASP.NET se ejecutarán bajo el CLR con lo cual se puede usar el conjunto de las clases del .NET Framework para desarrollarlas, obteniendo una versatilidad y potencia mayores en las aplicaciones ASP. También son destacables los servicios Web, que nos permitirán comunicarnos a través de Internet entre diferentes computadoras, incluso entre distintos sistemas. Así como .NET Remoting el cual permite tener objetos en máquinas remotas e invocarlos desde otras máquinas. Y las Windows Forms, parte del .NET Framework.

3.3.1.1 Vista general de ASP.NET

ASP.NET se ha construido bajo los siguientes principios:

- Facilidad de desarrollo
- Alto rendimiento y escalabilidad
- Mejorada fiabilidad
- Fácil distribución e instalación
- Facilidad de desarrollo

ASP.NET introduce un nuevo concepto, los "server controls", que permiten a modo de etiquetas HTML tener controles manejados por el servidor que identifican el navegador usado adaptándose para cada navegador. Tareas tediosas como la validación de datos se convierten en fáciles y sencillas. Permite la posibilidad de elección del lenguaje de programación, por defecto lleva integrado C#, VB.NET y J#, con las posibilidad usar otro lenguaje. Con respecto a la elección de una herramienta de desarrollo. Es posible utilizar desde el Notepad, hasta la sofistica y potente Visual Studio NET, o bien la gratuita Web Matriz. Dicha tecnología también ofrece una rica biblioteca de clases lo que facilita tareas como el envío de correo electrónico, hacer un “upload” de un archivo o generar gráficos en tiempo de ejecución.

3.3.1.2 Java 2 Enterprise Edition (J2EE)

La plataforma Java 2 Edición Empresarial (J2EE) es un conjunto de especificaciones y prácticas coordinadas que juntas permiten soluciones para el desarrollo, despliegue y administración de aplicaciones multicapa.

Al utilizar J2EE Sun Microsystems ofrece una reducción del costo y complejidad de desarrollo y despliegue en soluciones multicapa.

J2EE provee un framework para desarrollar y desplegar web services sobre la plataforma Java. El API Java para XML-based RPC (JAX-RPC) permite a los desarrolladores Java crear web services basados en SOAP. Utiliza “contenedores” para simplificar el desarrollo. Estos “contenedores” permiten a los desarrolladores centrarse en escribir la lógica de negocio en lugar que ocuparse

en la infraestructura empresarial. Por ejemplo el contendedor de Enterprise Java Beans (EJB) implementado por los vendedores de la tecnología J2EE maneja las comunicaciones distribuidas, hilos de ejecución, estabilidad, administración de transacciones etc. De forma similar, La tecnología Java Servlets simplifica el desarrollo web con una infraestructura para la comunicación de componentes, administración de sesiones en un contenedor Web que esta integrado con un Web Server.

Entre las ventajas de la utilización de la plataforma J2EE se encuentran:

- Libre Elección. La tecnología J2EE es un conjunto de estándares que algunos vendedores pueden implementar. Los vendedores son libres de competir sobre implementaciones pero no sobre estándares o APIs. Sun provee un amplio conjunto de prueba de compatibilidad J2EE para licencias J2EE. Estas pruebas aseguran la compatibilidad entre los vendedores para ayudar a asegurar al portabilidad para las aplicaciones y los componentes escritos para la plataforma J2EE. La plataforma J2EE sigue con la filosofía “escribe una vez, y corre donde en cualquier parte” (WORA) para el servidor.

- Conectividad simplificada. La tecnología J2EE logra facilitar la conexión entre las aplicaciones y sistemas que ya se tiene y trae estas capacidades a la Web, teléfonos celulares y dispositivos. J2EE ofrece Java Message Service para integrar diversas aplicaciones. La tecnología J2EE también ofrece soporte CORBA para una estrecha unión de sistemas a través de llamadas remotas. Adicionalmente la plataforma tiene Conectores para unir a los sistema de información empresarial tales como ERP Systems, sistemas de aplicaciones financiera y customer relationship management (CRM).

El siguiente cuadro resume y compara las tecnologías J2EE y .NET

Stack Function	.NET	J2EE
Relational Database Access	ADO.NET	JDBC
Web Client	ASP.NET	Java Server Pages (JSP) and Servlets
Standalone Client	Windows Forms	AWT/Swing
Distributed Components	.NET Remoting	RMI/IDL
XML	System.Xml and .NET in general is built around XML.	JAX Pack (JAXM, JAXR, JAXB, JAXP)
Messaging	Microsoft Message Queuing (MSMQ)	Java Messaging Service (JMS)
Web Services Support	Built directly into .NET and Visual Studio	Java Web Services Developer Pack (JWSDP) as well as vendor specific tools.
Enterprise	COM+	Enterprise Java Beans (EJB)

Components/Transactions		
Integration	Host Integration Server, BizTalk Server	J2EE Connector Architecture
Component Registration	Active Directory	Java Naming and Directory Interface JNDI

Tabla 3.1 Breve comparativa entre las plataformas .NET y J2EE.

3.3.1.3 Conclusión de comparativa entre .NET y Java

Muchos documentos se han escrito declarando a uno u otro el ganador basándose en rendimiento, escalabilidad, características, portabilidad e independencia de vendedor, simplicidad de uso, líneas de código, seguridad, herramientas de productividad y conectividad empresarial. Sin embargo este trabajo plantea ofrecer una solución genérica que brinde bases para la elección de una plataforma de desarrollo. Es decir en lugar de elegir una plataforma basada en la comercialización o prejuicios técnicos, se debe tomar una elección mirando la totalidad de los factores involucrados:

I. ¿Qué activos la compañía ya posee (software, hardware, middleware)?

   Entrenar desarrolladores y obtener una infraestructura puede ser costoso. Se tiene que tomar en cuanta los recursos existentes la experiencia del grupo de desarrollo así como la plataforma con el que se cuenta, beneficios y flexibilidad para futuros desarrollos.

II. Importancia de la portabilidad de la aplicación en la compañía.

   La portabilidad nunca ha sido una preocupación para Microsoft, el cual se inclina por escribir aplicaciones optimizadas para correr sobre Windows para obtener un alto rendimiento. Afirmando que la interoperabilidad es más importante que la portabilidad. Aunque existen esfuerzos open source para que .NET pueda ser montado sobre plataformas no Windows, todavía no terminan de ser proyectos. En contraparte Microsoft dice a los usuarios que las aplicaciones basadas en .Net pueden compartir información a través de Web Services, en los cuales los mensajes basados en XML son enviados vía SOAP (Simple Object Access Protocol).

III. Tener presente el nivel de experiencia

   Se tiene en gente que conoce la infraestructura. La capacitación de desarrolladores resulta costosa. Por otra parte evaluar el nivel de experiencia que se tiene en gente que conoce tanto los negocios como la implementación de la tecnología.

IV. Analizar la complejidad de las aplicaciones a construir.

Evaluar la totalidad de herramientas a utilizar en el desarrollo así que como rendimiento que debe soportar la aplicación.

V. Analizar los costos con cuidado.

Por una parte Microsoft presenta una atractiva opción, ya que el sistema operativo Windows Server viene un con un servidor de aplicaciones empotrado. Por el lado de la plataforma Java existen otras opciones, por ejemplo Hewlett-Packard Co. ofrece un servidor de aplicaciones libre con su sistema operativo. JBoss es una alternativa open-source. Aunque actualmente las opciones más elegidas son San Jose-based BEA Systems Inc.'s WebLogic o WebSphere de IBM. Sin embargo más que simplemente evaluar el costo de un servidor de aplicaciones, se tiene que tener presente el impacto económico total, beneficios y flexibilidad para futuras opciones.

En conclusión la elección de una u otra plataforma depende de las necesidades y condiciones actuales del entorno donde se realizará el desarrollo de la aplicación en cuestión.

Después analizar las dos principales opciones de desarrollo para construir aplicaciones para arquitecturas distribuidas multicapa se ha elegido para llevar a cabo el desarrollo de este sistema la plataforma Java, elección basada principalmente en la gran ventaja que ofrece la posibilidad de elegir la plataforma sobre la que se ejecutara la aplicación, así como el uso de herramientas de desarrollo robustas de uso libre con diferentes niveles de productividad que simplifican muchos de los procedimientos necesarios para desarrollar y desplegar una aplicación empresarial. Con lo cual se cubren las necesidades de plataforma para este proyecto.

3.4 Tecnologías Java Utilizadas en el Sistema.

J2EE está formada por un gran número de tecnologías diferentes y una especificación que indica como deben trabajar estas tecnologías conjuntamente. Aunque una aplicación J2EE puede consistir de tres o cuatro capas. Las aplicaciones J2EE son consideradas para ser de tres capas porque ellas son distribuidas sobre tres locaciones: maquinas cliente, la maquina servidor J2EE y las maquinas de base de datos.

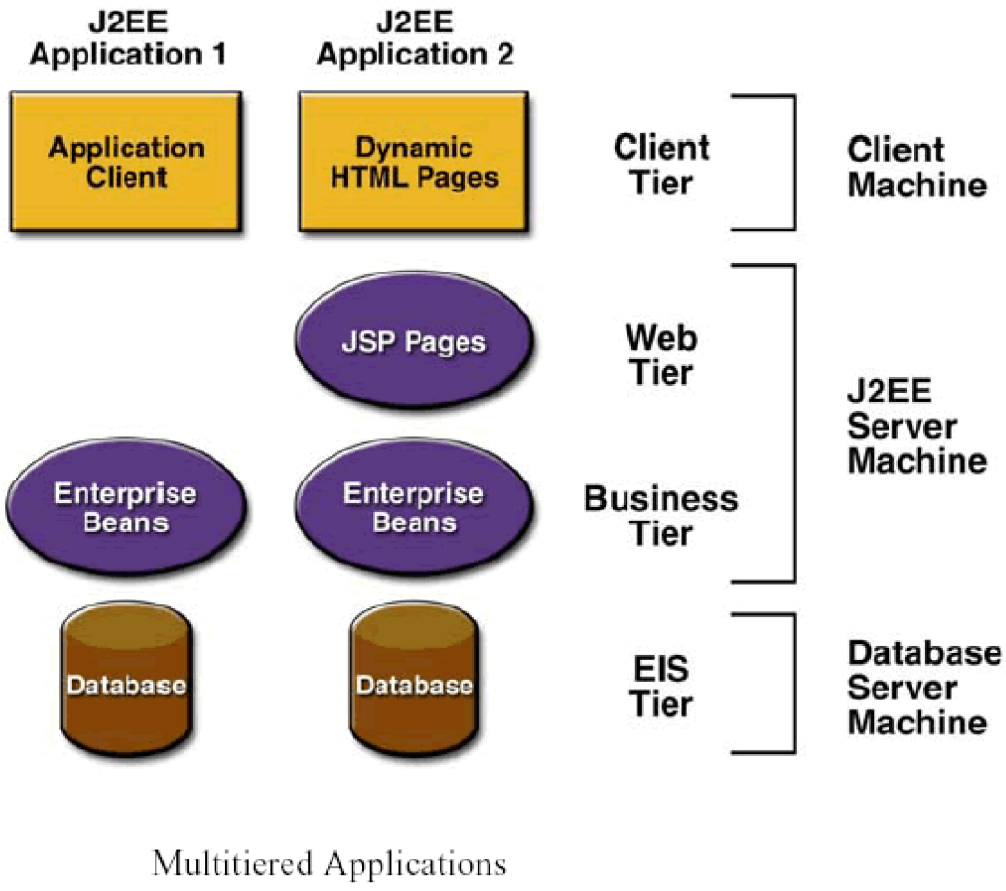


Figura 3.1 Posibles capas de una aplicación J2EE.

Las aplicaciones J2EE son una estructura de componentes.

Definición Un componente J2EE es una unidad funcional independiente de software que es ensamblada dentro de una aplicación J2EE con sus clases y archivos relacionados y que se comunica con otros componentes.

La especificación J2EE define los siguientes componentes J2EE:

- Aplicaciones clientes y applets son componentes que se ejecutan en el cliente.
- Java Servlet y Java Server Pages (JSP) son componentes que se ejecutan en el servidor.
- Enterprise Java Beans (EJB) componentes son componentes de negocio que se ejecutan en el servidor.

Los componentes son escritos en el lenguaje de programación Java. La diferencia entre los componentes J2EE y las clases estándar Java es que los componentes J2EE son ensamblados dentro de un contenedor J2EE, son verificados para ser

construidos correctamente bajo la especificación J2EE, desplegados para su ejecución y administrados por el servidor J2EE.

3.4.1 Clientes J2EE

Un cliente J2EE puede ser un cliente web o una aplicación cliente.

- Clientes Web: Un cliente Web consiste de dos partes (1) una pagina web conteniendo varios tipos de lenguaje de marcado (XTML, XML, etc.), los cuales son generados por componentes web que se ejecutan en la capa web, y (2) un web browser, el cual dibuja las paginas recibidas de el servidor. Un cliente Web es a veces llamado un cliente delgado. Los clientes delgados usualmente no requieren realizar consultas a la base de datos, ejecutar complejas reglas de negocio. Cuando se usa un cliente delgado las operaciones complejas son encargadas a enterprise beans los cuales se ejecutan en el servidor J2EE, donde se puede delegar la seguridad, velocidad de servicios y la confiabilidad de las tecnologías del lado del servidor.

- Applets: Son clientes que suelen formar parte de la interfaz de usuario y que se ejecutan dentro de un navegador Web. No es parte integrante de J2EE sino de J2SE, pero se integra con las tecnologías propuestas del J2EE en cuanto a contenidos centralizados se refiere.

- Clientes Aplicación: Un cliente aplicación se ejecuta sobre una máquina cliente y provee una manera para los usuarios puedan manejar tareas que requieren un interfaz gráfica rica. Típicamente es una interfaz gráfica de usuario (GUI) es creada con Swing o el Abstract Window Toolkit (AWT).

