Capitulo 4. Diseño del sistema

En este capítulo comprende la segunda fase de la ingeniería de software el diseño. Esté no puede ser realizado sin antes haber completado la fase de análisis debido a que están totalmente ligados. En pocas palabras detrás de un buen diseño hay un gran análisis. Pero en que consiste el diseño, pues básicamente se refiere a llevar al siguiente paso el análisis. El diseño es un producto más cercano a la realidad pero aún sigue en papel. Es decir, en el análisis se definió que va hacer el sistema y como lo debe hacer en esta parte se define a través de que tecnologías lo va hacer y de que medios se pueden valer, teniendo en cuenta el proceso que se definió en la primera fase.

4.1 Diseño de secuencias

A diferencia de los diagramas de caso de uso los diagramas de secuencia permiten ver no sólo el funcionamiento del sistema sino también la posible manera en que será implementado el diseño ya que es más detallado. En el diagrama de secuencias se muestran los objetos y los mensajes enviados cronológicamente de arriba hacia abajo, entre los objetos, lo que permite tener una idea más clara de la implementación (Fig. 4.1.1). A continuación se muestran un diagrama significativo creado en base a los casos de uso descritos en el apéndice D.

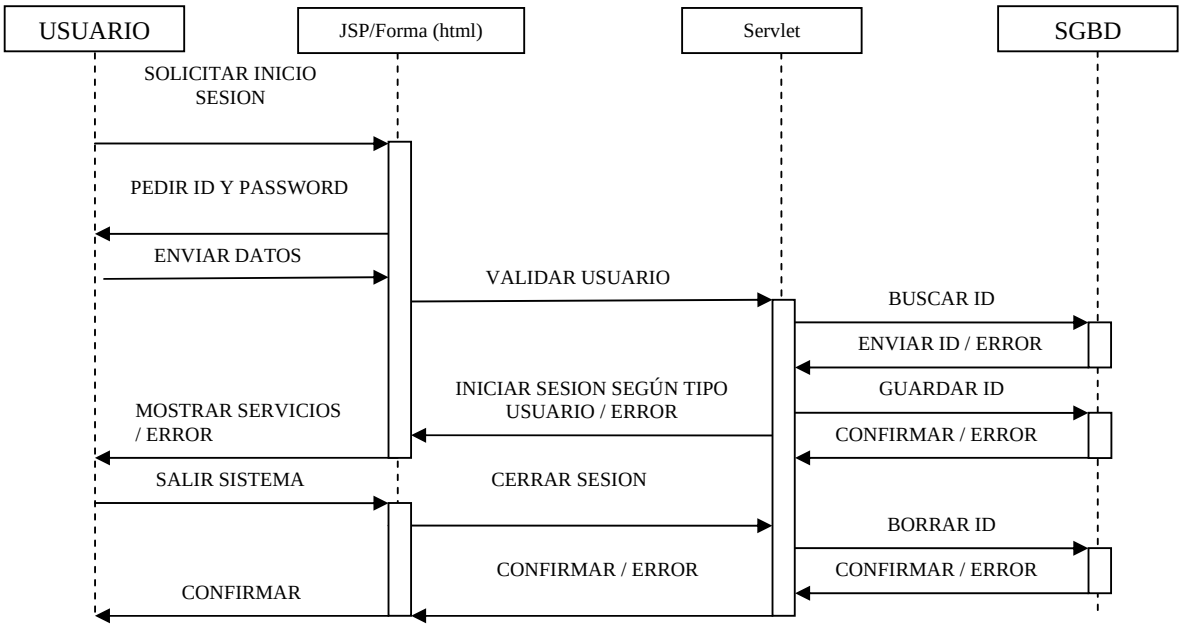

Fig. 4.1.1 Diagrama de secuencias manejo de sesión

Como se puede ver en la Fig.4.1.1 un usuario nunca observa directamente el procedimiento que tiene que realizar el sistema para que el usuario pueda acceder a éste. Ya que este únicamente interactúa con el sistema mediante un JSP o una forma HTML.

