UNIVERSIDAD TECNOLOGICA ISRAEL

FACULTAD DE SISTEMAS INFORMATICOS

Análisis de la Arquitectura de desarrollo de sistemas N-Capas

Estudiante

Paul Enrique Villagómez Bermeo

Tutor

Ing. Pablo Tamayo

Quito - Ecuador

Diciembre 2012

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS

INFORMÁTICOS

CERTIFICADO DE RESPONSABILIDAD

Yo, Ing. Pablo Tamayo. Certifico que la Sr. Paul Enrique Villagómez Bermeo con C.C. No. 010490659-9 realizó la presente tesis con título "Análisis de la Arquitectura de desarrollo de sistemas N-Capas", y que es autor intelectual del mismo, que es original, autentica y personal.

_____

Ing. Pablo Tamayo

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS

INFORMÁTICOS

ACTA DE SESION DE DERECHOS

Yo, Paul Enrique Villagómez Bermeo, estudiante de Ingeniería de sistemas informáticos, declaro conocer y aceptar las disposiciones del Programa de Estudios, que en lo pertinente dice: "Es patrimonio de la Universidad Tecnológica Israel, todos los resultados provenientes de investigaciones, de trabajos científicos, técnicos o tecnológicos y de tesis o trabajos de grado que se realicen a través o con el apoyo de cualquier tipo de la Universidad Tecnológica Israel. Esto significa la cesión de los derechos de propiedad intelectual a la Universidad Tecnológica Israel".

_____

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS

INFORMATICOS

CERTIFICADO DE AUTORIA

El documento de tesis con títulos **"**Análisis de la Arquitectura de desarrollo de sistemas N-Capas**"** ha sido desarrollado por Paul Enrique Villagómez Bermeo con C.I. No. 010490659-9 persona que posee los derechos de Autoría y responsabilidad, restringiéndose la copia o utilización de cada uno de los productos de esta tesis sin previa autorización.

Paul Enrique Villagómez Bermeo

DEDICATORIA

La persistencia es una de las virtudes y fortalezas que he aprendido de mis padres, a los que debo todo el esfuerzo que realizan día a día para que yo culmine mis estudios, además están también quienes de manera directa o indirecta fueron aporte indispensable para el desarrollo cabal de mi preparación entre ellos: mis profesores que fueron senderos y luz en el camino de la preparación, mis hermanos que me demostraron que con la perseverancia se llega a la cumbre; por último, pero no menos importante al amor de mi vida cual ha estado presente en cada etapa de la realización de mis estudios, a ellos dedico mi constancia y mis deseos de seguir adelante.

AGRADECIMIENTO

Agradezco a Dios por darme una familia excelente que me apoya siempre, también por brindarme salud y bienestar a lo largo de mi vida y de mis estudios, también agradezco infinitamente a padres ya que ellos son los pilares más importantes en mi vida, a la Universidad Tecnológica Israel, la institución que me permitió crecer profesionalmente capacitándome día a día, finalmente a mis Amigos y Maestros, Ing. Pablo Tamayo, personas que intervinieron en el desarrollo de este documento.

RESUMEN

El modelo n capas es una arquitectura idónea para el desarrollo de aplicaciones ya que ha superado a las demás arquitecturas como la predominante para la construcción de aplicaciones especificaremos unas de las ventajas del modelo desarrollo en cada capa aplicaciones más robustas debido al encapsulamiento mientras tanto en el desarrollo de las aplicaciones veremos las ventajas y desventajas que obtienen al momento de desarrollarlas en n capas revisando el diseño y la forma estructural en la que está realizado su funcionamiento, se logra desarrollar aplicaciones de manera clara nítida y transparente esto es mediante componentes con la ventaja de poder desarrollar varias capas al mismo tiempo se logran aplicaciones robustas debido al encapsulamiento mantenimiento más sencillo separación adecuada de funciones además de tener mayor escalabilidad entre otros.

SUMMARY

N The model is an architectural layers suitable for the development of applications has passed since the other architectures as the predominant building applications the advantages specify a development model for each layer more robust due to encapsulation in the meantime application development will see the advantages and disadvantages that obtained when n layers develop in reviewing the design and structural form in which this operation performed, is able to develop applications in a clear and transparent clear this is to the advantage components layers can be developed simultaneously achieve robust applications due to easier maintenance encapsulation adequate separation of functions in addition to scalability among others.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE CUADROS

CAPITULO

I

1. INTRODUCCION

El presente Estudio se basa en Análisis de la Arquitectura de desarrollo de sistemas en el modelo n-tier (n-capas) de informática es una arquitectura muy dominante en el momento para la creación de aplicaciones.

Por medio del estudio se lograran saber el cambio radical en los modelos de computación, con esto sabremos sobre los sistemas y su desarrollo ya sea en monolíticos basados en mainframe los tradicionales sistemas cliente-servidor.

En lo cuan nos daremos cuenta que en la creación de sistemas nos da benéficos ya sea con los clientes por que nos deja trabajar con clientes ligeros sean navegadores y teléfonos inteligentes y otros más, por lo cual la creación en n-capas se está posicionando como la principal al momento del desarrollo de aplicaciones podemos decir que muchas compañías están realizando esto lo cual les hace crecer en la economía que tenga su base en red.

Con este estudio se generara alternativas para el desarrollo y benéficos que ofrece la creación de sistemas en dicho modelo.

1.1 Antecedentes

Con el tiempo las empresas se han vuelto más dependiente de los computadores y redes para el manejo de sus actividades, la alta disponibilidad de los sistemas informáticos se han vuelto de vital importancia en las empresas.

Actualmente, gran mayoría de las empresas necesitan un nivel alto de disponibilidad y algunas requieren incluso un nivel continuo de disponibilidad, ya que les resultaría extremadamente difícil funcionar sin los recursos informáticos.

A medida que vaya aumentando la dependencia de la disponibilidad de los recursos informáticos, este porcentaje seguramente crecerá.

Por lo tanto, la capacidad para recuperarse exitosamente de los efectos de un desarrollo de aplicación en n-capas para realizar este análisis es que después de un gran tiempo que las aplicaciones cliente servidor estuvieron en un apogeo con el pasar del tiempo y las apariciones de nuevas herramientas como el internet y los sistemas distribuidos vamos dándonos cuenta la innovación del desarrollo de aplicaciones

1.2 Formulación del problema

¿Al momento del desarrollo de aplicaciones en n-capas se toman las debidas precauciones para prevenir las eventualidades que están latentes todo el tiempo?

1.3 Sistematización

1.3.1 Diagnóstico

Dentro del país se a tomado en cuenta que no todas las instituciones que pueden utilizar aplicaciones en sistemas de n capas lo hacen ya que existen diferentes razones por las cuales no son utilizadas, diremos alguna por las cuales no lo hacen:

- Dentro del país algunas Instituciones como se dijo anteriormente, todavía mantienen el temor a la implementación primero del trabajo en n-capas y después el desarrollo de aplicaciones.
- También existe falta del modelo n_capas y el diseño de aplicaciones incluso de los recursos para hacerlo.
- Además el monto y el tiempo de inversión hace el que muchas empresas no lo realicen.

1.3.2 Pronóstico

- Con la mejora del trabajo en n-capas se ha visto un gran crecimiento de las empresas situación que no se da en nuestro país ya que existe un temor al trabajar en n-capas.
- Para realizar un buen trabajo en n-capas tenemos que tener suficiente conocimientos para realizar aplicaciones siendo este eficaz al momento del desarrollo.
- Para la innovación de las empresas en nuestro país con el desarrollo de aplicaciones en n-capas debe existir recursos y personal con el suficiente conocimiento para que muchos propietarios de empresas mejoren lo cual causa que veamos el avance en nuestro país y en las empresas.