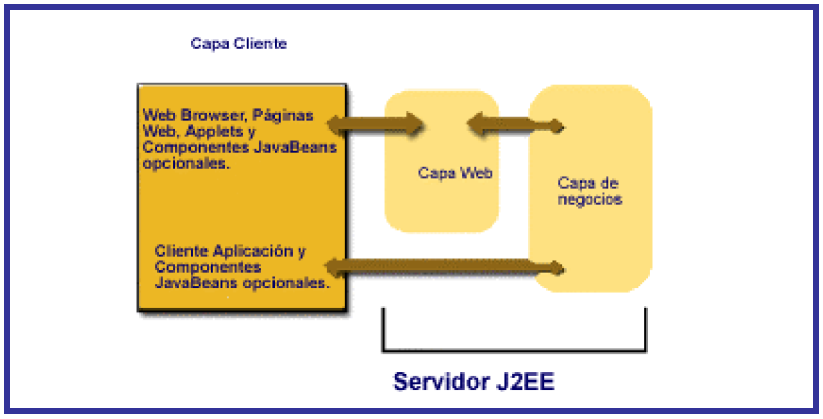


Figura 3.2 Clientes de una aplicación J2EE.

3.4.2 Componentes Web.

Los componentes Web J2EE son Servlets o páginas creadas usando la tecnología JSP. Los Servlets son clases del lenguaje de programación Java que se ejecutan en el contenedor Web de un servidor de aplicaciones. JSP es un acrónimo de Java Server Pages, es una tecnología orientada a crear página Web con programación en Java. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales y con trozos de código escrito en Java.

3.4.3 Componentes de Negocio.

La lógica de negocio que resuelve o reúne las necesidades de un caso particular como actividades bancarias, ventas o finanzas, es manejada por los Java Enterprise Beans. Los Enterprise JavaBeans (EJB) o simplemente Enterprise Beans. Son componentes que se ejecutan en un servidor J2EE, concretamente en el contenedor EJB del mismo, y que contienen las reglas de negocio. Existen tres tipos de enterprise beans: session beans, Entity Beans y message-Driven Beans.

SESSION BEANS: Gestionan el flujo de la información en el servidor. Generalmente sirven a los clientes como una fachada de los servicios proporcionados por otros componentes disponibles en el servidor. Puede haber dos tipos:

- con estado (stateful). Los beans de sesión con estado son objetos distribuidos que poseen un estado. El estado no es persistente, pero el acceso al bean se limita a un solo cliente.
- sin estado (stateless). Los beans de sesión sin estado son objetos distribuidos que carecen de estado asociado permitiendo por tanto que se los acceda concurrentemente. No se garantiza que los contenidos de las variables de instancia se conserven entre llamadas al método.

ENTITY BEANS: Un Entity Bean representa información almacenada en una línea de la base de datos. Su objetivo es encapsular los objetos del lado del servidor que almacenan los datos. Los EJBs de entidad presentan la característica fundamental de la persistencia:

- Persistencia gestionada por el contenedor (CMP): el contenedor se encarga de almacenar y recuperar los datos del objeto de entidad mediante un mapeado en una tabla de una base de datos.
- Persistencia gestionada por el bean (BMP): el propio objeto entidad se encarga, mediante una base de datos u otro mecanismo, de almacenar y recuperar los datos a los que se refiere, por lo cual, la responsabilidad de implementar los mecanismos de persistencia es del programador.

MESSAGE-DRIVEN BEAN

Los únicos beans con funcionamiento asíncrono. Usando el Java Messaging System (JMS), se suscriben a un tópico (topic) o a una cola (queue) y se activan al recibir un mensaje dirigido a dicho tópico o cola. No requieren de su instanciación por parte del cliente.

- Actúa como un listener para el Servicio API de Mensajes de JAVA, procesando los mensajes de manera asíncrona.
- Permite a las aplicaciones de J2EE procesar mensajes de manera asíncrona.
- Normalmente actúa como un listener de mensajes, similar a un listener de eventos.
- Los mensajes pueden ser enviados desde cualquier componente de J2EE.

3.4.4 Capa de Información

Definición La capa de sistema de información de la empresa maneja el software de sistemas de información de la empresa, Enterprise Information Systems o EIS e incluye sistemas de infraestructura empresarial tales como los sistemas de información integrados, Enterprise Resource Planning, o ERP, sistemas de bases de datos, etc.

Contenedores J2EE

Definición Un contenedor es una interfaz entre un componente y el sistema de inferior nivel que da soporte al componente. Mientras que un contenedor se encarga de la persistencia de los datos, la gestión de recursos, la seguridad, los hilos y otros servicios a nivel de sistema para los componentes asociados a él, estos son los responsables de implementar la lógica de negocio.

La anterior definición significa que durante el desarrollo de la aplicación el programador puede concentrar todos sus esfuerzos en codificar las reglas de negocio sin tener que preocuparse de los servicios del sistema. Comparativamente, la relación entre un componente y un contenedor es muy similar a la que existe entre un programa y el sistema operativo; en este caso, el sistema operativo proporciona al programa servicios, por ejemplo, de E/S, no es necesario modificar el programa, basta con reconfigurar el sistema operativo.

Antes de que un componente Web, EJB, o aplicación cliente pueda ser ejecutado, debe ser ensamblado en una aplicación J2EE y desplegado en su contenedor.

3.4.5 Servicios J2EE

Los Servlets, los JSP y los EJB son componentes clave de J2EE. Además de ellos, J2EE incluye otros servicios como:

- ***COMPATIBILIDAD CON CORBA.*** Permite a las aplicaciones Java comunicarse con cualquier sistema empresarial compatible con esta tecnología.

- ***JAVAMAIL.*** Permite a una aplicación J2EE enviar y recibir mensajes de correo electrónico.

- ***JAVA MESSAGE SERVICE (JMS)*** Servicio de mensajería, permite a los clientes (aplicaciones y componentes) comunicarse entre sí para solicitar y entregar servicios.

- ***JAVA NAMING AND DIRECTORY INTERFACE (JNDI)*** Interfaz de directorios y nombres de Java. Los objetos se pueden buscar en distintos servidores. Un servicio de nombres se utiliza para asociar el nombre de un objeto con su ubicación y de esta forma permitir a una aplicación obtener acceso al objeto mediante su nombre.

- ***JAVA TRANSACTION API (JTA).*** API Java para transacciones. Permite a los desarrolladores incluir código para gestionar las transacciones dentro de los componentes.

- ***JAVA DATABASE CONNECTIVITY (JDBC).*** Conexión con bases de datos desde Java. Permite a una aplicación conectarse con cualquier sistema gestor de bases de datos.

- ***DEPLOYTOOL.*** Una aplicación J2EE empaquetada, incluye un conjunto de descriptores en formato XML para instalar la aplicación y sus componentes. Para facilitar el proceso de empaquetado, J2EE proporciona la herramienta Deploytool que se encarga de todo, incluso, de escribir los descriptores a los que nos hemos referido.

4. Implementación del Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés.

Este capítulo esta dedicado a mostrar y explicar la solución a nuestro problema planteado. La infraestructura y los componentes de negocio que fueron identificados y diseñados previamente. En esta parte del desarrollo del sistema el diseño se convierte a código.

4 Implementación del sistema

4.1 Capas del Sistema

Para realizar el sistema planteado se utilizarán patrones de diseño. Una vez identificados los componentes de negocio involucrados en el sistema es momento de montar la infraestructura necesaria para la implementación del sistema propuesto sobre la base del análisis realizado previamente así como la arquitectura elegida.

Se ha identificado previamente que el sistema implementado bajo una arquitectura J2EE se puede dividir en varias capas, para fines de este sistema se ha decidido dividirlo en las siguientes capas:

- Capa de presentación
- Capa de control
- Capa de negocios
- Capa de recursos o base de datos

Definición Una capa aplicación es formada por componentes del sistema agrupados por la funcionalidad que éstos proveen a los usuarios y otros subsistemas de la aplicación.

El número de capas de una aplicación J2EE varía según su complejidad y/o necesidades. La estructura tomada como modelo para el desarrollo de esta aplicación J2EE es la que se aprecia en la figura siguiente:

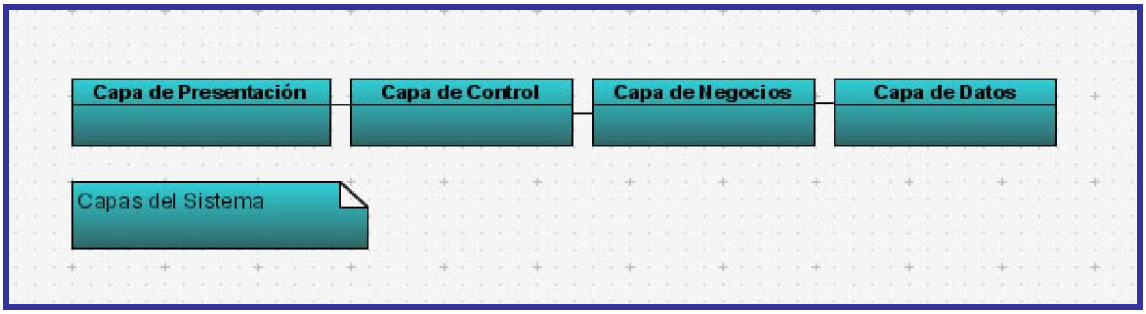


Tabla 4.1 Capas que integran el sistema SWGHAE.

CAPA CLIENTE: Es la capa donde se localiza el cliente de nuestra aplicación. Para el sistema aquí desarrollado solo se contempla un cliente browser.

CAPA DE PRESENTACIÓN: La capa de presentación contiene toda la lógica de interacción entre el usuario y la aplicación. Además, es la capa encargada de controlar la interacción entre el usuario y la lógica de negocio generando las vistas necesarias para mostrar información al usuario en forma adecuada.

CAPA DE LÓGICA DE NEGOCIO: Para la implementación de cada una de estas capas se utilizarán los patrones de Diseño de Sun Microsystems para una arquitectura empresarial.

CAPA DE DATOS: En esta capa se localizan los diferentes sistemas de información disponibles en nuestra empresa. Para nuestro caso en particular es nuestra base de datos.

En los siguientes apartados se desarrollará cada una de estas capas aplicando los patrones de diseño de Sun Microsystems.

4.2 Patrones de diseño

Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso.

Definición Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

Los diseñadores de software extendieron la idea de patrones de diseño al proceso de desarrollo de software. Debido a las características que proporcionaron los lenguajes orientados a objetos (como herencia, abstracción y encapsulamiento) les permitieron relacionar entidades de los lenguajes de programación a entidades del mundo real fácilmente, los diseñadores empezaron a aplicar esas características para crear soluciones comunes y reutilizables para problemas frecuentes que exhibían patrones similares.

Fue por los años 1994, que apareció el libro "***Design Patterns: Elements of Reusable Object Oriented Sofware***" escrito por los ahora famosos Gang of Four (GoF, que en español es la pandilla de los cuatro) formada por Erich Gamma,

Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Ellos no son los inventores ni los únicos involucrados, pero luego de la publicación de ese libro fue que empezó a difundirse con más fuerza la idea de patrones de diseño.

El grupo de GoF clasificaron los patrones en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

- **Creacionales**: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de creacion y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **Comportamiento**: Los patrones de comportamiento nos ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

En un segundo nivel, ellos clasificaron los patrones en 2 ámbitos: Clases y objetos. Es así que, tenemos 6 tipos de patrones:

CREACIONALES

- **Creacional de la Clase**
  Los patrones creacionales de Clases usan la herencia como un mecanismo para lograr la instancia de la Clase. Por ejemplo el método Factoría.

- **Creacional del objeto**
  Los patrones creacionales de objetos son más escalables y dinámicos comparados de los patrones creacionales de Clases. Por ejemplo la Factoría abstracta y el patrón Singleton.

ESTRUCTURALES

- **Estructural de la Clase**
  Los patrones estructurales de Clases usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo el patrón Adaptador (Clase).

- **Estructural de Objetos**
  Los patrones estructurales de objetos crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición del patrón estructural del objeto puede ser cambiado en tiempo de ejecución, el cual nos da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite.

COMPORTAMIENTO

- **Comportamiento de Clase**
  Los patrones de comportamiento de Clases usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo Interpreter.

- **Comportamiento de Objeto**
  Los patrones de comportamiento de objetos nos permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor.

4.3 Patrones J2EE

Con la aparición del J2EE, todo un nuevo catálogo de patrones de diseño apareció. Desde que J2EE es una arquitectura por si misma que involucra otras arquitecturas, incluyendo servlets, Java Server Pages, Enterprise JavaBeans, y más, merece su propio conjunto de patrones específicos para diferentes aplicaciones empresariales.

De acuerdo al libro "***J2EE PATTERNS BEST PRACTICES AND DESIGN STRATEGIES***", existen 5 capas en la arquitectura J2EE:

- **Cliente**
- **Presentación**
- **Negocios**
- **Integración**
- **Recurso**

El libro explica 15 patrones J2EE que están divididos en 3 de las capas: presentación, negocios e integración. El diagrama de continuación muestra la interacción entre cada uno de los patrones J2EE de Sun Microsystems.