4.2 Creación de Base de Datos

A pesar de que en está aplicación se puede permitir el uso o reutilización de ciertos sistemas de base de datos, es importante definir la estructura que debe tener dicha BD, para un funcionamiento adecuado. Ya que sino se definiera cierta estructura el sistema caería en problemas de redundancia o albergaría información innecesaria lo que puede provocar inconsistencias en un futuro [Silberschatz 2003].

4.2.1 Diseño Modelo Entidad-Relación

Antes que nada se deben Identificar entidades que interactuaran con el sistema directamente, es decir, aquellos objetos que tienen atributos y pueden tener relación con alguna otra de las entidades, por decir algo la entidad Profesor se puede relacionar con la entidad Curso mediante "imparte el". Siguiendo esté enfoque se definieron las siguientes entidades:

Alumno **Profesor**

Curso **Examen**

Pregunta **Respuesta**

Acceso

Una vez definidas las entidades que interactuaran directamente en este sistema se deben establecer las relaciones que existen entre todas las entidades como se muestra a continuación en la siguiente tabla:

ENTIDAD	RELACIÓN	ENTIDAD
Profesor	**Consigue**	acceso
Alumno	**Obtiene**	acceso
Profesor	**imparte el**	curso

Alumno	**esta inscrito en**	curso
Alumno	**tiene resultado de**	examen
Examen	**pertenece a**	curso
Examen	**contiene la**	pregunta
Pregunta	**Tiene**	respuesta

En base a lo anterior se procede a la elaboración del Diagrama Entidad-Relación (Fig.4.2.1) poniendo a las entidades dentro de rectángulos y a las relaciones dentro de semicírculos con sus respectivos atributos.

Según el diagrama de la Fig.4.2.1, podemos ver que de las siete entidades resultan ocho relaciones entre dichas entidades. Esto tentativamente puede provocara la creación de ocho a quince tablas en la base de datos.

4.2.2 Tablas derivadas

Para poder definir las tablas resultantes del diagrama 4.2.1 se deben considerar los criterios de derivación de tablas según las cardinalidades. De acuerdo a [Silberschatz 2003] antes de comenzar con la derivación de tablas se debe tener en cuenta tres reglas básicas:

- Todo tipo de **entidad** se convierte en una **relación**.
- Todo tipo de **interrelación N:M** se transforma en una **relación**.
- Para todo tipo de interrelación **1:N** se realiza lo que se denomina **propagación de clave** (regla general), o se crea una nueva relación.