1.3.3 Control del Pronóstico

Con este estudio de obtendremos alternativas para satisfacer las necesidades que tenemos en el momento de realizar aplicaciones en el modelo n-capas y nos da los inconvenientes más notables sea en el aprendizaje y el costo de el mismo.

1.4 Objetivos

1.4.1 Objetivo General

Elaboración de un documento sobre el Análisis de la Arquitectura de desarrollo de sistemas en el modelo n-capas.

1.4.2 Objetivos Específicos

- Describirlas aplicaciones en n-capa.
- Describir ventajas de desarrollo de aplicaciones en n-capas.
- Identifica Desventajas de desarrollo de aplicaciones en n-capas.

- Diagnosticar requerimientos de las aplicaciones en n-capas.

1.5 Justificación

1.5.1 Justificación Teórica

El desarrollo de este documento pretende dar a conocer a los desarrolladores o personas que tengan poco o no tengan conocimiento sobre la arquitectura de desarrollo de sistemas en el modelo n-capas, de una manera comprensible y fácil dando a conocer beneficios, características y las posibles debilidades que pueda tener.

1.5.2 Justificación Práctica

Con la presentación que ofrece este documento los desarrolladores aprenderán a utilizar de mejor manera el modelo n-capas al momento de desarrollar aplicaciones y escoger la arquitectura que se va a desempeñar la aplicación despejando dudas e inconvenientes para el desarrollador.

1.5.3 Justificación Metodológica

El presente proyecto será soportado por la investigación tecnológica, ya que con esta investigación podemos obtener un conocimiento útil para resolver uno o varios problemas concretos que puedan surgir principalmente en las necesidades de la sociedad.

Igualmente realizaremos la investigación campo teórico ya que al saber que empresas trabajan en n-capas podremos obtener información necesaria parta la realización del documento si tienen realizados aplicaciones en este modelo.

1.6 Alcance y Limitaciones

1.6.1 Alcance

Con la elaboración de este documento daremos a conocer y capacitar a los programadores de la carrera de sistemas, sobre el desarrollo de aplicaciones en n-capas así evitando inconvenientes que se puedan presentar al momento de desarrollar. A más de cumplir con los objetivos y limitaciones propuestos en este documento.

1.6.2 Limitaciones

En el Marco de Referencias (Capitulo II), que es el cuerpo de nuestra investigación, se dará a conocer la información y conceptos más importantes relacionados con nuestra temática.

En el Desarrollo (Capítulo III), se creara un documento sobre el desarrollo de aplicaciones en n_capas, la cual será un documento entregable que solo tendrá sustento teórico y no será implementado para la presentación del tema.

1.7 Estudios de Factibilidad

1.7.1 Técnica

El proyecto de investigación es variable ya que se utilizaran aspectos tecnológicos (pc, internet) y humanos para llegar a la culminación de el mismo

1.7.2 Operativa

El documento tiene como fin la capacitación de personas en el manejo Análisis de las Arquitectura de desarrollo de sistemas en el modelo n-capas, mejorando la interacción del usuario con la misma.

1.7.3 Económica

En este punto se detallara el monto de la parte financiera necesario para el desarrollo del proyecto que rodeara los 200.00 $ dólares los cuales se detallan a continuación.

Gastos	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Total
Transporte	5	5	5	5	5	25
Copias	5	5	5	5	5	25
Impresiones	5	5	5	5	5	25
Internet	6	6	5	5	5	27
Extras	15	15	15	15	10	70

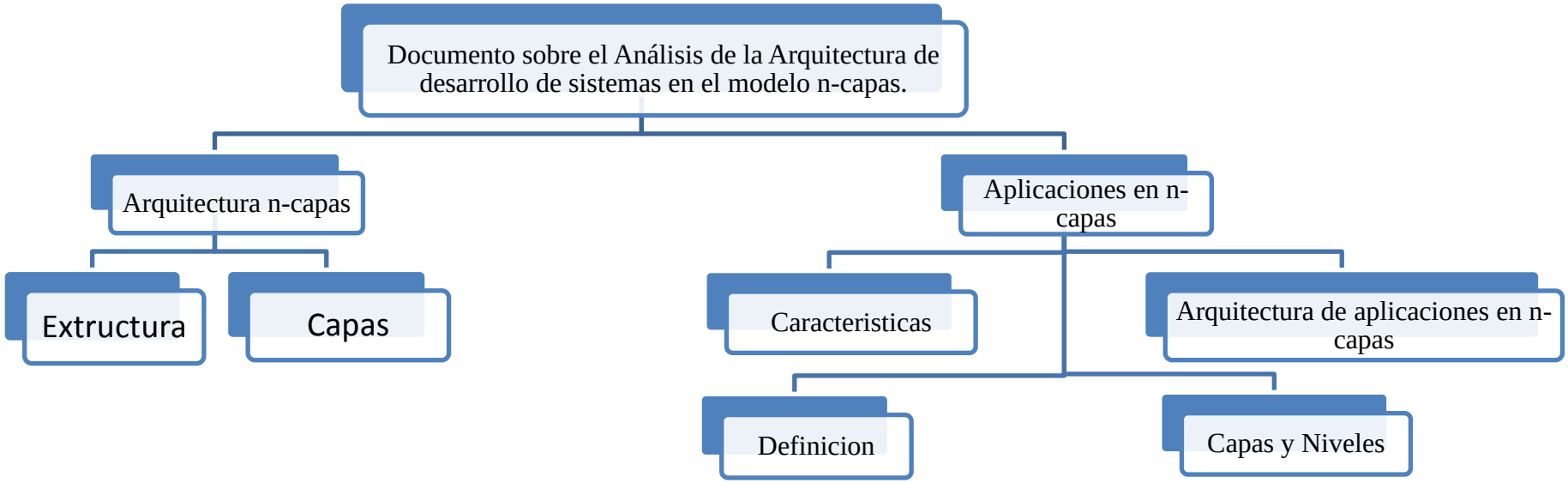
CAPITULO

II

2. MARCO DE REFERENCIA

2.1 Marco Teórico

Mapa conceptual de las teorías solo temas



2.2 Marco Conceptual

Introducción

El desafío actual enfrentado por los desarrolladores de aplicaciones es la implementación y el mejor desempeño de la aplicación es decir continuar produciendo sistemas más livianos y más rápidos y, la forma de administrar dichos sistemas o aplicaciones.

La industria ha realizado demasiados esfuerzos escogiendo una arquitectura idónea para la realización de aplicaciones en su modo de mantenimiento seguridad y por su puesto la interfaz con el usuario.

2.2.1 Aplicación

Figura 1 Aplicaciones
http://sanchezgomezcristina.blogspot.com/2010/04/software-de-aplicacion-una-aplicacion.html

Es un tipo de software que permite al usuario utilizar una computadora para realizar uno o más tipos de trabajo o también conocido más comúnmente como programas de computadoras.

Dichas aplicaciones pueden haber sido desarrolladas a medida para así satisfacer necesidades únicas de un usuario o formar parte de un solo paquete, las aplicaciones son parte del software de una computadora, y suelen ejecutarse sobre el sistema operativo.

Son series de intrusiones que dirigen una computadora para presentar una secuencia deseada de las operaciones, lo que le permitirá realizar una tarea específica.

Suele tener un único objetivo entre estas tenemos navegar en la web, revisar correo, explorar el disco duro, editar textos, juegos. Una aplicación que posee múltiples programas se considera un paquete en las que podemos nombrar a Internet Explorer, Outlook, Word, Excel.

2.2.2 Características de las aplicaciones

- ✓ Una aplicación puede ser desarrollada en cualquier lenguaje de programación.
- ✓ Las aplicaciones son programas compilados aunque a veces interpretado.
- ✓ Una aplicación en muchos casos tiene distintas licencias de distribución como: freeware, shareware, trialware.
- ✓ Las aplicaciones siempre se ejecutan en un tipo de interfaz sea esta una interfaz de texto o una interfaz gráfica varias ocupan ambas.