Catálogo de patrones J2EE

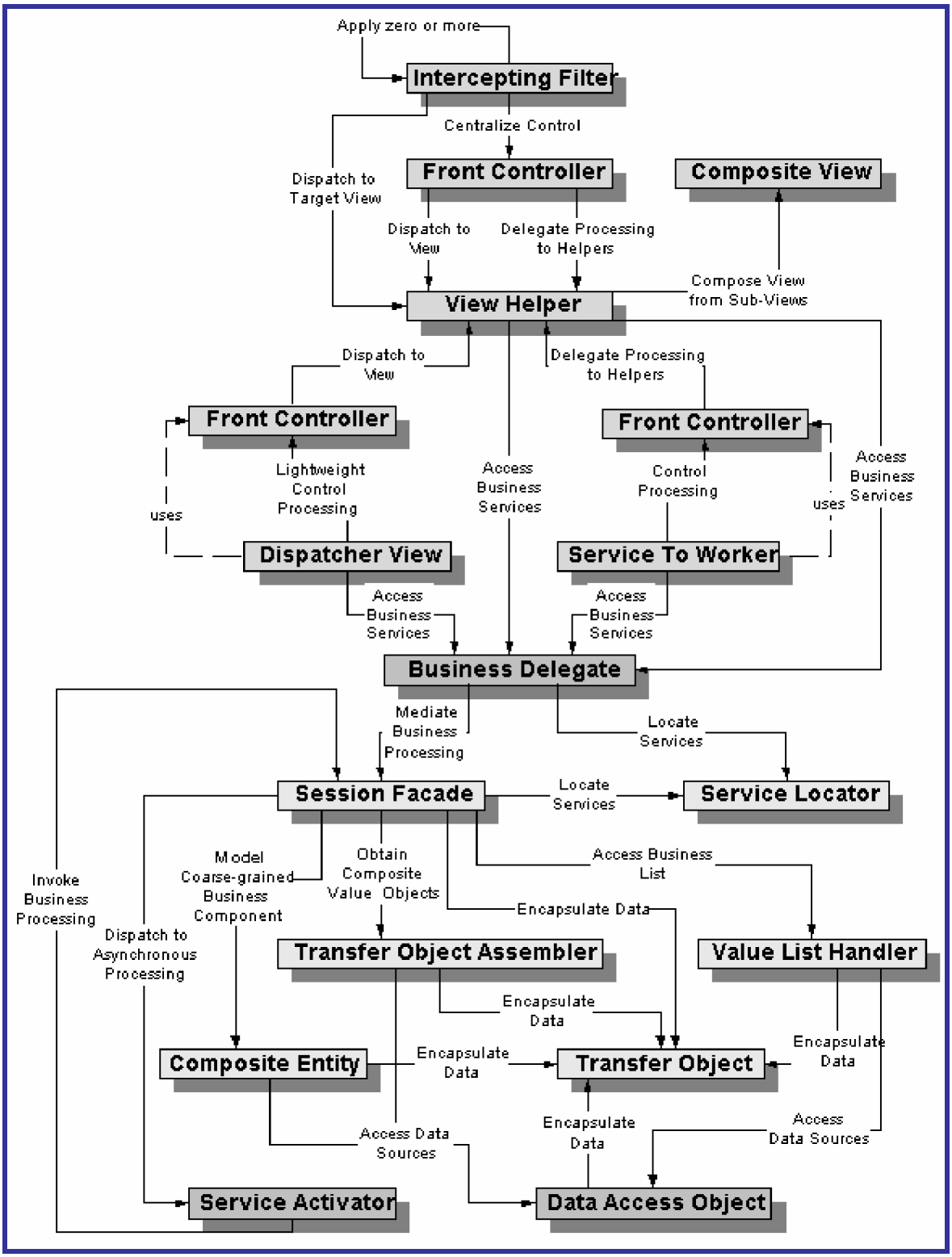


Figura 4.1 Diagrama de patrones J2EE de Sun Microsystems

No es el objetivo de este trabajo profundizar en cada uno de estos patrones, simplemente explicar que son los patrones, la importancia que tienen en el desarrollo de un proyecto.

Los siguientes cuadros muestran un breve resumen de los principales patrones de Sun Microsystems.

Capa de Presentación	
Decorating Filter / Intercepting Filter	Un objeto que está entre el cliente y los componentes Web. Este procesa las peticiones y las respuestas.
Front Controller/ Front Component	Un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en 2 diferentes objetos: el Front Controller y el Dispatcher. En ese caso, El Front Controller acepta todos los requerimientos de un cliente, y el Dispatcher direcciona los requerimientos a un manejador apropiado.
View Helper	Un objeto helper que encapsula la lógica de acceso a datos en beneficio de los componentes de la presentación. Por ejemplo, los JavaBeans pueden ser usados como patrón View Helper para las páginas JSP.
Composite view	Un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View.
Service To Worker	Es como el patrón de diseño MVC con el Controlador actuando como Front Controller pero con una cosa importante: aquí el Dispatcher (el cual es parte del Front Controller) usa View Helpers a gran escala y ayuda en el manejo de la vista.
Dispatcher View	Es como el patrón de diseño MVC con el controlador actuando como Front Controller pero con un asunto importante: aquí el Dispatcher (el cual es parte del Front Controller) no usa View Helpers y realiza muy poco trabajo en el manejo de la vista. El manejo de la vista es realizado por los mismos componentes de la Vista.

Tabla 4.2 Patrones J2EE de Sun Microsystems para la capa de presentación.

Capa de Negocios	
Business Delegate	Un objeto que reside en la capa de presentación y en beneficio de los otros componentes de la capa de presentación llama a métodos remotos en los objetos de la capa de negocios.
Value Object/ Data Transfer Object/ Replicate Object	Un objeto serializable para la transferencia de datos sobre la red.
Session Façade/ Session Entity Façade/ Distributed Façade	El uso de un bean de sesion como una fachada (facade) para encapsular la complejidad de las interacciones entre los objetos de negocio y participantes en un flujo de trabajo. El Session Facade maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes.
Aggregate Entity	Un bean de entidad que es construido o es agregado a otros beans de entidad.
Value Object Assembler	Un objeto que reside en la capa de negocios y crea Value Objets cuando es requerido.
Value List Handler/ Page-by-Page Iterator/ Paged List	Es un objeto que maneja la ejecución de consultas SQL, caché y procesamiento del resultado. Usualmente implementado como beans de sesión.
Service Locator	Consiste en utilizar un objeto Service Locator para abstraer toda la utilización JNDI y para ocultar las complejidades de la creación del contexto inicial, de búsqueda de objetos home EJB y recreación de objetos EJB. Varios clientes pueden reutilizar el objeto Service Locutor para reducir la complejidad del código, proporcionando un punto de control.

Tabla 4.3 Patrones J2EE de Sun Microsystems para la capa de negocios.

Capa de Integración	
Data Access Object Service Activator	Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.
Service Activator	Se utiliza para recibir peticiones y mensajes asíncronos de los clientes. Cuando se recibe un mensaje, el Service Activator localiza e invoca a los métodos de los componentes de negocio necesarios para cumplir la petición de forma asíncrona.

Tabla 4.4 Patrones J2EE de Sun Microsystems para la capa de integración.

Cuando se empezó a documentar los patrones J2EE, se tomó la decisión de documentarlos con un nivel de abstracción relativamente alto. Al mismo tiempo, cada patrón incluye varias estrategias que proporcionan detalles de su implementación a bajo nivel. A través de las estrategias, cada patrón documenta

una solución con varios niveles de abstracción. Se ha reconocido que se podría haber documentado algunas de estas estrategias como patrones por sí mismas. Sin embargo, se ha creído que la estructura de plantilla actual comunica más claramente la relación de las estrategias con la estructura de patrón de alto nivel en que se ha incluido. Se han anotado algunos de los problemas con respecto a la relación entre las estrategias y los patrones:

- Los patrones existen a un nivel de abstracción más alto que las estrategias.
- Los patrones incluyen implementaciones más comunes o más recomendadas que las estrategias.
- Las estrategias proporcionan un punto de extensibilidad para cada patrón.
- Los desarrolladores descubren e inventan nuevas formas de implementar patrones, produciendo nuevas estrategias para patrones conocidos ampliamente.
- Las estrategias promueven una mejor comunicación, proporcionando nombres para aspectos de bajo nivel de una solución particular.

4.4 Implementación del sistema

Para la implementación de cada una de las capas del Sistema no se utilizarán todos los patrones que se pueden aplicar a cada una de éstas, sino solamente aquellos que según los requerimientos del sistema se consideraron indispensables para dar una solución que permita tener capas desacopladas, es decir que no dependan unas de otras. Para ejemplificar como se implemento dicho sistema en cada una de las capas previamente analizadas se tomará el módulo del framework del sistema el cual nos presta los servicios básicos para ejecución de los demás módulos del sistema.

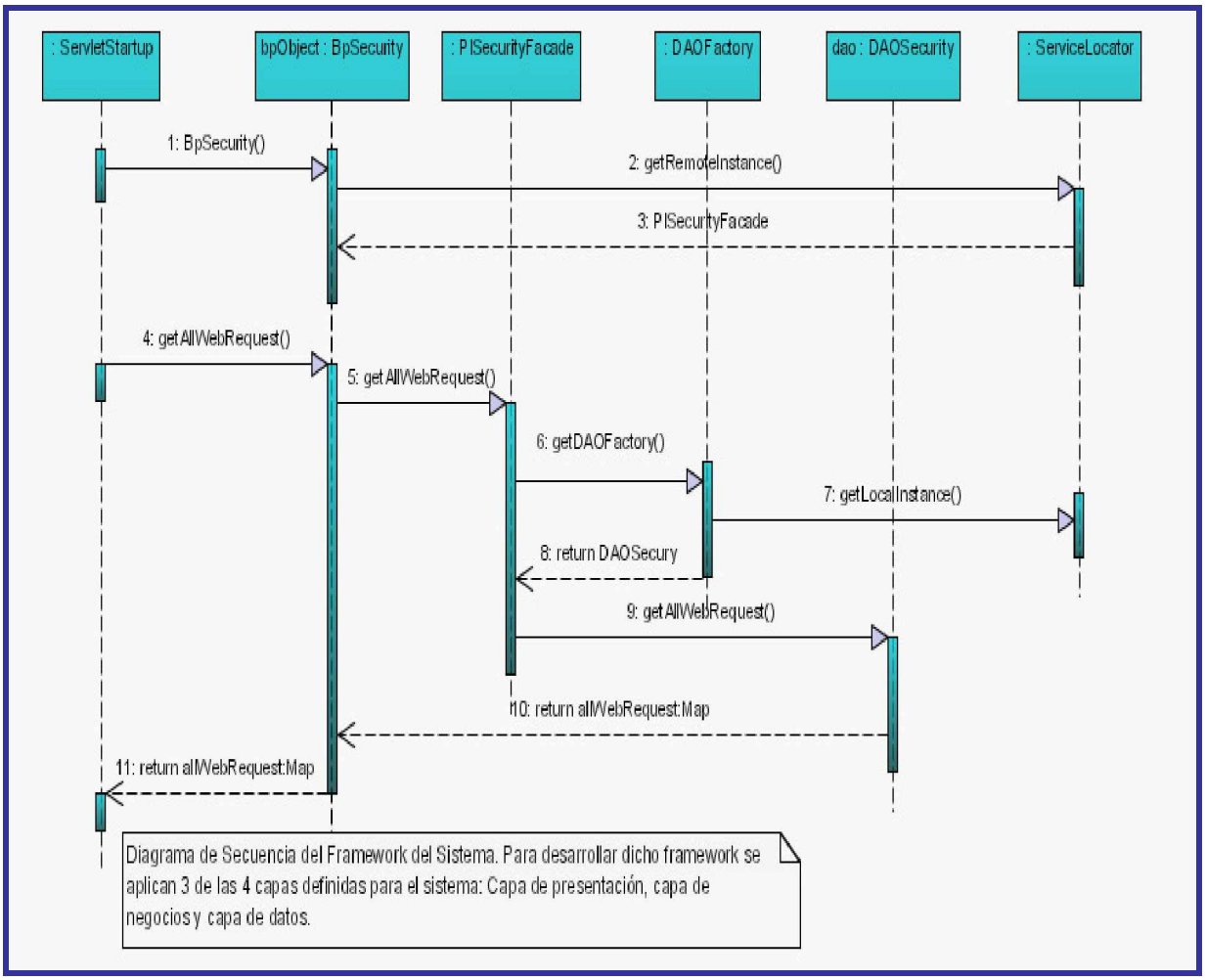


Figura 4.2 Diagrama de Secuencia del Framework del sistema.

El diagrama anterior muestra la interacción entre las clases involucradas del sistema para proporcionar el framework del sistema.

Definición Un framework es una estructura de software orientada a mejorar la productividad a través de la reutilización de código. Es un conjunto de clases creadas para apoyar la escritura de código en un contexto determinado.

Para atender a detalle a la elaboración del framework del sistema nos ocuparemos de la primera capa de la aplicación: la capa cliente.

4.4.1 La capa cliente

A la hora de crear una aplicación empresarial con J2EE nos encontraremos habitualmente con los siguientes tipos de aplicaciones cliente:

- Aplicaciones de escritorio tradicionales.
- Navegadores Web.
- Aplicaciones para pequeños dispositivos.

Los dos primeros tipos son, sin duda, los más habituales. El debate entre las ventajas y desventajas entre estos dos tipos esta fuera del alcance de este trabajo. Ya que la decisión acerca de un cliente Web fue requerimiento de este proyecto, por lo que la aplicación únicamente soportará un cliente tipo Web.

4.4.2 La capa de presentación

La capa de presentación se ocupa de los componentes responsables de la creación y el manejo de las interfaces de la aplicación con los usuarios. El siguiente cuadro resume los componentes integrantes de la capa de presentación J2EE. Todos los componentes con excepto los applets, se ejecutan en el contenedor Web del servidor.

Componentes J2EE en la capa de presentación	
Componente J2EE	Propósito
Applets	Componente Java que se descarga y ejecuta en la máquina cliente
Servlets	Componente Java invocado a través de la petición de un cliente para actuar del lado del servidor procesando y generando una respuesta.
Java Server Pages(JSPs)	Página de respuesta que permite la inclusión de código Java mezclado con contenido estático.
JavaBeans	Componente re usable utilizado en la arquitectura.
JSP custom tags	Clases que permiten que la lógica compleja pueda ser referenciada a través de tags dentro un JSP.
Filtres	Componente del lado del servidor que intercepta las peticiones para pre y post procesarlas.

Tabla 4.5 Componentes de la capa de presentación de una aplicación J2EE.

La capa de presentación puede localizarse en una aplicación de escritorio o dentro de un contenedor Web. Independientemente de su localización, la capa de

presentación es la encargada de controlar la generación del contenido que se mostrará al usuario final.