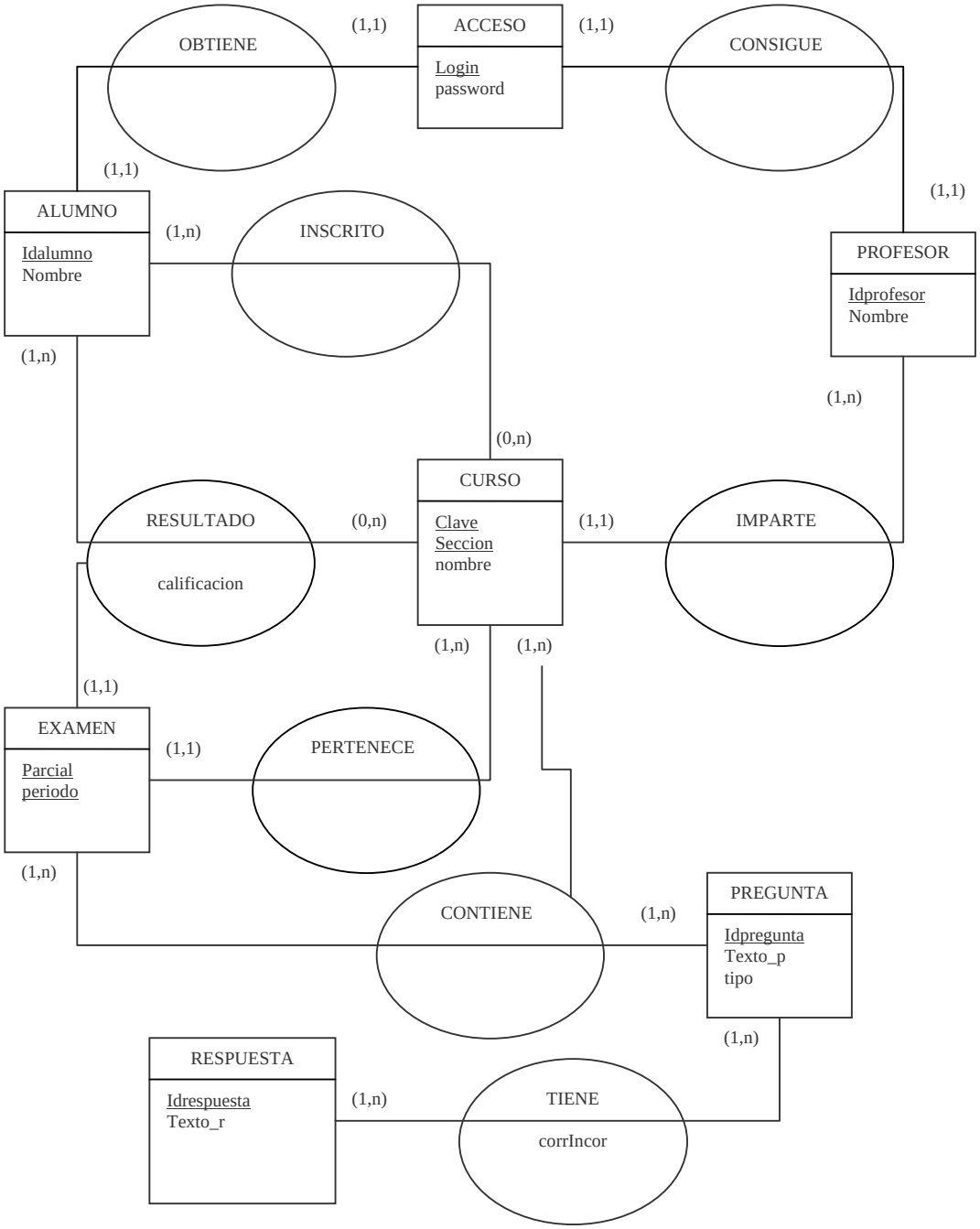

Fig. 4.2.1 Diagrama Entidad Relación

A continuación se muestra un ejemplo del empleo de las reglas para la derivación de tablas:

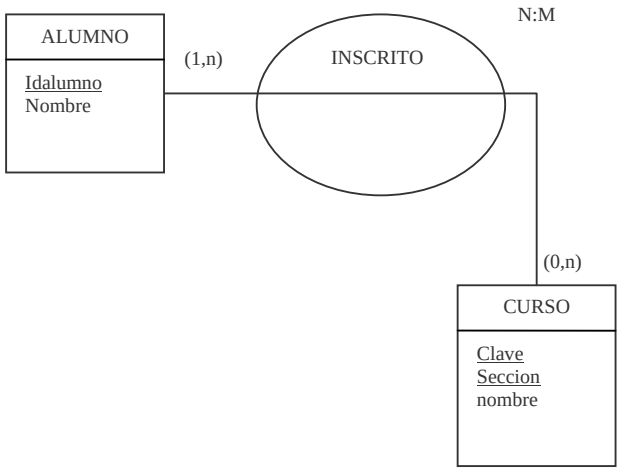

Fig 4.2.1.1 Relación Alumno - Curso

Debido a que la entidad CURSO en la Fig.4.2.1.1 puede tener varios alumnos inscritos o no tener alumnos **(0,n)**, y por otro lado un ALUMNO debe estar inscrito en uno o mas cursos **(1,n)**, en ninguna entidad se puede propagar alguna clave, por lo tanto, la relación INSCRITO conviene convertirla en una tabla con llaves extranjeras derivadas de las llaves primarias de CURSO y ALUMNO. En pocas palabras como las dos cardinalidades son de orden n no se puede propagar la relación en ninguna de las entidades, entonces se deben crear 3 tablas.