2.2.3 Arquitectura

Es la estructura en la que se encuentran las características que afectan el diseño y el desarrollo de programas de software, la organización interna de los componentes de equipos con particular referencia a la forma en que los datos se transmiten, el arreglamiento de los varios componentes de un sistema informático en la red.

2.2.4 Estilos de las Arquitecturas

Los estilos arquitecturales son la herramienta básica de un arquitecto a la hora de dar forma a la arquitectura de una aplicación. Un estilo de la arquitectura se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación.

Un estilo arquitectural viene definido por un conjunto de componentes, un conjunto de conexiones entre dichos componentes cualesquiera conectados.

Los estilos arquitecturales pueden organizarse en torno al aspecto de la aplicación sobre el que se centran.

Los principales aspectos son: comunicaciones, despliegue, dominio, interacción y estructura.

Lo normal en una arquitectura es que no se base en un solo estilo arquitectural, sino que combine varios de dichos estilos arquitecturales para obtener las ventajas de cada uno. Es importante entender que los estilos arquitecturales se “instancian” en una serie de componentes e interacciones concretas.

Esta es la principal diferencia existente entre un estilo arquitectural y un patrón de diseño. Los patrones de diseño son descripciones estructurales y funcionales de cómo resolver de forma concreta un determinado problema mediante orientación a objetos, mientras que los estilos arquitecturales son descripciones abstractas y no están atados a ningún paradigma de programación específico.

A continuación presentados los principales estilos arquitecturales con su descripción, características, principios, beneficios y los factores que pueden determinar su uso.

Aspecto	Estilos arquitecturales
Comunicaciones	SOA, message bus, Tuberías y filtros
Despliegue	Cliente / servidor , n – Niveles, N capas
Dominio	Modelo de dominio, repositorio
Interacción	Presentación separada

Tabla 1 Cuadro explicativo de estilos de arquitecturas

2.2.5 Arquitectura n-capas

Las arquitecturas empresariales de n-capas se están convirtiendo en la nueva base para el desarrollo de aplicaciones que se basa en una colocación jerárquica de los roles y responsabilidades que interactúan entre sí en varios niveles o capas para la división efectiva de los problemas a resolver, el modelo n-capas a emergido como la principal arquitectura para la construcción de aplicaciones multiplataforma de misión crítica y ofrecen la única arquitectura funcional para la siguiente generación de soluciones informáticas distribuidas basadas en Internet.

Los sistemas distribuidos de n-capas proporcionan un conjunto de avances tecnológicos sin precedentes, como pooling de conexiones,

multiplexado de conexiones, balanceo de carga dinámico y rendimientos excelentes en hardware trabajando en clúster.

Las aplicaciones industriales basadas en n-capas pueden ayudar a las compañías a desarrollar un nuevo núcleo de habilidades en prácticamente todo, desde la gestión del conocimiento hasta los sistemas relacionados con comercio electrónico.

N-Capas Como se ha podido ver hasta este momento, n-capas no es una tecnología, sino una estrategia de uso de las tecnologías para crear un negocio a la vez que se obtiene todo el potencial de éste el Internet.

La informática basada en n-capas no se refiere solamente al despliegue de clientes ligeros de bajo costo conectados a servidores de aplicaciones muy flexibles con balanceo de carga e integrados con bases de datos distribuidas existentes a lo largo de diferentes plataformas y localizaciones.

En realidad tiene que ver con la aplicación de las tecnologías relacionadas con desarrollos en n-capas para mejorar el conocimiento de los negocios y proveer un servicio de valor mediante la aplicación de esta avanzada tecnología como una solución para envolver oportunidades del mundo real.

El estilo de arquitectura basado en capas se identifica por las siguientes características:

- ✓ Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.

- ✓ Las capas de una aplicación pueden residir en la misma maquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- ✓ Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- ✓ Este modelo ha sido descrito como una "pirámide invertida de re-uso" donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.

Principios fundamentales

Los principios comunes que se aplican cuando se diseña para usar este estilo de arquitectura incluyen:

Abstracción. La arquitectura basada en capas abstrae la vista del modelo como un todo mientras que provee suficiente detalle para entender las relaciones entre capas.

Encapsulamiento. El diseño no hace asunciones acerca de tipos de datos, métodos, propiedades o implementación.

Funcionalidad claramente definida. El diseño claramente define la separación entre la funcionalidad de cada capa. Capas superiores como la capa de presentación envía comandos a las capas inferiores como la capa de negocios y la capa de datos y los datos fluyen hacia y desde las capas en cualquier sentido.

Alta cohesion. Cada capa contiene funcionalidad directamente relacionas con la tarea de dicha capa.

Reutilizable. Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.

Desacople. La comunicación entre las capas está basada en la abstracción lo que provee un desacople entre las capas.

2.2.6 Aplicaciones en n-capas

La construcción de aplicaciones n-tier (n-capas) distribuidas ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma en la mayor parte de las empresas.

Este cambio radical en los modelos de computación, desde los sistemas monolíticos basados en mainframe y los tradicionales sistemas cliente-servidor, hacia sistemas distribuidos multiplataforma altamente modularles, representa el desarrollo rápido y avance de la investigación en el mundo del desarrollo de aplicaciones, tal y como se pone de manifiesto en las últimas tendencias de las grandes empresas de tecnología, como Sun con su estrategia SunOne, o Microsoft con DotNET (.Net).

- ✓ Características
- ✓ Definición
- ✓ Capas y Niveles
- ✓ Arquitectura de Aplicaciones en n-capas

2.3 Marco Legal

En el marco del Software se realizó un foro para conocer los avances en la implementación del decreto presidencial #1014 que establece al software como prioritario para el sector público, al cumplirse más de 2 años de su ejecución.

Mario Albuja Subsecretario de Informática de la Presidencia de la República manifestó que al momento se avanza en un proyecto de ley para establecer el uso de Software en todas las instituciones del estado ecuatoriano, lo que se busca es implementar una herramienta que obligue a todas las instituciones a usar Software.

2.4 Marco Espacial

La investigación de este proyecto se la realizara en la ciudad de Cuenca con una duración de 3 meses una semana, empezando el 10 de septiembre con el diseño del proyecto hasta el 24 de noviembre que es la entrega de los trabajos empastados.

El proyecto va destinado a personas interesadas en la capacitación sobre el desarrollo de aplicaciones en n-capas

CAPITULO

III

3. Metodología

3.1 Proceso de Investigación

3.1.1 Unidad de Análisis

Es un tema dirigido a desarrolladores que deseen capacitarse y profundizar los conocimientos sobre el desarrollo de aplicaciones en n-capas, el cual se realizara mediante una investigación en el internet, de donde se tomara parte de la información para el desarrollo de esta temática.

3.1.2 Tipo de Investigación

Es de tipo explicativa para dar a conocer el desarrollo de aplicaciones en n-capas antes mencionada, y dar a conocer a los desarrolladores en forma de una documentación.

3.1.3 Método

El método de investigación que se aplicara en nuestro trabajo es el deductivo por ser un proyecto investigativo o de recopilación de información, el cual se utilizara en el desarrollo de la temática generando un módulo teórico para el uso de los desarrolladores interesados en el tema.

3.1.4 Técnica

Para desarrollar el presente proyecto nos basaremos en la técnica de observación se lo detallara brevemente en el punto 3.1.5 Instrumentos.

3.1.5 Instrumento

Los instrumentos utilizados fueron los siguientes:

- ✓ Investigar, recopilar información y bibliografía relacionada con el proyecto.

- ✓ Consultas que se la realizara al tutor, docentes de la institución, etc.
- ✓ Aplicación de los conocimientos aprendidos en el transcurso de ciclos anteriores.
- ✓ El desarrollo general en si del tema propuesto.