La capa de presentación es una de las más complejas ya que engloba muchas tareas de diferente índole. Es por ello que hemos de ser cuidadosos en su creación para no penalizar en exceso factores como la extensibilidad. Esta capa ha de ser de fácil mantenimiento y extensible ya que es la más sensible al cambio (la lógica de negocio de la empresa no es algo que cambie tan frecuentemente) y por lo tanto es muy importante que pequeñas modificaciones y añadidos no interfieran en el funcionamiento de las aplicaciones existentes.

Nos encontremos ante una aplicación de escritorio o una aplicación Web, el patrón más común a la hora de implementar la capa de presentación es, sin duda alguna, el MVC (Model View Controller), que permite una separación casi perfecta entre lo que se conoce como modelo (lógica de negocio), el controlador y la vista.

Los componentes del Servidor deben ser diseñados para colaborar unos con otros adoptando el patrón de diseño de software MVC. Las ventajas de este patrón son múltiples. Entre algunas de las más tratadas en la literatura que trata sobre este tema destacan:

- **Separación de responsabilidades.** Cada una de las partes de este patrón se encarga de tareas muy diferentes que se encuentran aisladas y no interfieren entre ellas.

- **Múltiples vistas**. El escaso acoplamiento entre las partes de este patrón permite que se puedan generar fácilmente múltiples vistas para un mismo modelo. Cada vez que el modelo se ve modificado todas las vistas se actualizan automáticamente.

- **Tests de funcionalidad**. La separación entre vista y modelo permite implementar de manera muy sencilla conjuntos de tests para probar el funcionamiento del modelo y del controlador utilizando una vista mucho más sencilla que la que tendrá la aplicación final. Una vez que se comprueba el correcto funcionamiento del modelo y el controlador se puede desarrollar una vista mucho más robusta.

Dado que las aplicaciones Web se ejecutan en un servidor, es necesario prestar atención a temas de sincronización de múltiples usuarios, prestación de calidad de servicio, tener mayor control transaccional, etc. En una aplicación de escritorio, estos factores no son tan importantes ya que la lógica de presentación se localiza en el mismo cliente. En las aplicaciones Web, son múltiples los clientes que acceden simultáneamente a la lógica de presentación y por lo tanto es necesario prestar especial atención a todos esos factores.

A continuación se atenderán cada uno de los patrones aplicados para el desarrollo de este sistema.

4.4.2.1 Front Controller

La primera decisión, y una de las más importantes a la hora de desarrollar una aplicación Web, es decidir entre si se va a utilizar un framework Web o si se van a implementar manualmente todas sus funcionalidades.

Definición Un framework Web, es un conjunto librerías que proporcionan automáticamente una serie de servicios al desarrollador, de modo que éste no tenga que preocuparse de la implementación de los mismos. La mayor parte de estos frameworks implementan el modelo MVC y basan la creación de aplicaciones en dicho modelo.

Para la realización de este sistema se tomó la decisión de implementar manualmente las funcionalidades de un framework utilizando patrones de diseño para la elaboración de este proyecto debido a la experiencia que se tiene con el lenguaje de programación Java. Para tal fin se utilizó el patrón Front Controller, el cual será el encargado de controlar la generación del contenido que se mostrará en el cliente Web. Esta generación comprende la comunicación con la capa de lógica de negocio para obtener los datos necesarios, el control del flujo de pantallas, el control de la interacción entre diferentes componentes, ensamblar las diferentes vistas que puedan formar una pantalla y limitar la información con base a perfiles de usuario y reglas de autorización, entre otras tareas.

4.4.2.1.1 Introducción y contexto de la aplicación del Patrón Front Controller

El mecanismo de manejo de peticiones de la capa de presentación debe controlar y coordinar el procesamiento de todos los usuarios a través de varias peticiones. Dichos mecanismos de control se pueden manejar de una forma centralizada o descentralizada. Para resolver tal situación el sistema requiere un punto de acceso centralizado para que el manejo de peticiones de la capa de presentación soporte la integración de los servicios del sistema, recuperación de contenidos, control de vistas, y navegación. Cuando el usuario accede a la vista directamente sin pasar un mecanismo centralizado, sin embargo podrían ocurrir dos problemas:

- Se requiere que cada vista proporcione sus propios servicios del sistema, lo que normalmente resulta en duplicación de código.
- La vista de navegación se deja a los visores. Esto podría resultar en una mezcla de contenidos y navegación.

Además, el control distribuido es más difícil de mantener, ya que los cambios se tienen que realizar en numerosos lugares.

CAUSAS:

- El procesamiento de servicios del sistema comunes se completa por cada petición. Por ejemplo, el servicio de seguridad completa los chequeos de autentificación y autorización.
- La lógica se maneja mejor en una localización central en lugar de estar replicada dentro de varias vistas.
- Existen puntos de decisión con respecto a la recuperación y manipulación de los datos.
- Se utilizan varias vistas para responder a peticiones de negocio similares.
- Puede ser muy útil un punto de contacto centralizado para manejar una petición, por ejemplo, para controlar y grabar el camino de un usuario por el sitio.
- Los servicios del sistema y la lógica de control de vistas son relativamente sofisticados.

Definición El patrón Front Crontroller propone usar un controlador como el punto inicial de contacto para manejar las peticiones. El controlador maneja el control de peticiones, incluyendo la invocación de los servicios de seguridad como la autentificación y autorización, delegar el procesamiento de negocio, controlar la elección de una vista apropiada, el manejo de errores, y el control de la selección de estrategias de creación de contenido.

El controlador proporciona un punto de entrada centralizado que controla y maneja las peticiones Web. Centralizando los puntos de decisión y control, el controlador también ayuda a reducir la cantidad de código Java, llamadas a scriptles, embebidos en la página Java Server Pages (JSP). Centralizando el control y reduciendo la lógica de negocios en la vista es posible reutilizar el código entre peticiones. Es una aproximación preferible a la alternativa de embeber código en varias vistas porque esta aproximación trata con entornos más propensos a errores, y de reutilización del tipo copiar y pegar.

4.4.2.1.2 Aplicación del patrón Front Controller en el sistema.

Típicamente, un controlador se coordina con un componente **dispatcher**. Los **dispatchers** son los responsables del control de la vista y de la navegación. Así, un **dispatcher** elige la siguiente vista por el usuario y dirige el control al recurso. Los **dispatchers** podrían encapsularse directamente dentro del controlador o se puede extraer en un componente separado. En el caso del sistema el **dispatcher** esta integrado dentro del controlador.

```
196     /**
197      * Invoca la ejecución del handler asociado a la peticion en curso
198      * @param webReq La peticion que se esta procesando
199      * @param request la peticion http
200      * @return
201      */
202     private int executeHandler(VoWebRequest webReq, HttpServletRequest request)
203             throws ServletException {
204         String className = WebConstants.PACKAGE_HANDLER + webReq.getIdHandler();
205         logger.info("getting handler " + className);
206         try {
207             HandlerMaster handler = (HandlerMaster) getClass().getClassLoader()
208                     .loadClass(className).newInstance();
209             return handler.doProcess(request, webReq.getHandSwitchValue());
210         } catch (InstantiationException e) {
211             logger.severe(LogMessage.getStackMessage(e));
212             throw new ServletException("CANNOT INSTANCIATE THE  HANDLER:"
213                     + webReq.getIdHandler());
214         } catch (IllegalAccessException e) {
215             logger.severe(LogMessage.getStackMessage(e));
216             throw new ServletException("CANNOT ACCESS THE  HANDLER:"
217                     + webReq.getIdHandler());
218         } catch (ClassNotFoundException e) {
219             logger.severe(LogMessage.getStackMessage(e));
220             throw new ServletException(
221                     "CANNOT INSTANCIATE THE  HANDLER, CLASS NOT FOUND:"
222                             + webReq.getIdHandler());
223         }
224     }
```

Figura 4.3 Segmento de código fuente de la implementación del patrón Front Controller.

En este caso el Front Controller utiliza la llamada a diversos manejadores (“Clases Handlers”) para utilizar los servicios de la capa de negocios de acuerdo a la petición en curso.

```
133             //obtiene la peticion que le corresponde al comando en curso
134             webReq = WebRequestInfo.getRequest(pathInfo);
135             if (webReq == null) {
136                 throw new ServletException("Web request not found in pool for " + pathInfo);
137             }
138             opcionesPeticion = webReq.getReqOpcionResultado();
139             handResultado = executeHandler(webReq, request);
140             switch (handResultado) {
141             case WebConstants.HR_CORRECT:
142                 webReq.setIdPagina(opcionesPeticion.get(new Integer(handResultado)).toString());
143                 nextScreen = contextPath + webReq.getIdPagina();
144                 break;
145             case WebConstants.HR_CORRECT_BY_OPTION:
146                 if (webReq.getReqOpcionResultado() == null) {
147                     throw new ServletException(
148                             "Web Request hasn't optional results: "+ webReq.getIdPeticion());
149                 }
150                 objAttr = webReq.getReqOpcionResultado().get(
151                         request.getAttribute(WebConstants.HR_OPCION_VALUE));
152                 if (objAttr == null) {
153                     throw new ServletException("Web Request not found for handler opcion: "
154                             + request.getAttribute(WebConstants.HR_OPCION_VALUE));
155                 }
156                 nextScreen = contextPath + objAttr.toString();
157                 break;
158             case WebConstants.HR_ERROR:
159                 nextScreen = contextPath + ERROR_PAGE;
160                 break;
161             default:
162                 throw new ServletException(" Cannot get next screen, Invalid Handler result: " + handResultado);
163             }
```

Figura 4.4 Segmento del código fuente de la implementación del patrón Front Controller.

Aunque el patrón **Front Controller** sugiere la centralización del manejo de peticiones, no limita el número de manejadores en el sistema, como lo hace **Singleton**. Una aplicación podría utilizar varios controladores en un sistema, cada uno mapeado a un conjunto de servicios distintos. En este caso en esta aplicación todas las peticiones están asociadas a un solo controlador, como podemos observar el siguiente fragmento del archivo web.xml

```
24      <servlet-mapping>
25              <servlet-name>FrontController</servlet-name>
26              <url-pattern>/command/*</url-pattern>
27      </servlet-mapping>
28      <servlet>
29          <servlet-name>FrontController</servlet-name>
30          <servlet-class>
31              etg.framework.web.controller.FrontController
32          </servlet-class>
33                  <init-param>
34              <param-name>ERROR_PAGE</param-name>
35              <param-value>admin/error.jsp</param-value>
36          </init-param>
37          <init-param>
38              <param-name>PUBLIC_MAIN_PAGE</param-name>
39              <param-value>public/principal.jsp</param-value>
40          </init-param>
41          <init-param>
42              <param-name>ADMIN_MAIN_PAGE</param-name>
43              <param-value>admin/principal.jsp</param-value>
44          </init-param>
```

Tabla 4.6 Segmento del archivo web.xml del sistema.

La siguiente figura representa el diagrama de la secuencia del patrón Front Controller. Muestra como el controlador maneja una petición:

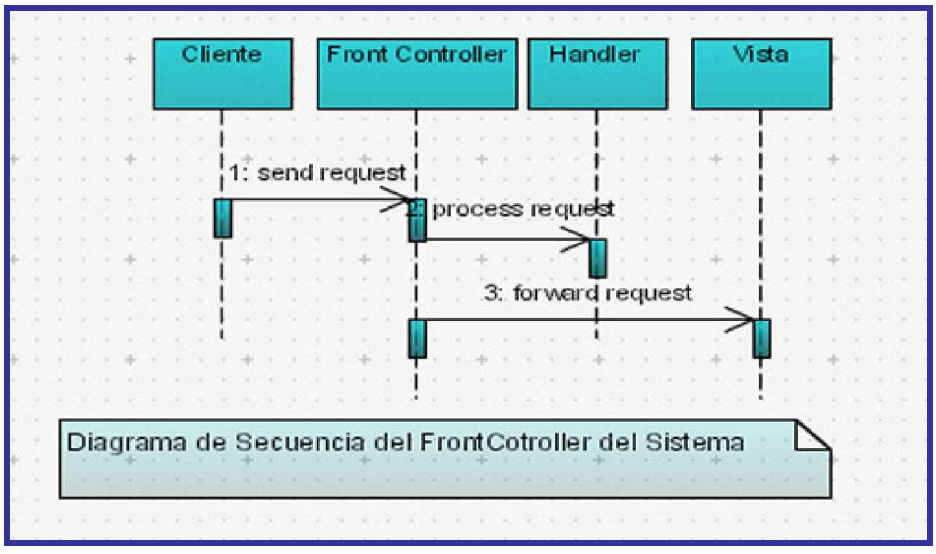


Figura 4.5 Diagrama de secuencia del patrón Front Controller.

CONTROLADOR (FRONT CONTROLLER): El controlador es el punto de contacto inicial para manejar todas las peticiones en el sistema. El controlador podría delegar a un Handler para completar información, una vez completada la información el Controller también es el responsable del manejo de la vista y de la navegación, controlando la elección de la siguiente vista que se le presentará al usuario, y proporcionando el mecanismo para dirigir el control a ese recurso con base a la respuesta del Handler. El Controller utiliza un objeto RequestDispatcher (Soportado en la especificación Servlet).

HANDLER: El Puede completar información necesaria para atender la petición a la vez indica cual será la siguiente vista a presentar comunicándoselo al Controller.

VISTA: Una Vista representa y muestra información al cliente. La vista recupera información desde el modelo. Dado que es sistema se basa en un sistema Web nuestro cliente es un browser y nuestras vistas son páginas HTML.

4.4.3 Capa de Negocios

Dentro de la capa de negocios del sistema se utilizarán los patrones Service Locator, Business Delegate y Value Object**.**

4.4.3.1 Service Locator

Los clientes J2EE interactúan con componentes de servicio, como componentes JavaBeans Enterprise (EJB) y Java Message Service (JMS), que proporcionan servicios de negocio y capacidades de persistencia. Para interactuar con estos componentes, los clientes deben localizar el componente de servicio (referido como una operación de búsqueda) o crear un nuevo componente. Por ejemplo, un cliente EJB debe localizar el objeto home del bean enterprise, que el cliente puede utilizar para encontrar un objeto o para crear uno o más beans enterprise. De forma similar, un cliente JMS primero debe localizar la Factoría de Conexiones JMS para obtener una Conexión JMS o una Sesión JMS.