Una vez derivadas todas las tablas siguiendo este procedimiento (véase apéndice A) tenemos como resultado las siguientes tablas:

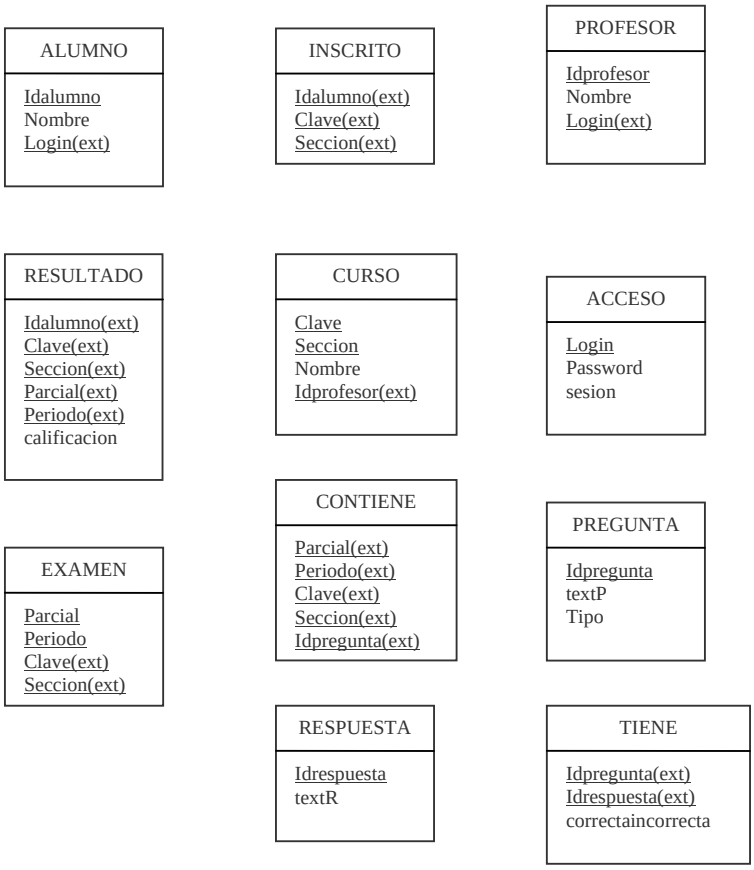

Fig 4.2.2 Tablas resultantes.

4.2.3 Establecimiento de Base de datos

Ahora que se conoce bien los datos y posibles tablas necesarias se debe aplicar la teoría de las formas normales (apéndice A) para obtener el diseño final, de tal manera que la base de datos no sea inestable, contenga la mínima redundancia y sea coherente para el sistema, con lo que se evitaría anomalías al modificar, insertar o eliminar algún dato.

Con todo lo anterior se asegura una coherencia de la base de datos ya que permite que por ejemplo los alumnos no estén inscritos en cursos que no aparezcan el la base de datos o que en un curso esté registrado con un profesor que no exista. El siguiente y último paso es la creación física de dicha base de datos, para esto se utilizaron los *querys* definidos en el *apéndice A.*

4.3 Diseño heurístico para generar exámenes

Una vez que un alumno entre al sistema este debe ser registrado en una tabla de usuarios que estén conectados dentro de la base de datos para evitar que inicie alguna otra sesión, luego este debe elegir el examen a tomar, sólo así el examen será generado y presentado al usuario por la aplicación de una manera transparente.

Para la generación de exámenes en está aplicación se debe crear una tabla dentro de la base de datos donde se almacenara el id de cada pregunta que este en uso y así se asegurara que los exámenes generados simultáneamente no sean iguales debido a que no podrán tomar las preguntas que tengan el id registrado en dicha tabla, claro si en la base de datos hay más de 100 preguntas disponibles para generar un examen de cierto curso. Si lo anterior no es posible, la generación de exámenes consistirá primero en mostrar el orden de las preguntas de forma aleatoria, de la misma manera para las respuestas asociadas a cada pregunta.

