una aplicaci´on Web utiliza una p´agina Web para introducir datos (por ejemplo formularios), enviarlos al servidor, procesar dichos datos (de acuerdo a las pol´ıticas del negocio) y mostrar los resultados por medio de otra p´agina Web. El desarrollo de aplicaciones Web result´o ser de gran ayuda para las organizaciones, pero a la vez requirio la soluci´on de problemas asociados tales como la integraci´on con sistemas legados, la evoluci´on constante y la capacidad de manejo de grandes cantidades de usuarios. Otros puntos cr´ıticos que se tienen que considerar son la seguridad y la presentaci´on como parte de la experiencia de usuario.

En ocasiones se confunde el termino “capas” con el termino “nivel”. El primer t´ermino se utiliza para referenciar a las distintas “partes” en las que un aplicaci´on se divide desde el punto de vista l´ogico. Mientras que el segundo t´ermino corresponde a la forma f´ısica en que una aplicaci´on se organiza. Un ejemplo muy simple (descrito mediante la Figura 4.1) y que es muy com´un encontrar, es una aplicaci´on que tiene dos niveles (nivel de aplicaci´on y nivel de datos), en donde cada nivel puede tener varias capas. En este caso, el nivel de aplicaci´on puede estar constituido por la capa de presentaci´on y por la capa de l´ogica de negocio y el nivel de datos puede contener s´olo la capa de datos. Como podemos ver en el nivel de aplicaci´on las capas interact´uan entre s´ı por medio de una interfaz. Esta interfaz permite a la capa de l´ogica de negocio proveer los recursos que necesita la capa de presentaci´on. Las capas inferiores se encargar´an de brindar sus servicios a las capas superiores por medio de sus interfaces. Al separar una aplicaci´on en capas y niveles permitimos modificar de forma independiente cada capa. De acuerdo al nivel de complejidad de la aplicaci´on se pueden seguir incorporando capas o niveles de acuerdo a las necesidades. En nuestro caso requerimos una aplicaci´on que pueda interactuar con diferentes gestores de bases de datos. Para solucionar este problema agregamos otra capa en el nivel de aplicaci´on que se encargue del acceso a los datos (Figura 4.2) independientemente del tipo de gestor de la base de datos. Si en alg´un momento se necesita incorporar soporte para otro gestor de base de datos, s´olo necesitamos modificar la capa de acceso a datos haciendo transparente los cambios para las otras capas. Para poder agregar otra capa a nuestra arquitectura necesitamos identificar que componentes se pueden aislar de los dem´as. Este proceso lo podemos seguir hasta obtener una arquitectura que satisfaga las necesidades de la aplicaci´on. La cual en nuestro caso son las aplicaciones Web. En la arquitectura descrita en la Figura 4.2 es necesario recalcar que los componentes de la capa de l´ogica de negocios necesitan referenciar a instancias de las “clases del dominio”(las que representan las entidades del negocio). De la misma forma, los componentes de la capa de acceso a datos tambi´en tienen que referenciarlas para poder “rellenar”tales instancias con los datos que obtienen de las capas inferiores. Por tal motivo vemos como una nueva necesidad la capacidad de tener acceso a las clases de dominio desde cualquier capa de la aplicaci´on

**Patr´on modelo vista controlador**

Uno de los patrones de arquitectura que m´as se utilizan para el desarrollo de aplicaciones Web es el Modelo-Vista-Controlador (MVC) [23]. MVC es un patr´on que fue utilizado para construir las interfaces de usuario en el lenguaje Smalltalk-80 [24], siendo la aportaci´on m´as importante de este patr´on la separaci´on de los componentes relacionados con los datos de la aplicaci´on de los componentes de la interfaz de usuario. La separaci´on de las capas permite tener, a nivel de desarrollo, un c´odigo m´as claro, flexible y reusable. Durante el desarrollo de un patr´on multicapa se puede observar que cada capa tiende a encapsular elementos que comparten ciertas caracter´ısticas, creando capas que contienen componentes para una determinada funci´on. El patr´on MVC descompone la aplicaci´on en capas permitiendo tener una separaci´on entre la l´ogica de negocio de la aplicaci´on, la representaci´on y la persistencia. El patr´on MVC identifica tres capas que son importantes para cualquier aplicaci´on las cuales son: Modelo encapsula los datos de la aplicaci´on y la l´ogica para interactuar con ellos. Vista maneja la interacci´on con el usuario y la representaci´on del modelo. Controlador es el intermediario entre el modelo y la vista ante las peticiones generadas por el cliente en la vista. El controlador se encarga de seleccionar el modelo solicitado por el usuario y la vista adecuada para representarlo. La separaci´on que propone MVC por medio de las distintas capas se puede observar a nivel de dise˜no, ayudando a los dise˜nadores a identificar los componentes de cada capa y la comunicaci´on que existe con los dem´as componentes.