3.2 Características de las arquitecturas n-capas

Las aplicaciones n-capas proporcionan una gran cantidad de beneficios para las empresas que necesitan solución flexible y fiable para resolver complejos problemas inmersos en Cambios constantes

Entre las principales características de las arquitecturas n-capas tenemos:

- **Clientes ligeros**

  Todas las aplicaciones basadas en n-capas permitirán trabajar con clientes ligeros, tal como navegadores de internet, web tv, teléfonos inteligentes, pdas (asistentes personales digitales) y muchos otros dispositivos preparados para conectarse a internet

- **Red**

  Las arquitecturas basadas en n-capas permiten a los componentes de negocios correr en una LAN, WAN o internet. Esto significa que cualquiera con un ordenador y conexión a la red posee toda la funcionalidad que tendría si se encontrase delante de su sistema de escritorio

- **Subdivisión de sistemas**

  Los sistemas de n-capas subdivididos ayudan a facilitar el desarrollo rápido de aplicaciones y su posterior despliegue con beneficios incrementales fruto de los esfuerzos del desarrollo en paralelo coordinado y del outsourcing

inteligente resultado un enorme decremento del tiempo de desarrollo de sus costos

La arquitectura de n-capas provee flexibilidad rendimiento y seguridad en el diseño así como soporte para estándares de desarrollo abierto (independientemente de base de datos lenguaje o sistema operativo)

3.3 Estructura de la arquitectura n-capas

La arquitectura n-capas forma parte también de un revolucionario proceso basado en la aplicación de estas nuevas tecnologías (componentes y estándares de internet). Estas tecnologías son los bloques para crear software de negocios y sistemas de información adaptables que ayuden a las empresas a integrar todos sus sistemas de tecnología de la información así como las inversiones realizadas en estos mientras que obtienen una ventaja clara en el uso de internet.

La separación de las presentación lógica de negocio y de datos es realizada en un número indefinido de capas lógicas permitido a cada capa ser desarrollada mejorada gestionada y desplegada de forma independiente esta es precisamente la base para el modelo de informática de red en n-capas las plataformas multicapa funcionan consistentemente a lo largo de un simple portátil hasta un datacenter desde el dispositivo más simple hasta el más complejo de los mainframes

Figura 2 Arquitectura n-capas
http://icomparable.blogspot.com/2008/10/arquitectura-n-tier-o-arquitectura-n.html

En la imagen vemos la organización en n-capas donde se ve la capa de presentación la capa de negocios y la de acceso a datos

- **Capa de presentación**

  Es la encargada de los servicios de presentación proporciona interfaz necesaria para presentar información y reunir datos. También asegura los servicios de negocios necesarios para ofrecer las necesidades de transacciones requeridas e integrar al usuario con la aplicación para ejecutar un proceso de negocios

  Los servicios de presentación generalmente son identificados con la interfaz de usuario y normalmente residen en un programa ejecutable localizado en la estación de trabajo del usuario final. Aun así existen oportunidades para identificar servicios que residen en componentes separados.

  El cliente proporciona el contexto de presentación, generalmente el navegador como Microsoft Internet Explorer o Netscape Navigator, que

permiten ver los datos remotos a través de una capa de presentación HTML o también una aplicación WIN32 como son los formularios de Visual Basic

La capa de aplicaciones cliente se compone de aplicaciones cliente (como un pedido o mantenimiento de un producto) las cuales se crean a partir de componentes de aplicaciones cliente.

La capa de presentación es responsable de:

- ✓ Obtener información del usuario
- ✓ Enviar la información del usuario a los servicios del negocios para su procesamiento
- ✓ Recibir los resultados del procesamiento de los servicios de negocios
- ✓ Presentar estos resultados al usuario

- **Capa de negocios**

Se encarga de los servicios de negocios son el "puente" y los servicios de datos. Responde a peticiones del usuario (u otros servicios de negocios) para ejecutar una tarea. Cumplen con esto aplicando procedimientos formales y reglas de negocios a los datos relevantes. Cuando los datos necesarios residen en un servidor de base de datos, garantizan los servicios de datos indispensables para cumplir con la tarea de negocios o aplicar su regla. Esto aísla al usuario de la interacción directa con la base de datos

Una tarea de negocios es una operación definida por los requerimientos de la aplicación como introducir una orden de compra o imprimir una lista de clientes. Las reglas de negocios son políticas que controlan el flujo de las tareas.

Como las reglas de negocios tienden a cambiar más frecuentemente que las especificas de negocios a las que dan soporte, son candidatos ideales para

encapsularla en componentes que están lógicamente separados de la lógica de la aplicación en sí.

La capa del servidor se compone de servidores de negocios (como el proceso de órdenes y el manejo del almacén) la cual se crea a partir de componentes de aplicaciones de servidor de negocios.

El nivel de servicios de negocios es responsable de:

- ✓ Recibir la entrada del nivel de presentación.
- ✓ Interactuar con los servicios de datos para ejecutar las operaciones de negocios para los que la aplicación fue diseñada a auto misar (por ejemplo, la operación de impuestos por ingresos, el procesamiento de órdenes y así sucesivamente).
- ✓ Enviar el resultado procesado al nivel de presentación.

- **Capa de Acceso de datos**

La capa del servidor de datos se compone de servidores de datos (como órdenes y productos) que se crean a partir de componentes de servidores de datos. En esta capa es donde van a residir los datos, es también utilizada en la arquitectura cliente servidor.

El nivel de servicios de datos es responsable de:

- ✓ Almacenar los datos
- ✓ Recuperar los datos
- ✓ Mantener los datos
- ✓ La integridad de los datos

En una arquitectura tradicional, una capa puede comunicarse sólo con otra directamente arriba o debajo de ella. En este caso los servicios de usuarios,

de negocios y de servicios, dado que, lejos del comportamiento de un modelo de capas, cualquier servicio puede invocar a otro dentro de su capa.

Lo que realmente es nuevo en el modelo de n-capas es la posibilidad de distribuir objetos independientes sobre el número de capas que sean necesarias y enlazarlas dinámicamente, cuando sea necesario, para proporcionar una flexibilidad ilimitada a la aplicación.

3.4 Comparación entre arquitecturas

Para realizar un análisis y ver el por qué escoger la arquitectura hablaremos sobre algunas arquitecturas dándonos cuenta el por qué utilizar N-capas dando una conclusión y recomendación al final de la exposición de dichas arquitecturas en las que se hablara una breve descripción, características, principios claves, beneficios, cuando usarlos.

- **N-capas**

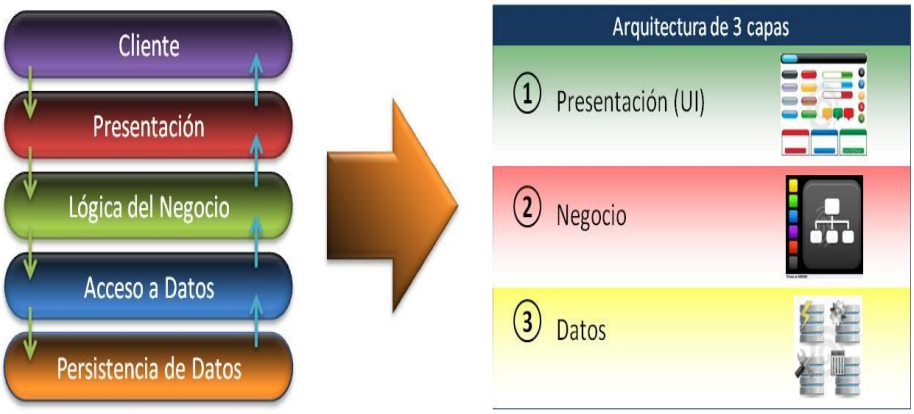


Figura 3 Estructura de la arquitecturas n-capas
http://www.icde.org.co/web/ide_gig/blogs/-/blogs/un-acercamiento-a-hibernate-spatial;jsessionid=D55E4CC607163FEE21926901464E931B

Descripción

Se basa en una distribución jerárquica para proporcionar una división efectiva de los problemas a resolver.