Todos los clientes de aplicaciones J2EE utilizan la facilidad JNDI para buscar y crear componentes EJB o JMS. El API JNDI permite a los clientes obtener un objeto Contexto Inicial que contiene el nombre del componente a uniones de objetos. El cliente empieza obteniendo el contexto inicial para un objeto home de un bean. El contexto inicial permanece válido mientras sea válida la sesión del cliente. El cliente proporciona el nombre registrado en JNDI del objeto requerido para obtener una referencia a un objeto administrado. En el contexto de una aplicación EJB, un objeto administrado típico es un objeto home de un bean enterprise. Para aplicaciones JMS, el objeto administrado puede ser una Factoría de Conexiones JMS (para un Topic o una Queue) o un Destino JMS (un Topic o una Queue).

Por eso, localizar un objeto servicio administrado JNDI es una tarea común para todos los clientes que necesiten acceder al objeto de servicio. Por ejemplo, es fácil

ver que muchos tipos de clientes utilizan repetidamente el servicio JNDI, y que el código JNDI aparece varias veces en esos clientes. Esto resulta en una duplicación de código innecesaria en los clientes que necesitan buscar servicios. También, crear un objeto de contexto JNDI inicial y realizar una búsqueda para objeto home EJB utiliza muchos recursos. Si varios clientes requieren de forma repetitiva el mismo objeto home de un bean, dicha duplicación de esfuerzos puede impactar de forma negativa en el rendimiento de la aplicación. Examinemos el proceso de búsqueda y creación de varios componentes J2EE.

1. La búsqueda y creación de Beans Enterprise trata sobre lo siguiente:

- Una correcta configuración del entorno JNDI para que se conecte con el servicio de nombrado y directorio utilizado por la aplicación. Esta configuración proporciona la localización del servicio de nombrado y las credenciales de autentificaciones necesarias para acceder a ese servicio.
- Entonces el servicio JNDI puede proporcionar al cliente un contexto inicial que actúa como un almacén para las uniones de componentes nombre-a-objeto. El cliente solicita a este contexto inicial que busque el objeto EJBHome del bean enterprise requerido proporcionando un nombre JNDI para ese objeto EJBHome.
- Encontrar el objeto EJBHome utilizando el contexto de búsqueda inicial
- Después de obtener el objeto EJBHome, crea, elimina o encuentra el bean enterprise utilizando los métodos create o find (solo para beans de entidad) del objeto EJBHome.

2. La búsqueda y creación de componentes JMS (Topic, Queue, QueueConnection, QueueSession, TopicConnection, TopicSession, etc.) implica los siguientes pasos. Cabe mencionar que en estos pasos, Topic se refiere al modelo de mensajería publica/subscribe y que Queue se refiere al modelo de mensajería punto a punto.

- Una correcta configuración del entorno JNDI para que se conecte con el servicio de nombrado y directorio utilizado por la aplicación. Esta configuración proporciona la localización del servicio de nombrado y las credenciales de autentificaciones necesarias para acceder a ese servicio.
- Obtener el contexto inicial para el proveedor de servicio JMS desde el servicio de nombrado JNDI.
- Utilizar el contexto inicial para obtener un Topic o un Queue suministrando el nombre JNDI para ese Topic o Queue. Ambos son objetos JMSDestination.
- Utilizar el contexto inicial para obtener un TopicConnectionFactory o un QueueConnectionFactory suministrando el nombre JNDI para la factoría adecuada.
- Usar el TopicConnectionFactory para obtener un TopicConnection o el QueueConnectionFactory para obtener un QueueConnection.

- Utilizar el TopicConnection para obtener un TopicSession o el QueueConnection para obtener un QueueSession.
- Utilizar el TopicSession para obtener un TopicSubscriber o un TopicPublisher para el Topic Requerido. Utilizar el QueueSession para obtener un QueueReceiver o un QueueSender para el Queue requerido.

El proceso de búsqueda y creación de componentes implica una implementación de una factoria de contextos suministrada por un vendedor. Esto introduce dependencias del vendedor en los clientes de la aplicación que necesitan utilizar la facilidad de búsqueda JNDI para localizar beans enterprise y componentes JMS.

Definición Un objeto Service Locator se utiliza para abstraer toda la utilización JNDI y para ocultar las complejidades de la creación del contexto inicial, de búsqueda de objetos home EJB y de re-creación de objetos EJB. Varios clientes pueden reutilizar el objeto Service Locator para reducir la complejidad del código, proporcionando un punto de control, y mejorando el rendimiento proporcionando facilidades de caché.

Al utilizar el patrón Service Locutor se reduce la complejidad del cliente que resulta de las dependencias del cliente y de la necesidad de realizar los procesos de búsqueda y creación, que consumen muchos recursos. Para eliminar estos problemas, este patrón proporciona un mecanismo para abstraer todas las dependencias y detalles de red dentro del Service Locator.

Como resultado de la aplicación del patrón DAO tenemos los siguientes puntos:

- Flexibilidad en la instalación/configuración de una aplicación.
- Independencia del vendedor de la fuente de datos.
- Independencia del recurso (base de datos relacional, base de datos orientado a objetos, ficheros planos, etc.).
- Reduce la complejidad de la implementación de la lógica de negocios (Session Facade)
- Más complejidad en el diseño, debido principalmente al conjunto de clases que colaboran en el DAO y en forma general, cada DAO es utilizado para modelar una tabla del modelo relacional.

4.4.3.1.1 Implementación en el sistema del Patrón Service Locator

El siguiente diagrama de secuencia muestra como se implemento el patrón Service Locator en el sistema el cual posteriormente veremos que es usado por el patrón Business Delegate para lograr un desacoplamiento entre capas.

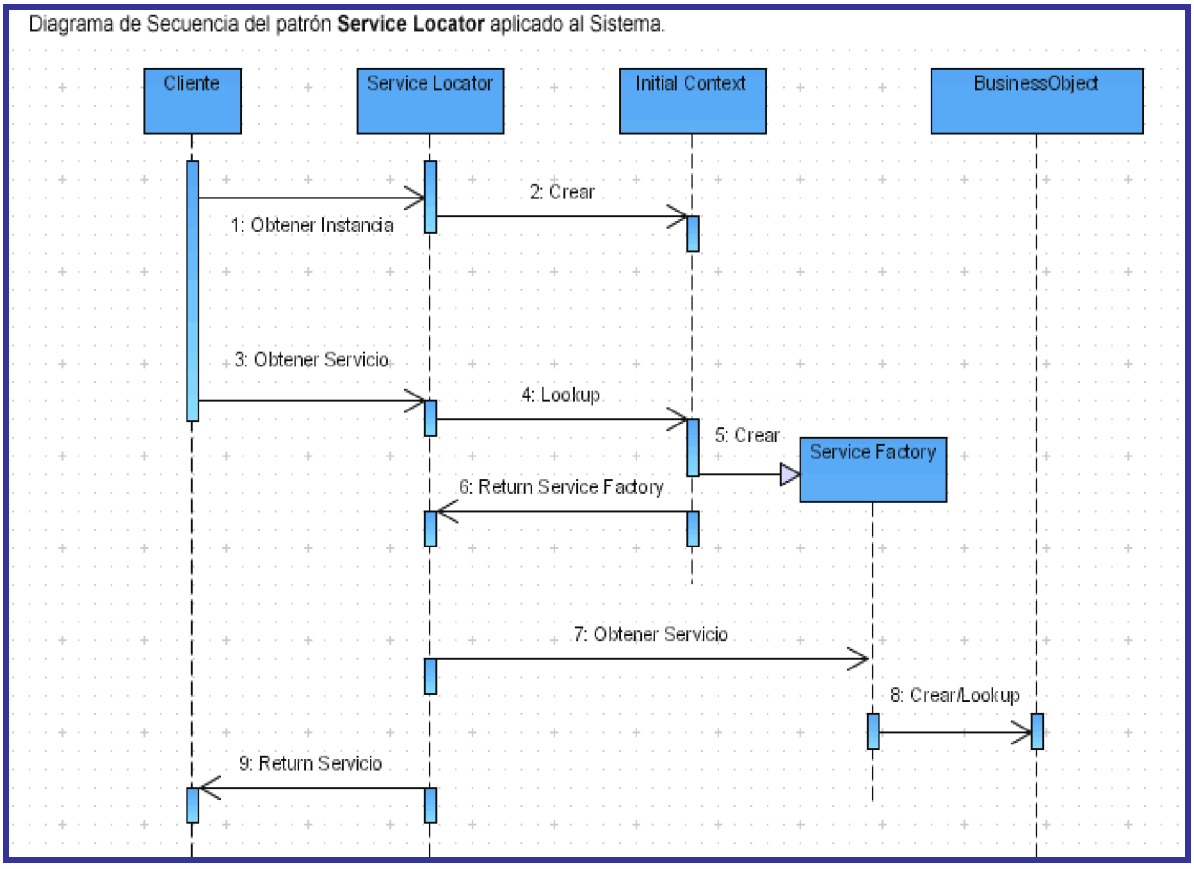


Figura 4.6 Diagrama de Secuencia de la aplicación del patrón Service Locator.

El siguiente fragmento de código de la clase ServiceLocator implementada para el sistema muestra como el constructor ServiceLocator crea una instancia del objeto context del tipo InitialContext ya sea para un contexto remoto o local dependiendo del parámetro recibido. Encapsulado en un solo método genérico la construcción del context adecuado.

De esta forma solo se tendrá que utilizar una llamada al método getLocalInstance para obtener una instancia de un contexto local

```
/**
 * Inicializa el contexto para la generación de lookups al servidor.
 * De requerirse,  inicializa el pool de conexiones para la optimización de
 * las búsquedas remotas.
 *
 * @param contextType            El tipo de contexto: local o remoto
 * @throws ServiceLocatorException  Si se genera algun severe al obtener los objetos Home
 */
private ServiceLocator(int contextType) throws ServiceLocatorException {
   if (poolLookUp == null)
      poolLookUp = new HashMap();
   try {
      switch (contextType) {
      case REMOTE_CONTEXT:
         context = new InitialContext();
         break;
      case LOCAL_CONTEXT:
         context = new InitialContext();
         break;
      }
   } catch (NamingException e) {
      throw new ServiceLocatorException("at getting initial context "
            + e.toString(), e);
   }
}

/**
 * Obtiene una instancia de esta clase para accesos locales.
 * @return                      el objeto <code>ServiceLocator</code>
 * @throws ServiceLocatorException  Si ocurre un severe al instanciar la clase.
 */
public static ServiceLocator getLocalInstance()
      throws ServiceLocatorException {
   if (service == null) {
      service = new ServiceLocator(LOCAL_CONTEXT);
      return service;
   } else
      return service;
}
```

Figura 4.7 Segmento del código fuente de la aplicación del patrón Service Locator.

La misma clase Service Locator puede implementar servicios para su uso común de la aplicación como es la conexión a la base de datos.

```
/**
 * Obtiene una conexión del pool de conexiones del servidor.
 * Optimiza  la búsqueda del datasource al servidor mediante
 * el atributo estático <code>ds</code>
 * @return Una nueva conexión  para  base de datos.
 * @throws ServiceLocatorException  Si ocurre algún severe de conexion.
 */
public Connection getDataBaseConnection() throws ServiceLocatorException {
   Connection con = null;
   try {
      if (ds == null) { //la constante JNDIKeys.DATA_SOURCE_JNDI_NAME es igual a jdbc/projectTesisDS
         ds = (DataSource) context.lookup(JNDIKeys.DATA_SOURCE_JNDI_NAME);
         logger.info("Getting new DB connection (DATASOURCE)...");
      }

      con = ds.getConnection();
      openConnection++;
      logger.info("Getting connection from pool.. C==" + openConnection);
      return con;
   } catch (NamingException e) {
      throw new ServiceLocatorException(" at getting JNDI DS Object"
            + JNDIKeys.DATA_SOURCE_JNDI_NAME + " " + e.toString(), e);
   } catch (SQLException e) {
      throw new ServiceLocatorException(e.toString(), e);
   }
}
```

Figura 4.8 Segmento del código fuente de la aplicación del patrón Service Locator.

4.4.3.2 Business Delegate

En un sistema multi-capa requiere la invocación remota de métodos para enviar y recibir datos entre las capas. Los componentes de la capa de presentación pueden interactúar directamente con servicios de negocio. Esta interacción directa expone los detalles de la implementación del API del servicio de negocio a la capa de presentación. Como resultado, los componentes de la capa de presentación son vulnerables a los cambios en la implementación de los servicios de negocio: cuando cambia la implementación del servicio de negocio, la implementación del código expuesto en la capa de presentación también debe cambiar.

Además, podría haber una reducción de rendimiento en la red porque los componentes de la capa de presentación que utilizan el API de servicio de negocio hacen demasiadas invocaciones sobre la red. Esto sucede cuando los componentes de la capa de presentación usan directamente el API subyacente, sin cambiar el mecanismo del lado del cliente o agregar servicios.

Por último, exponer directamente los APIs de servicios al cliente fuerza a éste a tratar con los problemas de red asociados con la naturaleza distribuida de la tecnología Enterprise JavaBeans (EJB).

Estos problemas de desarrollo se deben a que los clientes de la capa de presentación necesitan acceder a servicios de negocio. Diferentes clientes, dispositivos, clientes Web, y programas, necesitan acceder a los servicios de negocio. Los APIs de los servicios de negocio podrían cambiar según evolucionan los requerimientos del negocio. Es deseable minimizar el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio, y así ocultar los detalles de la implementación del servicio.

Los clientes podrían necesitar implementar mecanismos de caché para la información del servicio de negocio. Es deseable reducir el tráfico de red entre el cliente y los servicios de negocio.