Por último, nótese que en la base de datos las preguntas están asociadas a un examen y dicho examen a un parcial de cierto curso para una sección, entonces se puede generar un examen final. Bastara con añadir una variante a la heurística de la generación de exámenes para este caso, es decir, se debe hacer un apartado en el código del sistema para que distinga cuando debe generar un examen parcial o final, ya que se puede obtener un dominio más grande de preguntas para generar exámenes finales y así se pueda lograr la generación real de exámenes dinámicos. De igual forma se podrían aplicar y/o generar exámenes departamentales, es decir, que las preguntas utilizadas no dependan de la sección a la que está inscrito el alumno, sino más bien sólo del curso, pero como podemos encontrarnos con las mismas preguntas simplemente planteadas de distinta forma por los profesores esta opción queda descartada.

4.4 Diseño modulo profesor

4.4.1 Activación de exámenes

Para esta sección se puede hacer un diseño simple en base al patrón de diseño MVC, en el JSP (vista) que se muestre al usuario debe existir una *forma* de tipo POST (de esta manera se pasan los parámetros como variables en un bloque de datos, y si se

utilizara el tipo GET los valores de la *forma* se enviarían como URL) para garantizar que los datos de la *forma* lleguen completos. Una vez enviados los datos al servlet (controlador) este se encargara de llamar al método, función o clase (modelo), que entrará a la base de datos y activara/desactivara el examen seleccionado en la *forma* por el usuario, por último el servlet notificara al usuario el éxito o fracaso de su petición mediante un JSP.

4.4.2 Creación, Actualización y/o modificación de preguntas y respuestas dentro del sistema

Como se comento en el capítulo anterior el profesor será el único encargado de proporcionar las preguntas al sistema. Por tal razón se deben considerar opciones alternativas para que realice de la manera más cómoda dicha acción. Es decir, se debe pensar en las posibles maneras que esté agregue preguntas al sistema de manera fácil, flexible y entendible, con esto me refiero a que si ya capturo en alguna ocasión una pregunta esta además de ser guardada en la base de datos también debe poder guardarla en un archivo ya que si por alguna razón dicha pregunta sea eliminada del sistema, el usuario profesor pueda agregarla de nuevo sin tener que capturarla sólo indicándole al sistema el archivo que la contenga. Por otro lado el usuario también debe tener el poder de modificar las preguntas que ha guardado previamente en el sistema.

Con esto hemos visto que básicamente se debe contar con tres opciones (Capturar manualmente, capturar desde archivo y modificar preguntas existentes), para la primera opción siguiendo el patrón de diseño MVC, se debe crear una *forma* tentativamente dinámica dentro del JSP, es decir, que el usuario indique cuantas preguntas desea capturar y automáticamente se genera una forma con cierto numero de campos para las preguntas y respuestas. Después de capturadas dichas preguntas esta *forma* es enviada al servlet que se encargara de leer sus parámetros y mandarlos a la función, método o clase, que guardara dichas preguntas en la base de datos asignándoles un id distinto a cada pregunta y respuesta cuidando que estos no se repitan ya que se crearía una gran inconsistencia en las tablas. Finalmente el servlet notificara al usuario el éxito o fracaso de su petición mediante un JSP.

Para la segunda opción antes de diseñar la manera en que el sistema obtendrá los datos del archivo, se debe pensar en el formato que debe tener esté archivo y como debe ser llenado, por lo que se debe hacer uso de alguna especie de etiquetado para diferenciar las preguntas de las respuestas y a su vez hay que pensar en como asociarlas cuando es buena o mala la respuesta. Entonces si se esta pensando como etiquetar el archivo, lo mas factible a mi parecer es diseñar el formato de dicho archivo en Base a XML y asociarle un xsd para validar su estructura y un xsl para generar una vista entendible al usuario mediante un navegador Web(véase Apéndice E).

Suponiendo que esto ya se tiene este formato definido este tipo de archivos se pueden crear utilizando los paquetes JDOM y Xerxes. Dichos paquetes también sirven para abrir y manipular el contenido de los archivos XML siempre y cuando estén bien construidos.