Características

- Descomposición de los servicios de forma que la mayoría de interacciones ocurre solo entre capas vecinas
- Las capas de una aplicación pueden residir en la misma máquina o pueden estar distribuidos entre varios equipos
- Los componentes de cada capa se comunica con los componentes de otras capas a través de interfaces bien conocidos
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior

Principios clave

- Muestra una vista completa del modelo y a la vez proporciona suficientes detalles para entender las relaciones entre capas
- No realiza ninguna suposición sobre los tipos de datos métodos propiedades y sus implementaciones
- Separa de forma clara la funcionalidad de cada capa
- Cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa
- Las capas inferiores no tienen dependencias de las capas superiores
- La comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre capas

Beneficios

- Abstracción ya que los cambios se realizan a alto nivel y se puede incrementar o reducir el nivel de abstracción que se usa en cada capa del modelo
- Aislamiento ya que se pueden realizar actualizaciones en el interior de las capas sin que esto afecte al resto del sistema

- Rendimiento ya que distribuye las capas en distintos niveles físicos se puede mejorar la escalabilidad la tolerancia a fallos y el rendimiento
- Testeailidad ya que cada capa tiene una interfaz bien definida sobre la que realizar las pruebas y la habilidad de cambiar entre diferentes implementaciones de una capa
- Independencia ya que elimina la necesidad de considerar el hardware y el despliegue así como las dependencias con interfaces externas

Cuando usarlo

- Ya tienes construidas capas de una aplicación anterior que pueden reutilizarse o integrarse
- Ya tienes aplicaciones que exponen su lógica de negocio a través de interfaces de servicios
- La aplicación es compleja y el alto nivel de diseño requiere la separación para que los distintos equipos puedan concentrarse en distintas áreas de funcionalidad
- La aplicación debe soportar distintos tipos de clientes y distintos dispositivos Quieres implementar reglas y procesos de negocios complejos o configurables

❖ **Cliente/Servidor**

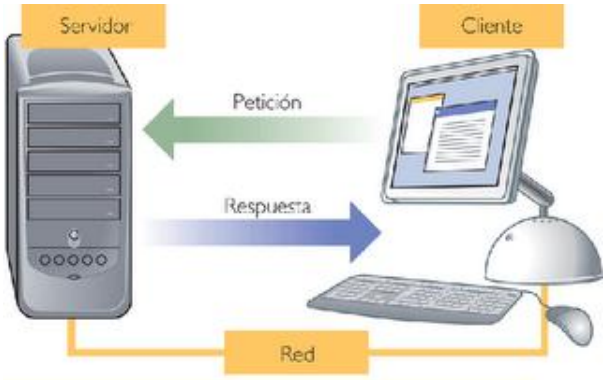


Figura 4 arquitecturas cliente servidor
http://arquitectura-magy.blogspot.com/

Descripción

Define una relación entre dos aplicaciones en las cuales una de ellas cliente envía peticiones a la otra servidor fuente de datos

Características

- Es un estilo para sistemas distribuidos
- Divide el sistema en una aplicación cliente una aplicación servidora y una red que las conecta
- Describe una relación entre el cliente y el servidor en la cual el primero realiza peticiones y el segundo envía respuestas
- Puede usar un amplio rango de protocolos y formatos de datos para comunicar la información

Principios clave

- El cliente realiza una o más peticiones espera por las respuestas y las procesa a su llegada
- El cliente normalmente se conecta solo a uno o a un número reducido de servidores al mismo tiempo

- El cliente interactúa directamente con el usuario por ejemplo a través de una interfaz grafica
- El servidor no realiza ninguna petición al cliente
- El servidor envía los datos en respuesta a las peticiones realizadas por los clientes conectados
- El servidor normalmente autentifica y verifica primero al usuario y después procesa la petición y envía los resultados

Beneficios

- Más seguridad ya que los datos se almacenan en el servidor que generalmente ofrece más control sobre la seguridad
- Acceso centralizado a los datos que están almacenados en el servidor lo que facilita su acceso y actualización
- Facilidad de mantenimiento ya que los roles y las responsabilidades se distribuyen entre los distintos servidores a través de la red lo que permite que un cliente no se vea afectado por un error en un servidor particular

Cuando usarlo

- La aplicación se basa en un servidor y soportara múltiples clientes
- La aplicación esta normalmente limitada a un uso local y área LAN controlada
- Estos implementados procesos de negocio que se usaran a lo largo de toda tu organización
- Quieres centralizar el almacenamiento backup y mantenimiento de la aplicación

- La aplicación debe soportar distintos tipos de clientes y distintos dispositivos

❖ **Basado en Componentes**

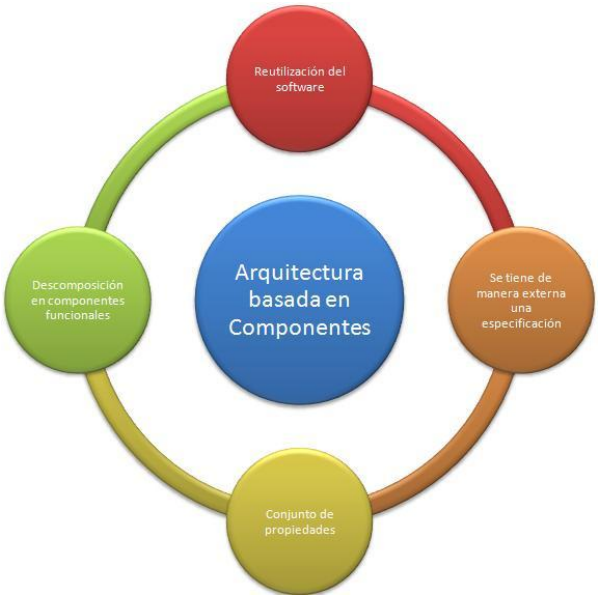


Figura 5 arquitectura basadas en componentes
http://es.wikibooks.org/wiki/Aplicaciones_Distribuidas:_Un_enfoque_pr%C3%A1ctico

Descripción

Describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema

Características

- Es un estilo para diseñar aplicaciones a partir de componentes individuales
- Enfatiza la descomposición del sistema en componentes con interface muy bien definidas
- Define una aproximación al diseño a través de componentes que se comunican mediante interfaces que exponen métodos eventos y propiedades

Principios clave

- Los componentes son diseñados de forma que puedan ser utilizados en distintos escenarios en distintas aplicaciones aunque algunos componentes son diseñados para una tarea específica
- Los componentes son diseñados para operar en diferentes entornos y contextos toda la información debe ser pasada al componente en lugar de incluirla en el o que este acceda a ella
- Los componentes pueden ser extendidos a partir de otros componentes para ofrecer nuevos comportamientos
- Los componentes exponen interfaces que permiten al código usar su funcionalidad y no revelan detalles internos de los procesos que realizan o de su estado
- Los componentes están diseñados para ser lo más independientes posible de otros componentes por lo que pueden ser desplegados sin afectar a otros componentes o sistemas

Beneficios

- Fácil despliegue ya que se puede sustituir un componente por su nueva versión sin afectar a otros componentes o al sistema
- Reducción de costos ya que se pueden usar componentes de terceros para abaratar los costos de desarrollo y mantenimiento
- Reusable ya que son independientes del contexto se pueden emplear en otras aplicaciones y sistemas
- Reducción de la complejidad gracias al uso de contenedores de componentes que realizan la activación gestión de ciclo de vida

Cuando usarlo

- Tienes componentes que sirvan a tu sistema o los puedes conseguir
- La aplicación ejecutara generalmente procedimientos con pocos o ningún dato de entrada
- Quieres poder combinar componentes escritos en diferentes lenguajes
- Quieres crear una arquitectura que permita reemplazar o actualizar uno de sus componentes de forma sencilla