Definición El patrón Business Delegate el cual propone reducir el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio. El Business Delegate oculta los detalles de la implementación del servicio de negocio, como los detalles de búsqueda y acceso de la arquitectura EJB

El patrón Business Delegate actúa como una abstracción de negocio del lado del cliente; proporciona una abstracción pura, y por lo tanto oculta, la implementación de los servicios del negocio. Utilizando Business Delegate se reduce el acoplamiento entre los clientes de la capa de presentación y los servicios de

negocio del sistema. Dependiendo de la estrategia de implementación, Business Delegate podría aislar a los clientes de la posible volatilidad en la implementación del API de los servicios de negocio. Potencialmente, esto reduce el número de cambios que se deben hacer en el código de cliente de la capa de presentación cuando cambie el API del servicio de negocio o su implementación subyacente.

Sin embargo, los métodos de interfaz en el Business Delegate aún podrían requerir modificaciones si cambia el API del servicio de negocio. Si bien es cierto, que los cambios se harán con más probabilidad en el servicio de negocio que en el Business Delegate.

4.4.3.2.1 Beneficios de la aplicación de patrón Business Delegate

El principal beneficio de aplicar el patrón Business Delegate es ocultar los detalles del servicio. Por ejemplo, el cliente puede ser transparente para los servicios de búsqueda y nombrado. El Business Delegate también maneja las excepciones de los servicios de negocio, como excepciones java.rmi.Remote, excepciones Java Messages Service (JMS), etc. El Business Delegate podría interceptar dichas excepciones a nivel de servicio y generar en su lugar excepciones a nivel de aplicación. Las excepciones de nivel de aplicación son fáciles de manejar por los clientes, y pueden ser amigables para el usuario. El Business Delegate también podría realizar de forma transparente cualquier operación de reintento o de recuperación necesaria en el caso de un fallo en el servicio no exponer el cliente al problema hasta que se haya determinado que el problema no es solucionable. Estas ganancias representan una razón competitiva para utilizar el patrón.

Otro beneficio es que el Business Delegate podría almacenar resultados y referencias a servicios de negocio remotos. El Caché puede mejorar el rendimiento de forma significativa, porque limita los innecesarios y potencialmente costosos viajes por la red.

Un Business Delegate utiliza un componente llamado Lookup Service. Este componente es el responsable de ocultar los detalles de implementación de código de búsqueda del servicio de negocio. El Lookup Service podría estar escrito como parte del Delegate, pero en este sistema se implementa el componente Lookup Service utilizando el patrón Service Locator.

Cuando el Business Delegate se utiliza con un Session Facade, normalmente hay una relación uno-a-uno entre los dos. Esta relación existe porque la lógica que podría haber sido encapsulada en un Business Delegate en relación con su interacción con varios servicios de negocio (creando una relación uno-a-uno) normalmente se construye de vuelta en un Session Facade.

Finalmente, se debería tener en cuenta que este patrón se podría utilizar para reducir el acoplamiento entre otras capas, no simplemente entre las capas de presentación y de negocio.

4.4.3.2.2 Aplicación de patrón Business Delegate.

En el sistema aquí planteado el patrón Business Delegate es representado por una clase intermedia entre la capa de negocio y las clases integrantes del Front Controller. El siguiente diagrama muestra como interactúa la clase Business Delegate dentro del sistema.

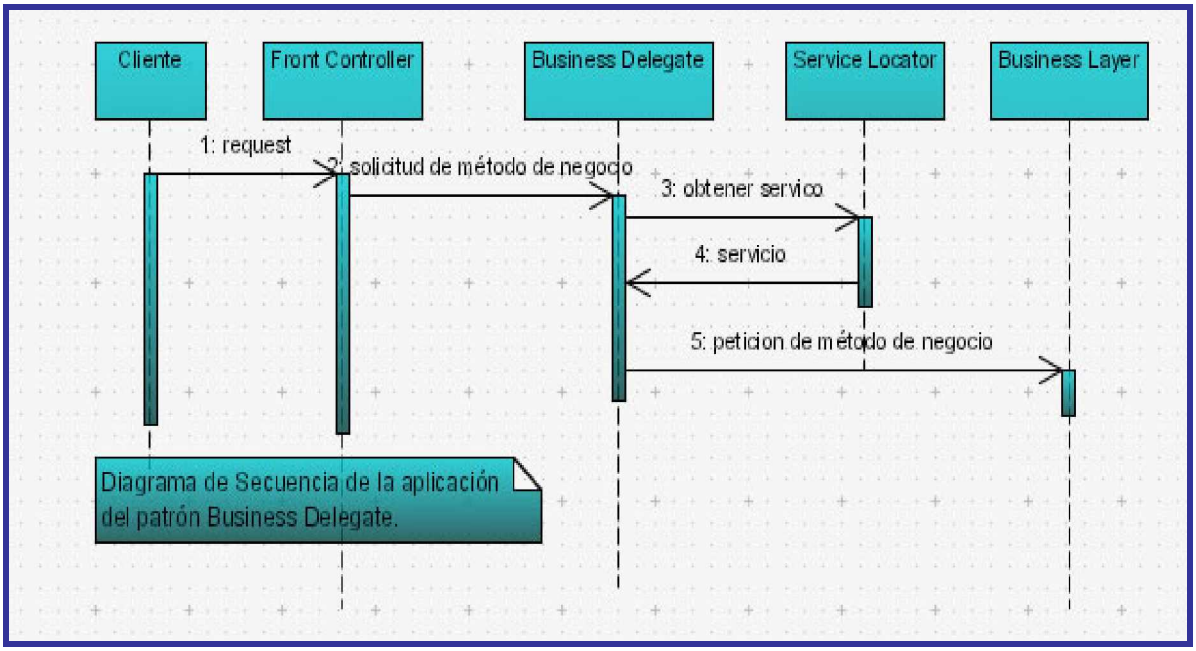


Figura 4.9 Diagrama de secuencia de la aplicación del patrón Business Delegate.

Como se puede observar Business Delegate utiliza un componente llamado Lookup Service, cual se refiere a la implementación del patrón Service Locator, el cual como se explicó previamente es el componente responsable de ocultar los detalles de implementación de código de búsqueda del servicio de negocio.

De esta forma se oculta los detalles de la implementación del servicio de negocio, como los detalles de búsqueda y acceso de la arquitectura EJB (capa de negocios). Veamos el siguiente fragmento de código para ver la implementación en código de este patrón.

En dicho código se puede observar que el constructor de esta clase encapsula la búsqueda al componente de negocio solicitado (en este caso se busca el bean de sesión PLSecurityBean). Además una vez creado un objeto de la clase Business Delegade deseada también se obtiene acceso a los métodos de negocio proveídos por la misma clase Business Delegate.

```
24 /**
25  * @author César Garzón Trinidad (garzon_c@yahoo.com.mx)
26  * Implementa el patrón Business Delegate de Sun Microsystems.
27  */
28 public class BpSecurity {
29
30     PlSecurityFacade bean;
31     Logger logger = Logger.getLogger("BpSecurity");
32
33     public BpSecurity() throws DelegateException {
34         ServiceLocator service;
35         PlSecurityFacadeHome home;
36         try {
37             service = ServiceLocator.getRemoteInstance();
38             home = (PlSecurityFacadeHome) service.getRemoteHome(
39                     JNDIKeys.PL_SECURITY_FACADE_JNDI_NAME,
40                     JNDIKeys.PL_SECURITY_FACADE_HOME_CLASS);
41             bean = home.create();
42         } catch (ServiceLocatorException e) {
43             throw new DelegateException(e);
44         }
45         catch (CreateException e) {
46             throw new DelegateException(e);
47         } catch (RemoteException e) {
48             throw new DelegateException(LogMessage.getMessageToShow(e));
49         }
50     }
51
52     public Map getAllWebRequest() throws DelegateException {
53         try {
54             return bean.getAllWebRequest();
55         } catch (RemoteException e) {
56             throw new DelegateException(e);
57         }
58     }
59
```

Figura 4.10Segmento del código fuente de la aplicación del patrón Business Delegate.

El siguiente fragmento de código muestra como con solo crear una instancia de la clase BpSecurity accediendo al método de negocio deseado (en este caso el método getAllWebRequest) sin necesidad preocuparse por el código de búsqueda del servicio de negocio, el cual ya fue implementado en la clase BPSecurity. Obteniendo así un desacoplamiento entre capas.

```
28 /**
29  * @author Cesar Garzón Trinidad
30  * Servlet de inicialización de Sistema
31  */
32 public class ServletStartup extends HttpServlet {
33 
34     private ServletContext context;
35     private static Logger logger = Logger.getLogger("ServletStartup");
36     /**
37      * Inicializa el map de peticiones del sistema.
38      * @see javax.servlet.Servlet#init(javax.servlet.ServletConfig)
39      */
40     public void init(ServletConfig config) throws ServletException
41     {
42         super.init(config);
43         BpSecurity service;
44         Properties p;
45         File rootAppDir;
46         File dirToCheck;
47         context = config.getServletContext();
48         try{
49             System.out.println("Starting Tesis Web Site.....");
50             logger.info("Starting Tesis Web Site.....");
51             service = new BpSecurity();
52             logger.info("Loading Web Requests......");
53             WebRequestInfo.iniPoolRequest(service.getAllWebRequest());
54             logger.info("Web requests were loaded successfully");
55             logger.info("Loading file properties..... ");
56             System.out.println("-----------------------------------");
```

Figura 4.11 Segmento del código fuente del archivo ServletStartup.java

Definición El patrón Business Delegate el cual propone reducir el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio. El Business Delegate oculta los detalles de la implementación del servicio de negocio, como los detalles de búsqueda y acceso de la arquitectura EJB

4.4.4 Capa de Integración

La capa de integración es el puente entre la capa de lógica-de-negocio y la capa del sistema de información empresarial (EIS). Encapsula la lógica para interactuar con la capa EIS. Algunas veces a la combinación de las capas de integración y de lógica-de-negocio se le conoce como capa central.

4.4.4.1 Data Access Object (DAO).

El acceso al almacenamiento persistente, como una base de datos, varía en gran medida dependiendo del tipo de almacenamiento (bases de datos relacionales, bases de datos orientadas a objetos, archivos planos, etc.) y de la implementación del vendedor. Muchas aplicaciones de la plataforma J2EE en el mundo real necesitan utilizar datos persistentes en algún momento. Para muchas de ellas, este almacenamiento persistente se implementa utilizando diferentes mecanismos, y hay marcadas diferencias en los APIS utilizados para acceder a esos mecanismos de almacenamiento. Otras aplicaciones podrían necesitar acceder a datos que residen en sistemas diferentes, por ejemplo, los datos podrían residir en sistemas mainframe, repositorios LDAP, etc. Otro ejemplo es donde los datos los proporcionan servicios a través de sistemas externos como los sistemas de integración negocio-a-negocio (B2B), servicios de tarjetas de crédito, etc.

Normalmente, las aplicaciones java utilizan componentes distribuidos y compartidos como los beans de entidad para representar los datos persistentes. Se considera que una aplicación emplea consistencia manejada por el bean (BMP) cuando sus beans de entidad acceden explícitamente al almacenamiento persistente -- el bean de entidad incluye código para hacer esto. Una aplicación con requerimientos sencillos podría por lo tanto utilizar beans de entidad en lugar de beans de sesión o servlets para acceder al almacenamiento persistente y recuperar o modificar los datos. Otra opción es que la aplicación utilice beans de entidad con persistencia manejada por el contenedor, y así dejar que el contenedor maneje los detalles de las transacciones y de la persistencia.

Las aplicaciones pueden utilizar el API JDBC para acceder a los datos en un sistema de control de bases de datos relacionales (RDBMS). Este API permite una forma estándar de acceder y manipular datos en un almacenamiento persistente, como una base de datos relacional. El API JDBC permite a las aplicaciones J2EE utilizar sentencias SQL, que son el método estándar para acceder a tablas RDBMS. Sin embargo, incluso dentro de un entorno RDBMS, la sintaxis y formatos actuales de las sentencias SQL podrían variar dependiendo de la propia base de datos en particular. Incluso hay una mayor variación con diferentes tipos de almacenamientos persistentes.

Los mecanismos de acceso, los APIs soportados, y sus características varían entre los diferentes tipos de almacenamientos persistentes, como bases de datos relacionales, bases de datos orientadas a objetos, archivos planos, etc. Las aplicaciones que necesitan acceder a datos de un sistema un mainframe o un servicio B2B se ven obligados a utilizar APIs que podrían ser propietarios. Dichas fuentes de datos dispares ofrecen retos a la aplicación y potencialmente pueden crear una dependencia directa entre el código de la aplicación y el código de acceso a los datos. Cuando los componentes de negocio -- beans de entidad, beans de sesión e incluso componentes de presentación como servlets y beans de

apoyo para páginas JSP -- necesitan acceder a una fuente de datos, pueden utilizar el API apropiado para conseguir la conectividad y manipular la fuente de datos. Pero introducir el código de conectividad y de acceso a datos dentro de estos componentes genera un fuerte acoplamiento entre los componentes y la implementación de la fuente de datos. Dichas dependencias de código en los componentes hace difícil y tedioso migrar la aplicación de un tipo de fuente de datos a otro. Cuando cambia la fuente de datos, también deben cambiar los componentes para manejar el nuevo tipo de fuente de datos.

Definición Un **Data Access Object** (DAO) es un patrón utilizado para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar éstos.

El DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Esta fuente de datos puede ser un almacenamiento persistente como una RDMBS, un servicio externo como un intercambio B2B, un repositorio LDAP, o un servicio de negocios al que se accede mediante CORBA Internet Inter-ORB Protocol (IIOP) o sockets de bajo nivel.

4.4.4.1.1 Implementación del patrón DAO en el sistema.

Los componentes de negocio que tratan con el DAO utilizan una interfaz simple expuesta por el DAO para sus clientes. El DAO oculta completamente los detalles de implementación de la fuente de datos a sus clientes. Como la interfaz expuesta por el DAO no cambia cuando cambia la implementación de la fuente de datos subyacente, este patrón permite al DAO adaptarse a diferentes esquemas de almacenamiento sin que esto afecte a sus clientes o componentes de negocio. Esencialmente, el DAO actúa como un adaptador entre el componente y la fuente de datos. El siguiente diagrama de secuencia muestra la implementación del patrón DAO en el sistema.