**Lenguaje de modelado unificado**

El problema que ahora surge es c´omo representar la arquitectura de un sistema de software, de tal manera que sea entendible por la mayor´ıa de las personas que est´an involucradas en el desarrollo.

Para hacer frente a este problema se propone el uso del lenguaje de modelado unificado (UML) para la representaci´on de las arquitecturas. UML es un lenguaje est´andar que nos permite modelar los componentes de un sistema de forma que sea entendible [1, 28] para todos los integrantes del equipo de desarrollo.

UML es un lenguaje de modelado que utiliza conceptos orientados a objetos y tiene una sintaxis y sem´antica bien definidas lo que nos permite utilizarlo en todas las etapas de desarrollo

La arquitectura puede tener diferentes significados para cada stakeholder. Por ejemplo un ingeniero de redes podr´ıa estar ´unicamente interesado en el hardware y la configuraci´on de la red que utiliza el sistema; un administrador de proyecto en los componentes clave a desarrollar y sus l´ıneas de tiempo; un desarrollador en clases que conforman un componente; y un tester en escenarios para localizar los posibles errores del sistema. Por lo tanto necesitamos m´ultiples puntos de vista para las distintas necesidades de los actores involucrados en el desarrollo de un sistema de software

**Cliente** Para el cliente lo ´unico que necesitamos es un navegador Web para poder acceder a nuestra aplicaci´on. Recordemos que se necesita de componentes externos (Plug-ins, librer´ıas) para tener una interfaz m´as rica en recursos multimedia.

**Presentaci´on** La presentaci´on considera todas las interfaces que van a interactuar directamente con el cliente. En nuestro caso tenemos una p´agina de inicio que va a contener cuatro presentaciones para el cliente. • Home: es la interfaz que se muestra por default al acceder a la aplicaci´on. Se puede representar por medio de una p´agina Web que contengo el nombre de la instituci´on, logotipo e informaci´on general de la instituci´on. Esta p´agina tiene comunicaci´on con las dem´as interfaces de usuario. • Cursos: es la interfaz encargada de mostrar informaci´on relacionada con los cursos que se imparten en la instituci´on educativa. Se tiene acceso desde la p´agina de Home porque no se necesita ning´un privilegio para acceder a esta informaci´on. • Registro: es la interfaz encargada de dar de alta al usuario dentro del sistema. Cuando el usuario desea acceder al sistema y no ha ingresado su login y password, el sistema mostrar´a esta interfaz. • Login: es la interfaz encargada de autenticar al usuario por medio de un nombre de usuario (user) y una contrase˜na (password).

**L´ogica de Negocio** La capa de l´ogica de negocio tiene todas las reglas necesarias para atender a las peticiones de los usuarios. En la Figura 5.22 se observan los componentes que se incluyen en esta capa y las dependencias que existen. En la Figura 5.23 se puede observar con m´as detalle la capa de l´ogica de negocio. Se puede observar las clases que lo conforman y la relaci´on que existe entre ellas.

**Datos** Los detalles de la capa de datos se muestran en la Figura 5.24. Dentro de los componentes tenemos un manejador de datos para la administraci´on de la informaci´on obtenida de la capa de persistencia. Dentro de las propiedades tenemos un conector para establecer la comunicaci´on con la capa de persistencia. Un datasource para especificar las fuentes de datos a las que vamos a tener acceso

<http://delta.cs.cinvestav.mx/~pmalvarez/tesis-tahuiton.pdf>