❖ **Presentación Desacoplada**

Figura 6 arquitecturas presentación desacoplada
http://www.josecuellar.net/arquitectura-de-software/estilos-arquitecturales-en-el-diseno-de-un-sistema/

Descripción

Indica cómo debe realizarse el manejo de las acciones de usuario la manipulación de la interfaz y los datos de la aplicación este estilo separa los componentes de la interfaz del flujo de datos y de la manipulación

Características

- Es un estilo para diseñar aplicaciones basadas en patrones de diseño conocido
- Separa la lógica para el manejo de la interacción de la representación de los datos con que trabaja el usuario

- Permite a los diseñadores crear una interfaz gráfica mientras los desarrolladores escriben el código para su funcionamiento

Principios clave

- El estilo de presentación desacoplada separa el procesamiento de la interfaz en distintos roles
- Permite construir mocks que replican el comportamiento de otros objetos durante el testeo
- Usa eventos para notificar a la vista cuando hay datos del modelo que han sido modificados
- El controlador maneja los eventos disparados desde los controles de usuario en la vista

Beneficios

- Testeabilidad ya que en las implementaciones comunes los roles son simplemente clases que pueden ser testeadas y remplazadas por mocks que simulen su comportamiento
- Reusabilidad ya que los controladores pueden ser aprovechados en otras vistas compatibles y las vistas pueden ser aprovechadas en otros controladores compatibles

Cuando usarlo

- Quieres mejorar el testeo y el mantenimiento de la funcionalidad de la interfaz
- Quieres separar la tarea de crear la interfaz de la lógica que la maneja
- No quieres que la interfaz contenga ningún código de procesamiento de eventos

- El código de procesamiento de la interfaz no implementa ninguna lógica de negocios

❖ **Orientado a Objetos**

Figura 7 arquitecturas orientado a objetos
http://www.ufg.edu.sv/ufg/theorethikos/Julio04/mdp5.html

Descripción

Define el sistema como un conjunto de objetos que cooperan entre si en lugar de como un conjunto de procedimientos los objetos son discretos independientemente y poco acoplados se comunican mediante interfaces y permiten enviar recibir mensajes.

Características

- Es un estilo para diseñar aplicaciones basadas en un número de unidades lógicas y código reusable.
- Describe el uso de objetos que contienen los datos y el comportamiento para trabajas con esos datos y además tiene un rol o responsabilidad distinta.
- Hace hincapié en la reutilización a través de la encapsulación la modularidad el polimorfismo y la herencia.

- Contrasta con el acercamiento procedimental donde hay una secuencia predefinida de tareas y acciones el enfoque orientado a objetos interactuando unos con otros para realizar las tareas.

Principios clave

- Permite reducir una operación compleja mediante generalizaciones que mantienen las características de la operación.
- Los objetos se componen de otro objeto y eligen esconder estos objetos internos de otras clases o exponerlos como simples interfaces.
- Los objetos heredan de otros objetos y usan la funcionalidad de estos objetos base o redefinen dichas funcionalidad para implementar un nuevo comportamiento la herencia facilita el mantenimiento y la actualización ya que los cambios se propagan del objeto base a los herederos automáticamente.
- Los objetos exponen la funcionalidad solo a través de métodos propiedades y eventos y ocultan los detalles internos como el estado o las referencias a otros objetos esto facilita la actualización y el reemplazo de objetos asumiendo que sus interfaces son compatibles sin afectar al resto de objetos.

Beneficios

- Compresión ya que el diseño orientado a objetos define una serie de componentes mucho más cercanos a los objetos del mundo real.
- Reusabilidad ya que el polimorfismo y a la abstracción permiten definir contratos en interfaces y cambiar las implementaciones de forma transparente.

- Testeabilidad gracias a la encapsulación de los objetos
- Extensibidad gracias a la encapsulación el polimorfismo y la abstracción.

Cuando usarlo

- Quieres modelar la aplicación basándote en objetos reales y sus acciones.
- Ya tienes objetos que encajan en el diseño y con los requisitos operacionales.
- Necesitas encapsular la lógica y los datos juntos de forma transparente.

CAPITULO

IV

4. Resultados

4.1 Tipos de aplicaciones

Para la mejor forma de explicar los tipos de aplicaciones se a realizado un cuadro donde van a estar especificados la aplicación las ventajas que lleva y las consideraciones que se deben tomar en cuenta

Tipos de Aplicaciones	Ventajas	Consideraciones
Aplicaciones para dispositivos móviles	➢ Sirven en escenarios con conexión limitada ➢ Se puede llevar en dispositivos de mano ➢ Ofrecen alta disponibilidad y fácil acceso a los usuarios fuera de su entorno habitual	➢ Limitaciones a la hora de interactuar con la aplicación ➢ Tamaño de la pantalla reducido
Aplicación de escritorio	➢ Aprovechan mejor los recursos de los clientes ➢ Ofrecen la mejor respuesta a la interacción una interfaz más potente y mejor experiencia de usuario ➢ Proporcionan una interacción muy dinámica ➢ Soportan escenarios desconectados o con conexión limitada	➢ Despliegue complejo ➢ Versionado complicado ➢ Poca interoperabilidad
RIA (Rich Internet Applications)	➢ Proporcionan la misma potencia grafica que las aplicaciones de escritorio ➢ Ofrecen soporte para visualizar contenido multimedia ➢ Despliegue y distribución simples	➢ Algo más pesada que una aplicación web ➢ Aprovechan peor los recursos que las aplicaciones de escritorio ➢ Requieren tener instalado un plugin para funcionar
Aplicaciones orientadas a servicios	➢ Proporcionan una interfaz muy desacoplada entre cliente servidor ➢ Pueden ser consumidas por varias aplicaciones sin relación ➢ Son altamente interoperables	➢ No tienen interfaz grafica ➢ Necesitan conexión a internet
Aplicaciones web	➢ Llegan a todo tipo de usuarios y tienen una interfaz de usuarios estándar y multiplataforma ➢ Son fáciles de desplegar y de actualizar	➢ Depende de la conectividad a red ➢ No pueden ofrecer interfaces de usuarios complejas

Tabla 2 Cuadro explicativo de tipos de aplicaciones

Nombre de Sistema	Descripción	Arquitectura
Genesis 	Sistema Contable software ecuatoriano	CRM se define en (Cliente Servidor, 3 capas, n capas)
Soluciones técnicas integrales  *Soluciones Técnicas Integrales*	Contiene módulos contables producción de roles de pago y le genera automáticamente los anexos transaccionales sin necesidad de software	CRM se define en (Cliente Servidor, 3 capas, n capas)
Fénix 	Fénix es una herramienta enfocada solucionar problemas en el área contable y financiera y tributaria	CRM se define en (Cliente Servidor, 3 capas, n capas)
PC Gerente 	Herramientas contables	CRM se define en (Cliente Servidor, 3 capas, n capas)
Sitio web	Pedir solicitudes y consultas interrelación entre cliente y la otorgación del servicio Etapa http://www.etapa.net.ec/default.aspx	Modelo cliente servidor
Aplicación web **Guía interactiva**	Gestiona de áreas protegidas del ecuador ministerio de medioambiente	Arquitectura n capas

Tabla 3 Cuadro explicativo de tipos de aplicaciones y sus arquitecturas

4.2 Descripción de las arquitecturas n-capas

4.2.1 Arquitectura de aplicaciones

La arquitectura de aplicaciones estas permiten a las empresas disponer de un desarrollo y ejecución homogéneo es decir que sabremos cada función que cumple cada uno de los elementos dentro de una aplicación y que será común a todos los departamentos a la hora de ejecutar las aplicaciones.

Las arquitecturas nos indican la forma en que están distribuidas o diseñadas tanto físicamente como lógicamente que son:

- ✓ En el diseño físico nos indica exactamente donde se encontrarán las piezas de la aplicación como discos, ejecutables, cable de red y computadoras entere otros.
- ✓ En el diseño lógico se muestra la estructura de la aplicación y sus componentes sin tomar en cuenta dónde se localizará el software, hardware e infraestructura. Tales conceptos incluyen el orden de procesamiento, mantenimiento y seguimiento comunes en sistemas organizacionales.

