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

**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:**

Los usuarios necesitarán para la visualización del sistema únicamente su navegador web para poder acceder a la aplicación.

**Presentación:**

El patrón de arquitectura a utilizar para el desarrollo de la aplicación Web es el modelo-vista-controlador, puesto que la separación de las capas termite tener (a nivel desarrollo) un código más claro, flexible y reusable, el patrón modelo-vista-controlador identifica tres capas que son importantes para cualquier aplicación, las cuales son: Modelo encapusulado para los datos de la aplicación y la lógica para interactuar con ellos. Las vistas manejan la interacción con el usuario y la representación del modelo, y por último el controlador se encarga de seleccionar el modelo solicitado por el usuario y la vista adecuada para poder representarlo.

El patrón de arquitectura modelo-vista-controlador debe ser implementado mediante un framework, el framework para desarrollo en esta aplicación será laravel 5.

El sistema cuenta con interfaces que interactuarán con los usuarios, a continuación se enlistan y muestran las pantallas del sistema taskManager.

**Lógica de Negocio:** La aplicación taskManagerpresenta una solución para la administración de las labores de un equipo de desarrollo de software, facilita la comunicación entre los miembros del equipo(colaboradores) y su líder(jefe del proyecto) mediante la creación de proyectos y tareas específicas para cada proyecto, estas tareas son asignadas a cada colaborador, y cada colaborador trabaja sobre ellas, mediante la aplicación taskManager podrán también actualizar el estado de la tarea como un informe al jefe del proyecto, cada estado representa el nivel de progreso de dicha tarea, es decir, pendiente: si la tarea aún no comienza a realizarse. En progreso: si la tarea está en elaboración pero aún no concluye y Finalizado: si la tarea fue terminada en el periodo previsto, Además de suceder algún cambio en requisitos que afecte las tareas actuales asignadas, dichas tareas podrán ser modificadas al seleccionar la tarea y reajustarla a los cambios necesarios.

**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>