En el siguiente diagrama de secuencia se puede observar como la lógica de búsqueda de la fuente de datos queda encapsulada utilizando el patrón Service Locator utilizado dentro de la fabrica de DAOs. Posteriormente la fabrica de DAOs regresa una interfaz del DAO solicitado en la capa de negocios en este caso representado por el bean de sessión PlSecurityBean.

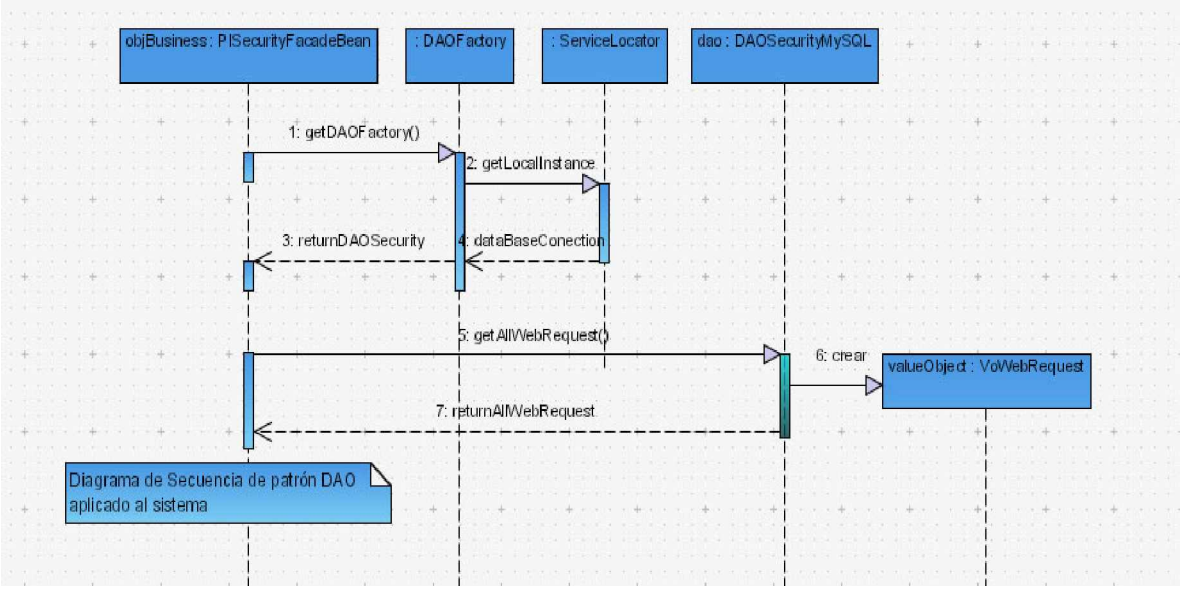


Tabla 4.12 Diagrama de secuencia de la aplicación del patrón DAO.

Como conclusión podemos decir que al aplicar el patrón DAO se obtienen los siguientes beneficios.

- Flexibilidad en la instalación/configuración de una aplicación.
- Independencia del vendedor de la fuente de datos.
- Independencia del recurso (base de datos relacional, base de datos orientado a objetos, ficheros planos, etc.).
- Reduce la complejidad de la implementación de la lógica de negocios (Session Facade)
- Más complejidad en el diseño, debido principalmente al conjunto de clases que colaboran en el DAO y en forma general, cada DAO es utilizado para modelar una tabla del modelo relacional.

4.5 Resultados obtenidos

Diseñar un curso para ser enseñado en línea requiere una preparación especial y una consideración por el material. Además de los retos normales de definir el contenido y objetivos de un curso, un curso en línea requiere la utilización de tecnología de punta y diseños de calidad. El diseño de un curso en línea requiere de un grupo de profesionales que apoye al instructor y lo ayude a generar un curso tecnológicamente viable, visualmente atractivo y pedagógicamente coherente.

Aunque los alcances del proyecto aquí propuesto no pretenden ser un curso en la Web si requiere de la preparación y creación de material organizado adecuadamente para que el objetivo de ser herramienta útil, la cual complemente un curso de Inglés presencial pueda llegar a ser una realidad.

Al término de este proyecto nos encontramos ante una alternativa para incrementar los medios que complementan una enseñanza presencial del idioma Inglés. La cual también contribuye a minimizar las desventajas en la educación al ofrecer un medio de acceso público disponible en cualquier momento.

El trabajo presente ofrece una serie de recursos pensados para fortalecer los conceptos aprendidos en un ambiente de enseñanza presencial.

La siguiente imagen muestra la página principal del proyecto la cual facilita una serie de recursos como herramientas de apoyo al aprendizaje del idioma Inglés al visitante del sitio.

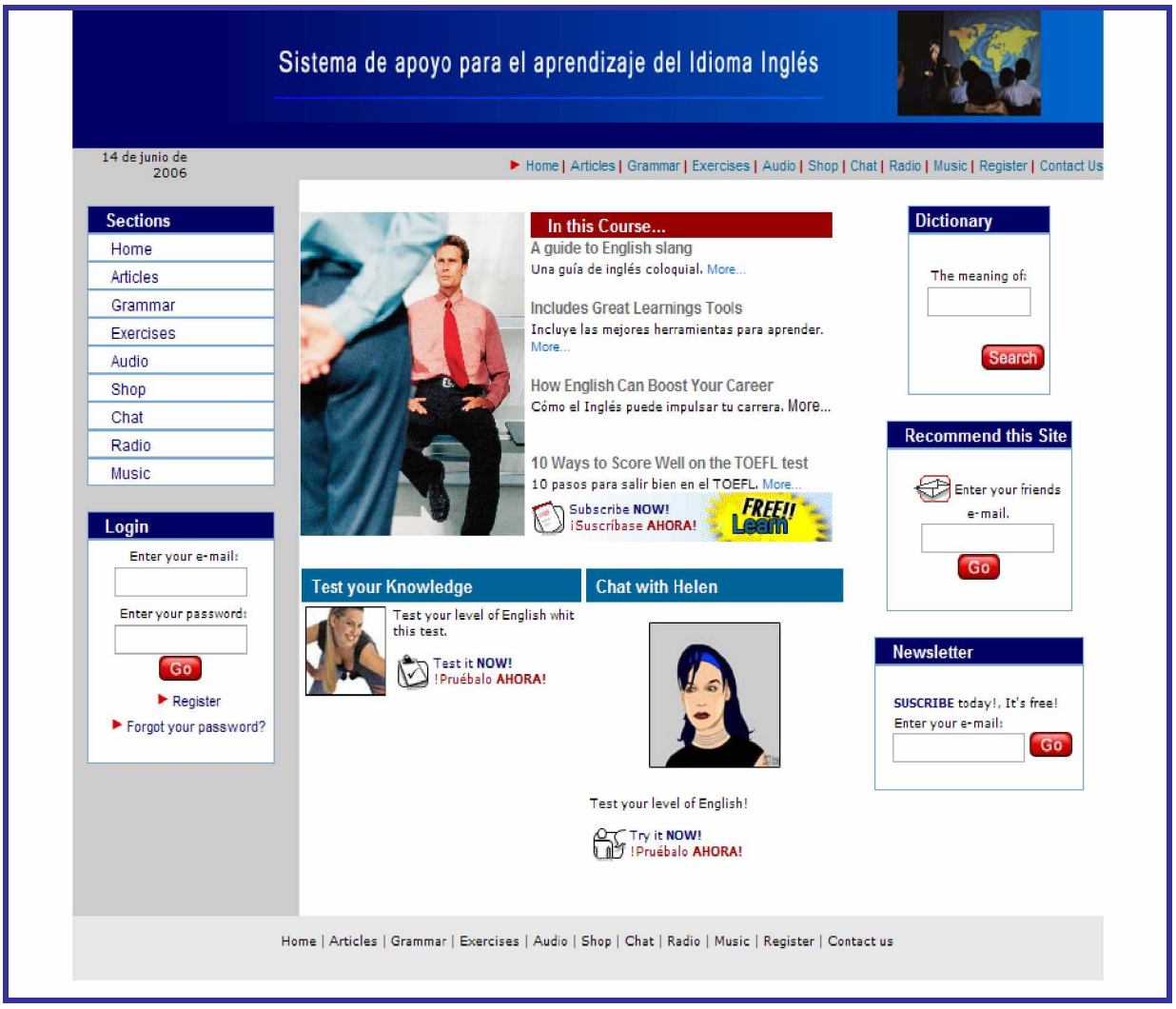


Figura 4.13 Página principal del sistema.

El sistema puede mostrar su contenido organizado en secciones agrupadas especialmente por el tipo de habilidad del Idioma Inglés que el usuario busca reforzar.

Figura 4.14 Página de administración de artículos publicados en el sistema.

Por otra parte para poder llevar el mantenimiento del contenido del sitio se cuenta con un sistema de administración al cual solo puede ser accedido por el administrador del sitio Web. La figura 4.16 imagen muestra la interfaz de usuario para el administrador del sitio en la cual se puede crear la traducción de artículo.

Se puede catalogar a este sistema dentro de lo que se denomina blended learning. Ya en el año 2002 - 2003, se acuñó el término “blended learning” para expresar la evolución que ha sufrido la antiguamente denominada formación virtual. El “blended learning” consiste en “mezclar” o complementar la formación presencial con la formación a través de las TIC. El “B-Learning” tomado de “Blended Learning”, término inglés que se traduce como “Formación Combinada” o “Aprendizaje Mixto”.

Contar con un sistema Web como el aquí propuesto contribuye a que el docente pueda trabajar a distancia, contando con la formación adecuada con el objetivo de que el alumno logre avanzar y conducir su aprendizaje a su ritmo, en el momento que mejor le convenga. Por la misma razón, el aprendizaje es más completo, ya que es el resultado de un compromiso activo del que aprende y no sólo del proceso unidireccional, que depende del profesor o formador y de sus

capacidades comunicativas, sus conocimientos y sus técnicas didácticas utilizadas en el aula.

Sin embargo este sistema también contempla retos a enfrentar los cuales podemos en listar en dos puntos de vista. Desde el punto de vista del proveedor de contenido, el reto es hacer el proceso de aprendizaje más interactivo y atractivo. Por lo anterior la elaboración de contenidos debe ser hecha por profesionales de la materia en colaboración con un grupo interdisciplinario que contribuya a contenidos de calidad que permitan la auto-retroalimentación del alumno.

Desde el punto de vista del receptor de la capacitación, entrenamiento o aprendizaje, su desafío es cambiar su mentalidad con respecto a la concepción que tiene de la educación, que es la educación tradicional y presencial. El e-learning reemplaza las limitaciones de los salones de clases comunes y los estudiantes deben comprender que la educación continua debe ser una parte rutinaria de su vida, tal como ha sucedido con muchas aplicaciones como el e-mail. Por otra parte carencia de computadora y/o conexión a Internet (Tan necesarios actualmente en una sociedad de la información) representan una limitante que sigue marcando una desigualdad en la educación.

5. Conclusiones.

La tecnología nos debe ayudar a construir un nuevo modelo empresarial, institucional e, incluso, un nuevo modelo de sociedad. Pero tenemos que ser nosotros quienes fijemos el ritmo preciso para avanzar, en ese nuevo paso hacia adelante, en la evolución humana. Una evolución en la que, al igual que siempre, Tecnología y Conocimiento aparecen como un binomio clave para el éxito.

5 Conclusiones

5.1 Alcances actuales del e-learning.

Según los puntos anteriormente estudiados actualmente el avance tecnológico proporciona nuevas herramientas que generan nuevos ambientes educaciones los cuales pueden ser aprovechados para mejorar la calidad de la educación así como la extensión de la cobertura de la misma. Uno de estos avances tecnológicos es el e-learning. Como se vio anteriormente e-learning implica la entrega de contenidos por medios electrónicos como Internet, intranets, televisión interactiva, CD ROMs, etc. Así pues, el e-learning incluye aquellas sesiones presénciales que utilicen herramientas digitales o electrónicas como medio para la difusión y la práctica de los contenidos de un curso.

Dentro del e-learning el Internet se presenta como una respuesta lógica para cubrir las nuevas demandas de educación y formación. Según la consultora Nielsen/Netrankings1, en Marzo del 2003 había en el mundo más de 607 millones de usuarios de Internet. América concentraba el 36,6% de la población online con más de 222 millones de usuarios. A continuación se encontraba Asia con un 35, 5% y Europa con el 28, 4% (más de 172 millones) de internautas.

Si bien el Internet como se analizó en el capítulo 1 **"Las computadoras en la educación de Hoy"**. Solo es una parte del concepto e-learning. Debido al crecimiento del World Wide Web, la gran capacidad de las redes corporativas y los grandes avances en las computadoras personales proporcionarán la estructura tecnológica suficiente para que el aprendizaje online pueda estar disponible 24 horas al día y 7 días a la semana (24×7). Además de una penetración más alta en los hogares con una tendencia en aumento. Proporcionando el medio idóneo para un nuevo modelo de formación flexible que proporcione la posibilidad de ser seguido "en cualquier lugar" (anywhere), "a cualquier hora" (anytime) y "para cualquiera" (anyone).

Sin embargo, éstos avances tecnológicos han ido más rápido que la adaptación de las estructuras sociales a este nuevo orden emergente. Las carencias ya se dejan ver en la falta de fuerza de trabajo cualificada para afrontar lo retos de estas nuevas estructuras organizacionales. La tecnología computacional puede jugar un papel importante para enriquecer las actividades docentes y hacer más interesante la participación activa del aprendiz en el proceso. Para poder incorporar la tecnología computacional en la sala clases (y fuera de ella) de manera efectiva puede ser de gran ayuda hacerlo de forma más bien implícita,

[1] Nielsen// Net Ratings: The gloenbal Standard for Internet Audience Measurements and Analysis. http://www.nielsen-netratings.com

como un instrumento más, pero totalmente integrado a las actividades docentes que se realicen.