4.2.2 Fundamentos de arquitectura de aplicaciones

El diseño de la arquitectura de un sistema es la forma en donde especificamos todos los requisitos sean estos técnicos y operacionales aquí vamos a definir cuales componentes van a formar el sistema y cómo van a ser relacionados esto nos especificara que mediante su interacción llevan a cabo sus funcionalidades especificas pero tomando muy en cuenta la eficiencia, seguridad y usabilidad

En el diseño de un sistema hay que ver los temas que nos indicara el éxito o fracaso unas preguntas que pueden hacer son:

- ✓ En que entorno va a ser desplegado nuestro sistema
- ✓ Como va a ser nuestro sistema puesto en producción
- ✓ Como van a utilizar los usuarios nuestro sistema
- ✓ Que otros requisitos debe cumplir el sistema (seguridad, rendimiento, concurrencia, configuración)
- ✓ Que cambios en la arquitectura pueden impactar al sistema ahora o una vez desplegado

Al momento de diseñar la arquitectura de un sistema es importante saber sobre lo que quieren cada uno de los agentes al momento de referirnos a los agentes decimos los usuarios del sistema y los objetivos que cumple el negocio y cada uno de los agentes in pone los requisitos y restricciones que necesitan y hay que saberlos para en un futuro no tener conflictos.

Para todos los usuarios es indispensable que el sistema responda rápidamente y mientras para la empresa tiene que costar poco.

Y por último en la arquitectura es asemejar los requisitos que brindara un impacto en la estructura del sistema y reducir los riesgos asociados con la construcción del sistema. La arquitectura debe soportar los cambios futuros del software, del hardware y de funcionalidad demandada por los clientes.

La arquitectura debería realizar lo siguiente:

- ➢ Mostrar la estructura del sistema pero ocultar los detalles.
- ➢ Realizar todos los casos de uso.

- Satisfacer en la medida de lo posible los intereses de los agentes.
- Ocuparse de los requisitos funcionales y de calidad.
- Determinar el tipo de sistema a desarrollar.
- Determinar los estilos arquitecturales que se usaran.
- Tratar las principales cuestiones transversales.

Una vez vistas las principales cuestiones que debe abordar el diseño de la arquitectura del sistema, ahora vamos a ver los pasos que deben seguirse para realizarlo. en una metodología ágil como scrum la fase de diseño de la arquitectura comienza durante en el pre-juego o en la fase de inicio en rup en un punto donde ya hemos capturado la visión del sistema que queremos construir en el diseño de la arquitectura lo primero que se decide es el tipo de sistema o aplicación que vamos a construir los principales tipos son aplicaciones móviles de escritorio, aplicaciones de servicios, aplicaciones que es importante entender que el tipo de aplicaciones viene determinado por la topología de despliegue y los requisitos y restricciones indicadas en los requisitos la selección de un tipo de aplicación determina en cierta medida el estilo arquitectural que se va a usar el estilo arquitectural es en esencia la partición más básica del sistema en bloques y la forma en que se relaciona estos bloques los principales estilos arquitecturales son cliente/servidor sistemas de componentes

Arquitectura en capas MVC, N-Niveles, SOA como ya hemos dicho el estilo arquitectural que elegimos depende del tipo de aplicación, una

aplicación que ofrece servicios lo normal es que se haga con un estilo arquitectural SOA

Por otra parte, a la hora de diseñar la arquitectura tenemos que entender también que un tipo de aplicación suele responder a más de un estilo arquitectural. Por ejemplo, una web hecha con ASP.NET MVC sigue un estilo Cliente/ Servidor pero al mismo tiempo el servidor sigue un estilo Modelo Vista Controlador.

Tras haber seleccionado el tipo de aplicación y haber determinado los estilos arquitecturales que más se ajustan al tipo de sistema que vamos a construir, tenemos que decidir cómo vamos a construir los bloques que forman nuestro sistema. Por ello el siguiente paso es seleccionar las distintas tecnologías que vamos a usar.

Estas tecnologías están limitadas por las restricciones de despliegue y las impuestas por el cliente. Hay que entender las tecnologías como los ladrillos que usamos para construir nuestro sistema. Por ejemplo, para hacer una aplicación web podemos usar la tecnología ASP.NET o para hacer un sistema que ofrece servicios podemos emplear WCF.

Cuando ya hemos analizado nuestro sistema y lo hemos fragmentado en partes más manejables, tenemos que pensar como implementamos todos los requisitos de calidad que tiene que satisfacer. Los requisitos de calidad son las propiedades no funcionales que debe tener el sistema, como por ejemplo la seguridad, la persistencia, la usabilidad, la mantenibilidad, etc.

Conseguir que nuestro sistema tenga estas propiedades va a traducirse en implementar funcionalidad extra, pero esta funcionalidad es la que otorga la funcionalidad básica del sistema.

Para tratar los requisitos de calidad el primer paso es preguntarse ¿Qué requisitos de calidad requiere el sistema? Para averiguarlo tenemos que analizar los casos de uso. Una vez hemos obtenido un listado de los requisitos de calidad las siguientes preguntas son ¿Cómo consigo que mi sistema cumpla estos requisitos? ¿Se puede medir esto de alguna forma? ¿Qué criterios indican que mi sistema cumple dichos requisitos? ¿Se puede medir esto de alguna forma? ¿Qué criterios indican que mi sistema cumple dichos requisitos?

Los requisitos de calidad nos van a obligar a tomar decisiones transversales sobre nuestro sistema. Por ejemplo, cuando estamos tratando la seguridad de nuestro sistema tendremos que decidir cómo se autentican los usuarios, como se maneja la autorización entre las distintas capas de nuestro sistema, etc.

De la misma forma tendremos que tratar otros temas como las comunicaciones, la gestión las excepciones, la instrumentación o el cacheo de datos.

Los procesos software actuales asumen que el sistema cambiara con el paso del tiempo y que no podemos saber todo a la hora de diseñar la arquitectura.

El sistema tendrá que evolucionar a medida que se prueba la arquitectura contra los requisitos del mundo real. Por eso, no hay que tratar de formalizar absolutamente todo a la hora de definir la

arquitectura del sistema. Lo mejor es no asumir nada que no se pueda comprobar y dejar abierta la opción de un cambio futuro. No obstante, sí que existirán algunos aspectos que podrán requerir un esfuerzo a la hora de modificarlos, donde para minimizar dichos esfuerzos es especialmente importante el concepto de desacoplamiento entre componentes.

Por ello es vital identificar esas partes de nuestro sistema y detenerse el tiempo suficiente para tomar la decisión correcta. En síntesis las claves son:

- Construir "hasta el cambio" más que "hasta el final".
- Utilizar herramientas de modelado para analizar y reducir los riesgos
- Utilizar modelos visuales como herramienta de comunicación
- Identificar las decisiones clave a tomar

A la hora de crear la arquitectura de nuestro sistema de forma iterativa e incremental, las principales preguntas a responder son:

- ¿Qué partes clave de la arquitectura representa el mayor riesgo si las diseño mal?
- ¿Qué partes de la arquitectura son más susceptibles de cambiar?
- ¿Qué partes de la arquitectura puedo dejar para el final sin que ello impacte en el desarrollo del sistema?
- ¿Cuáles son las principales suposiciones que hago sobre la arquitectura y como las verifico?
- ¿Qué condiciones pueden provocar que tenga que cambiar el diseño?

Como ya hemos dicho, los procesos modernos se basan en adaptarse a los cambios en los requisitos del sistema y en ir desarrollando la funcionalidad poco a poco.

En el plano del diseño de la arquitectura, esto se traduce en que vamos a ir definiendo la arquitectura del sistema final poco a poco.