El archivo de preguntas puede ser creado al mismo tiempo que el alumno ingresa manualmente las preguntas de examen. La información para llenar dicho archivo se procesara mediante una *forma* que será enviada al servlet que se encarga obtener los parámetros y enviarlos a la función, método o clase asignada a guardar dicha información. Es decir, que en lugar de que se guarde directamente en la base de datos dicha información primero se guardara en un archivo XML siguiendo un formato definido (ver apéndice E.2) y se mostrara al usuario mediante un XSL para que verifique si la información es correcta, es ahí donde se debe establecer nuevamente el dialogo con el usuario para saber si desea conservar dicho archivo, después de ésto se podrá hacer la inserción a la base de datos pero ahora la información será extraída del archivo mostrado al usuario, una vez terminado este proceso se deberá borrar el archivo del servidor.

Por último el usuario podrá realizar la modificación de las preguntas almacenadas en el sistema. Esto se realizara en caso de que el usuario quiera redefinir la estructura de cada pregunta o respuesta, esto mediante 2 peticiones al servidor, la primera petición será solicitando extraer las preguntas asociadas a cierto examen y serán presentadas al usuario a través de un JSP de tal modo que pueda modificar el texto directamente. Una vez terminada dicha acción el usuario realizara la segunda petición al servidor pero ahora será para almacenar la nueva versión de dichas preguntas. Este y proceso al igual que los siguientes se realizaran siguiendo la estructura del patrón de diseño MVC de manera similar a como se manejara en las situaciones previamente descritas (por ejemplo la descrita en el tema 4.4.1).

4.4.3 Consulta resultados sobre exámenes aplicados

Esta parte es de gran utilidad a un profesor, primero porque le puede ayudar a llevar un control de sus alumnos y en segundo ya que le puede servir como retroalimentación al observar mediante las calificaciones altas y bajas quien realmente está comprendiendo lo visto en clase y quien no. Por otro lado, también puede llegar a notar si sus métodos de enseñanza dan frutos o nadie comprende lo que intenta transmitir, si se diera el caso de que todos aprueban con buenas calificaciones o por el contrario nadie logra aprobar alguno de sus exámenes. Por tal razón, si lo desea puede elegir que alumnos pueden volver a tomar cierto examen.

4.5 Diseño modulo alumno

4.5.1 Toma de examen para ser evaluado

Al igual que en el proceso para modificar preguntas y respuestas éste consiste en hacer 3 peticiones al servidor, la primera es para obtener la lista de exámenes disponibles para el usuario, la segunda es para obtener el examen que elija el usuario y la tercera es para evaluar las respuestas contestadas y guardar la calificación obtenida en la tabla de resultado, de tal modo que a manera de vista el usuario reciba la calificación de dicha evaluación en pantalla. Y si dicho usuario no queda conforme con la calificación y vuelve a tomar el mismo examen su calificación permanecerá igual y no podrá ser removida más que por el profesor.

4.5.2 Consulta de calificaciones en exámenes

Al igual que el profesor un alumno debe llevar el control de sus calificaciones. Esto con la finalidad de enriquecer la retroalimentación de curso, ya que si va presentar un examen final al consultar las calificaciones obtenidas en varios exámenes podrá identificar los temas en los que obtuvo bajo desempeño y necesita reforzar su estudio. Por otro lado también es importante que tenga conocimiento de sus calificaciones debido a que si cambia sus hábitos de estudio puede observar si estos han sido positivos o negativos.

4.6 Discusión

Para finalizar el capítulo podemos concluir la arquitectura general de acuerdo a las características del sistema debe seguir un modelo de tres capas (Fig.4.6.1). La capa uno representa los servicios al usuario (interfaz de los servicios como iniciar sesión, tomar examen, etc.), la capa 2 servicios de trabajo (lógica de la creación y aplicación de cada examen, servidor)(Fig4.6.2) y capa 3 servicios de datos (la base de datos de dónde se obtengan las preguntas, se registren los resultados, etc.).