Por otra parte, cabe la pregunta acerca de cuales son las habilidades o las destrezas necesarias para poder hacer uso de cursos de formación basados en el e-learning. La literatura y los datos ponen de manifiesto que son muy pocos los prerrequisitos básicos necesarios para seguir cursos de formación a través de las TIC (Tecnologías de la Información y las Comunicaciones). Basta con tener una buena actitud hacia el aprendizaje, sentirse cómodo en situaciones de cambio permanente y no tener recelo a la utilización de las TIC. Los conocimientos informáticos previos necesarios son mínimos aunque un conocimiento básico de herramientas como navegadores Web, procesadores de texto, correo electrónico y transmisión de archivos son muy recomendables.

Así pues, las habilidades para la adopción al cambio permanente y la buena actitud o la buena relación entre las personas con la tecnología son factores que deben tenerse especialmente en cuenta. Se recomendaría trabajar también hacia una homogeneización de las habilidades de la población en materia tecnológica.

Los factores que generan mayores desigualdades en cuanto a las actitudes y habilidades con las TIC en la población pueden ser el no tener acceso a una PC en el hogar, no contar con fácil acceso a Internet y la carencia de habilidades o capacitación en el uso de las TIC.

En resumen, se puede afirmar que el e-learning es una respuesta a las necesidades propias de la evolución de nuestra sociedad y por lo tanto, se debe hacer un esfuerzo por preparar a la población para el uso, el acceso y el disfrute de las TIC y del e-learning en particular. Minimizar las desigualdades y fomentar la integración deben constituir una prioridad en momentos de cambio social con el fin de aspirar a una sociedad más integrada, más justa y más democrática.

Como consecuencia las áreas de oportunidad que podemos identificar en el e-learning son:

- Falta de relación personal.
- Falta de calidad pedagógica.
- Escasa participación. A nivel tecnológico ya es posible crear un ambiente interactivo online a través de estudios de casos, demostraciones, juegos de rol, simulaciones, difusión online de vídeos, tutorías personalizadas, grupos de discusión, trabajos en grupo, chats, email, tutoriales, etc.
- Desconfianza.
- Falta de costumbre.
- Falta de conocimientos y manejo de computadora.
- Equipos informáticos potentes.
- Inversión inicialmente fuerte.

En cuanto a las ventajas del e-learning encontramos:

REDUCCIÓN DE COSTOS: Tanto a nivel empresarial como individual dentro de los costos de la educación o de formación hay que incluir los gastos de transporte, alojamientos y manutención en el lugar físico donde tiene lugar la actividad formativa en cuestión. Para aquellos individuos que quieren optar por universidades, cursos o maestrías internacionales los costos de transporte, alojamiento y manutención suponen un costo tan elevado o más que el del curso que se desea seguir. Esto supone un impedimento para muchas personas que no se pueden permitir estos gastos adyacentes a la educación pero que sí se podrían permitir el precio de la actividad formativa en cuestión. Si consideramos el derecho a la educación como un servicio de primera necesidad, se puede decir que, un sistema como el aquí propuesto da un paso adelante a favor de la democratización de la educación.

EL ACCESO A LA INFORMACIÓN EN EL MOMENTO JUSTO: Los servicios de e-learning ofrecen la posibilidad al profesor de actualizar los materiales de forma casi instantánea. Esto proporciona a los estudiantes acceso inmediato a la información más actual. Además la información puede ser consultada tantas veces como sea necesario y en el momento justo en el que se necesite en lugar de ser "aprendida" una única vez durante una sesión presencial y correr el riesgo de haberla olvidado en el momento en el que se vuelva a necesitar.

POSIBILIDAD DE COMPAGINARLO CON OTRA ACTIVIDAD: Debido a que el aprendizaje online nos permite acceder a él en cualquier momento brinda la posibilidad de compaginar actividades. Permitiendo un mayor espacio para los distintos ritmos y estilos de aprendizaje a través de un proceso más personalizado. Con un acceso 24×7 los estudiantes pueden aprender a su propio ritmo y revisar los materiales tantas veces como lo necesiten.

LAS MEJORAS DE LA INTERACTIVIDAD EN EL E-LEARNING: Las nuevas tecnologías aplicadas a las plataformas de e-learning han eliminado muchas de las barreras de comunicación y de interactividad que se han identificado como una de las principales desventajas del e-learning.

Finalmente también podemos decir que México desde un punto de vista de la economía global está en desventaja por que nosotros tenemos que comprar a un precio muy elevado la tecnología; las licencias de uso para software, música, películas; licencias de patentes para los productos que producimos: farmacéuticos y tecnológicos. A cambio de lo anterior, México vende materias primas (petróleo, productos agrícolas, ganaderos, etc), y mano de obra. Es obvio que México no ha participado como líder en el desarrollo tecnológico y tiene que pagar el costo de esta dependencia a precios muy altos. Es por ello que el país debe identificar el e-learning como un área de oportunidad e invertir en proyectos que permitan brindar beneficios tanto a la educación como en el desarrollo de tecnologías propias.

5.2 Conclusiones Técnicas

Las plataformas tecnológicas son fundamentales en la definición de los sistemas de información. En este entorno, aparece la figura del arquitecto de sistemas, quién aportará la perspectiva necesaria para tomar las decisiones correctas. En el desempeño de sus funciones, deberá tener en cuenta los objetivos y necesidades de la organización así como los límites presupuestarios, de plazos y de riesgos aceptados por la misma.

A la hora de decidirse por alguna plataforma de desarrollo será necesario tener en cuenta el hardware y sistemas operativos que soportarán las aplicaciones. La elección estará determinada también por el conocimiento y la cultura existente en la organización: conocimiento de lenguajes de programación. Además, influirá también en la decisión la necesidad de integración con las aplicaciones ya existentes en la organización, así como la complejidad de la aplicación a construir y la envergadura del proyecto.

En este escenario, han surgido los Servicios Web que posibilitan soluciones que integran componentes al margen de la plataforma en la que han sido implementados. De esta forma, se consigue un marco de trabajo abierto y flexible que facilita la convivencia e integración de las plataformas.

Actualmente, el mercado reconoce que las dos plataformas de desarrollo con un mayor futuro son .NET y J2EE. De hecho, constituyen en la actualidad estándares de facto que se utilizan en la práctica totalidad de los nuevos desarrollos y están ofreciendo resultados altamente satisfactorios en plazo y calidad de resultados. (Debido a que se eligió la plataforma J2EE para el desarrollo del presente sistema los puntos a continuación son referentes a dicha plataforma).

En el proceso de construcción, la arquitectura multicapa debe integrar componentes funcionales ínter-operables y autónomos, generando aplicaciones Web basadas en clientes ligeros. Además dicha arquitectura se debe caracterizar por una alta inclinación en la reutilización de componentes para mejorar el mantenimiento del software y acortar el tiempo de construcción.

Debido a las múltiples posibilidades arquitectónicas de la arquitectura J2EE, es conveniente disponer de una amplia experiencia y conocimiento para tomar las decisiones adecuadas en la construcción de estas aplicaciones. Es importante también la utilización de ciclos de desarrollo iterativo. Finalmente, la concepción de la aplicación debe realizarse mediante métodos como UML (Unified Modeling Language) y metodologías específicas que aseguren la concepción e integración de componentes. En lo referente al despliegue de las aplicaciones generadas, la alta estructuración en capas requiere una puesta en producción cuidadosa, analizando y configurando cada uno de los elementos (servidor Web, servidor de aplicaciones, bases de datos, etc.).

5.3 Conclusiones Finales sobre el Proyecto "Sistema Web de Generación de Herramientas para el Aprendizaje del Idioma Inglés"

Hoy, la revolución de la información y las comunicaciones tiene gran influencia en la educación por lo que se requiere una comprensión cabal del lenguaje de las nuevas tecnologías y adelantos científicos a nivel mundial. Para esto el idioma Inglés ha sido y parece que seguirá siendo, por el momento, el vehículo fundamental para la comunicación internacional en todas las esferas. La limitada capacidad de manejo de este idioma impedirá el acceso a todo el caudal de información que se genera en todo el mundo a una gran velocidad. Para cualquier estudiante de nivel superior, entonces, el conocimiento del idioma inglés adquiere una importancia crucial para su futura vida profesional.

El aprendizaje del idioma Inglés es un punto estratégico para poder competir en un mundo globalizado de hoy día. Aprovechando el contexto tecnológico actual se puede utilizar la Web como medio para facilitar el acceso a herramientas especializadas que contribuyan tanto al aprendizaje como al reforzamiento del idioma Inglés, obviamente esto es extensible a toda la educación en general.

Sin embargo no solo se tiene que ampliar la cobertura de la educación así como mejorar su calidad. Es decir realizar un proyecto de e-learning como este tiene que ir asesorado con y de la mano con la enseñanza tradicional. Extendiendo el concepto de educación y viendo tanto las nuevas herramientas tecnológicas como las tradicionales como una sola que se complementan. Realizando planes de estudio que las complementen.

Programas como este puede ayudar a minimizar las desigualdades educacionales en los estudiantes. Es por eso que se debe tomar en consideración el amplio abanico de posibilidades que representa el e-learning al realizar nuevos planes de estudio.

Por otra parte el aumentar el uso de la tecnología como complemento a la educación tradicional implica resolver otros problemas como el llegar a la gente que tiene menos recursos, que son las más carentes de tecnología, por lo que es importante que se trabaje en la reducción de costo.

& Bibliografía

Architecting and Designing J2EE Applications
Sun Microsystems SL-425

Component Development for the Java Platform
Stuart Dabbs Halloway
Addison-Wesley

Computer Education for Teachers. Integrating Technology into Classroom Teaching.
Vicky Sharp
McGraw-Hill

Core J2EE Patterns Best Practices and Design Strategies
Alur Deepack, Crupi John, Marks Dan
Second Edition Ed. Prentice Hall

Developer's Guide
SUNONE Application Server version 7 Septiembre 2002, 816-7149-10

Developer's Guide to clients Guide
SUNONE Application Server version 7 Diciembre 2002, 817-0462-10

Developer's Guide to Enterprise Java Beans Technology
SUNONE Application Server version 7 Septiembre 2002, 816-7151-10

Developing Enterprise Applications with J2EE and UML
Zaman Ahmed Khawar, Umrysh Carye
Addison Wesley 2002

EJB™ Design Patterns Advanced Patterns, Processes, and Idioms
Marinescu Floyd
Wiley Computer Publishing, 2002

Enterprise JavaBeans Especification
Sun Microsystems Version 2.1

Getting started Guide
SUNONE Application Server version 7 September 2002 816-7146-10

Ingeniería del Software un Enfoque Práctico
Roger S Pressman.
McGraw-Hill

Jakarta Struts Live
Richard Hightower, SourceBeat
HighlandsRanch, Colorado EUA 2004

J2EE and XML Development
Kurt A. Gabrick. David B. Weiss
Manning

J2EE architect's handbook: how to be a successful technical architect for J2EE applications
Derek C. Ashmore.
DVT Press

Patrones de Diseño Aplicados a Java
Stelting Stephen, Maassen Olav
Ed. Prentice Hall 2003

Profesional Java 2 v5.0
W. Clay Richardson, Donal Avondolio.
Wrox 2005

Servlets y Java Server Pages. Guía Práctica.
Hall Marty
Ed. Prentice Hall 2001

Sun Certified Enterprise Architect for J2EE
Allen Paul, Bambara Joseph
Ed. Mc Graw Hill /Osborne 2003

Technology, Open Learning and Distance Education.
A.W Bates
Routledge

The J2EE Tutorial
Green Dale y otros
Addison Wesley 2003 Sun Microsystems.

The Use and Misuse of Computers in Education
Allan B. Ellis
McGraw-Hill

Understanding Computers and Cognition.
Winograd, T. and Flores, F. (1986).
Ablex, Norwood, NJ.

Using Technology in the Classroom
Gary G. Bitter. Melissa I. Pierson
Pearson

Writing Enterprise Applications with Java 2 SDK Enterprise Edition
Parlan Monica
SUN Microsystems, Septiembre 2000

: Sitios Web Consultados:

Tesis doctoral.
Una arquitectura cognitiva para el diseño de entornos telemáticos de enseñanza y aprendizaje. Miguel Rodríguez Artacho
http://sensei.lsi.uned.es/~miguel/tesis/

Berril@b
El grupo de investigación Berril@b se formó con el objetivo de investigar, diseñar y desarrollar proyectos que tengan relación con la integración de las Tecnologías de la Información y la Comunicación en el ámbito educativo.
http://www.sc.ehu.es/topcogoj/Berril@b/publica.htm

Análisis y diseño de sistemas.
George Manson University Prentice-Hall Hispanoamericana, S.A.
http://www.ccee.edu.uy/ensenian/catcomp/material/aydisis.pdf

Sistemas multimedia en Educación
Antonio-Ramón Bartolomé Pina
http://www.lmi.ub.es/personal/bartolome/articuloshtml/multimedia_94/index.html

Teoría del aprendizaje
http://www.llibreriapedagogica.com/butlleti20/teoria_del_aprendizaje.htm10.htm

Ciclo de vida del Software
http://www.biblioteca.co.cr/pdf/unidad12-4.pdf

Diseño Instruccional y Teoría del Aprendizaje
Brenda Vergel Estudiante de Postgrado del Programa Comunicaciones y Tecnología Educacional de la Universidad de Saskatchewan Canadá Mayo, 1998
http://www.usask.ca/education/coursework/802papers/mergel/espanol.pdf

Sitio Web de Sun Microsystems.
http://www.java.sun.com

The Future of English? was commissioned by the British Council and written by researcher David Graddol.
http://www.britishcouncil.org/learning-research-futureofenglish.htm

The global Standard for Internet Audience Measurements and Analysis.
Nielsen// Net Ratings:
http://www.nielsen-netratings.com