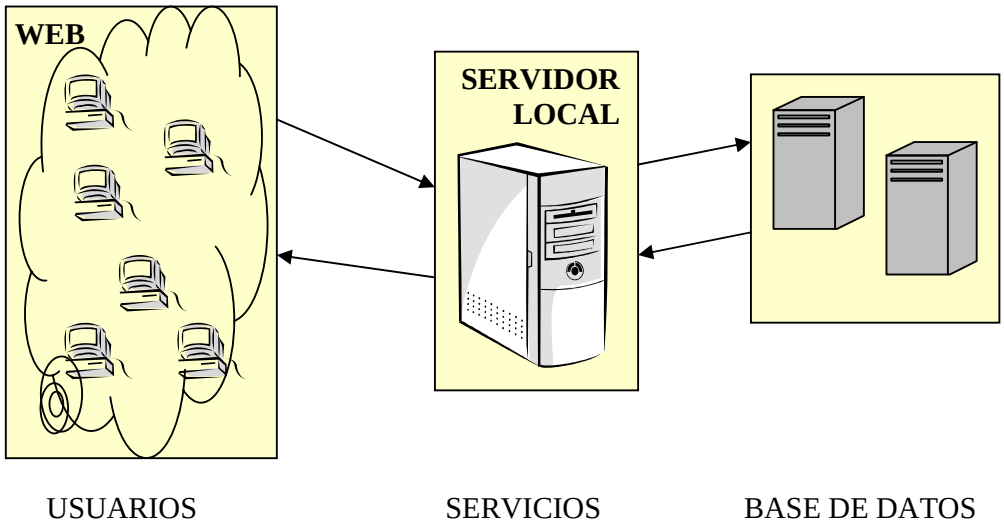

Fig.4.6.1 Modelo de Capas

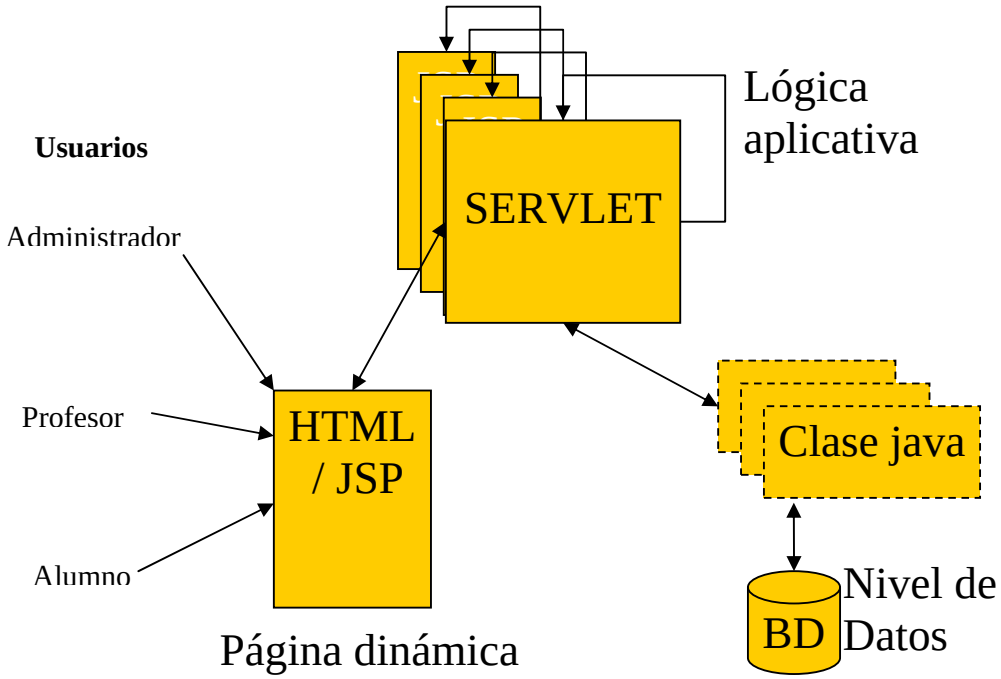

Fig.4.6.2 Esquema de arquitectura centrada en páginas