Podemos entenderlo como un proceso de maduración, como el de un ser vivo.

Primero tendremos una arquitectura a la a que llamaremos línea base y que es una visión del sistema en el momento actual del proceso.

Junto a esta línea base tendremos una serie de arquitecturas candidatas que serán el siguiente paso en la maduración de la arquitectura.

Cada arquitectura candidata incluye el tipo de aplicación, la arquitectura de despliegue, el estilo arquitectural, las tecnologías seleccionadas, los requisitos de calidad y las decisiones transversales.

Las preguntas que deben responder las arquitecturas candidatas son:

- ¿Qué suposiciones he realizado en esta arquitectura?
- ¿Qué requisitos explícitos o implícitos cumple esta arquitectura?
- ¿Cuáles son los riesgos tomados con esta evolución de la arquitectura?
- ¿Qué medidas puedo tomar para evitar esos riesgos?
- ¿En qué medida esta arquitectura es una mejora sobre la línea base o las otras arquitecturas candidatas?

Dado que usamos una metodología iterativa e incremental para el desarrollo de nuestra arquitectura, la implementación de la misma debe seguir el mismo patrón.

La forma de hacer esto es mediante pruebas arquitecturales. Estas pruebas son pequeños desarrollos de parte de la aplicación (Pruebas de Concepto) que se usan para evitar riesgos rápidamente o probar posibles vías de maduración de la arquitectura.

Una prueba arquitectural se convierte en una arquitectura candidata que se evalúa contra la línea base. Si es una mejora, se convierte en la nueva línea base frente a la cual crear y evaluar las nuevas arquitecturas candidatas.

Las preguntas que debemos hacerle a una arquitectura candidata que surge como resultado de desarrollar una prueba arquitectural son:

- Introduce nuevos riesgos
- Soluciona algún riesgo conocido esta arquitectura
- Cumple con nuevos requisitos del sistema
- Realiza casos de uso arquitecturalmente significativos
- Se encarga de implementar algún requisito de calidad

Los casos de uso importantes son aquellos que son críticos para la aceptación de la aplicación o que desarrollan el diseño lo suficiente como para ser útiles en la evaluación de la arquitectura.

En resumen, el proceso de diseño de la arquitectura tiene que decidir que funcionalidad es la más importante a desarrollar.

A partir de esta decisión tiene que decidir el tipo de aplicación y el estilo arquitectural, y tomar las decisiones importantes sobre seguridad, rendimiento… que afectan al conjunto del sistema.

El diseño de la arquitectura decide cuales son los componentes más básicos del sistema y como se relacionan entre ellos para implementar la funcional. Todo este proceso debe hacerse paso a paso, tomando solo las decisiones que se puedan comprobar dejando abiertas las que no.

Esto significa disminuir los riesgos rápidamente y explorar la implementación de casos de uso que definan la arquitectura.

4.2.3 Desarrollo en N - Capas

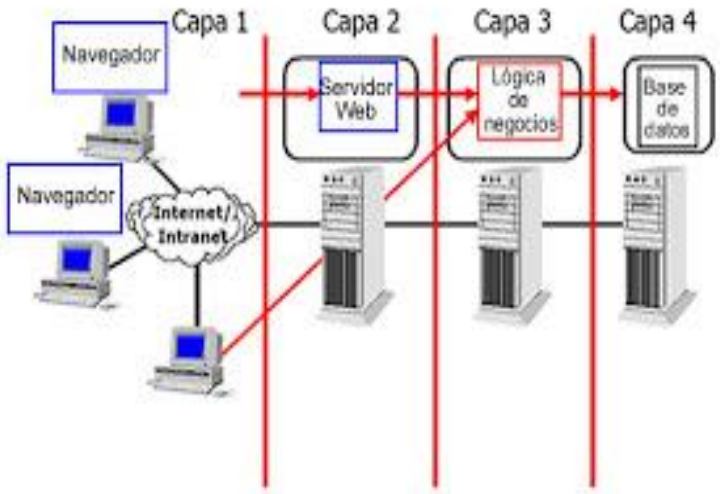


Figura 8 Desarrollo en N Capas
http://angeleser.blogspot.com/2011/02/113-aplicaciones-23-y-n-capas.html

El desarrollo en n-capas no es un producto o un estándar, es un concepto estratégico que ayuda a la construcción y despliegue lógico

de un sistema distribuido hablaremos sobre tres formas de desarrollarlos que son:

- Los sistemas de n-capas subdivididos ayudan a facilitar el desarrollo rápido de aplicaciones y su posterior despliegue, con beneficios incrementales fruto de los esfuerzos del desarrollo en paralelo coordinado y del outsourcing inteligente, resultando un enorme decremento del tiempo de desarrollo y de sus costas.
- Muchas de las aplicaciones de que se utilizan actualmente simplemente utilizan un navegador de Internet como cliente ligero que implementa una interfaz universal.
- Una arquitectura basada en clientes ligeros desplaza la capa de presentación de la aplicación en el lado del cliente, mientras que la lógica de negocio y los datos residen en el middleware y los servidores de back-end minimizando los problemas de despliegue de las aplicaciones y maximiza la accesibilidad desde varias plataformas.

Utilizando estos potentes estándares abiertos se permite asegurar la efectividad y consistencia de la comunicación y mensajería a todos los niveles en aplicaciones cruzadas Interdepartamentales e Inter empresariales.

El diseño de aplicaciones basado en n-capas considera a la red como un pool de servicios distribuidos, un concepto mucho más ambicioso que el simple acceso de un cliente a un servidor.

La separación de la presentación, lógica de negocio y datos es realizada en un número indefinido de capas lógicas, permitiendo a cada capa ser desarrollada y mejorada de forma independiente.

Esta es precisamente la base para el modelo de informática de red en n-capas.

Las plataformas multicapa funcionan consistentemente a lo largo de un variado conjunto de hardware, permitiendo escalar las operaciones del negocio desde un simple portátil, hasta un datacenter.

Es importante distinguir los conceptos de "Capas" (Layers) y "Niveles" (Tiers), pues es bastante común que se confundan y o se denominen de forma incorrecta.

Las Capas (Layers) se ocupan de la división lógica de componentes y funcionalidad y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lugares. Por el contrario, los Niveles (Tiers) se ocupan de la distribución física de componentes y funcionalidad en servidores separados, teniendo en cuenta topología de redes y localizaciones remotas.

Aunque tanto las Capas (Layers) como los Niveles (Tiers) usan conjuntos similares de nombres (presentación, servicios, negocio y datos), es importante no confundirlos y recordar que solo los Niveles (Tiers) implican una separación física.

Se suele utilizar el termino"Tier" refiriéndonos a patrones de distribución física como "2 Tier", "3- Tier" y "N- Tier"

Por último, destacar que todas las aplicaciones con cierta complejidad, deberían implementar una arquitectura lógica de tipo N-Capas

(estructuración lógica correcta), sin embargo, no todas las aplicaciones tienen porque implementarse en modo “N- Tier”, puesto que hay aplicaciones que no requieren de una separación física de sus niveles (Tiers), como pueden ser muchas aplicaciones web.

CAPAS

Si queremos diseñar una aplicación compuesta por un número considerable de componentes en capas debemos tomar en cuenta ciertos puntos:

- Realizar cambios de un tipo en una parte de la aplicación minimiza el impacto en otras partes, reduce el trabajo requerido en arreglar defectos, facilita el mantenimiento de la aplicación y mejora la flexibilidad general de la aplicación.
- Separación de responsabilidades entre componentes (por ejemplo, separar el interfaz de usuario de la lógica de negocio, y la lógica de negocio del acceso a la base de datos) aumenta la flexibilidad y escalabilidad.
- Ciertos componentes deben ser reutilizables entre diferentes módulos de una aplicación o incluso entre diferentes aplicaciones.
- Equipos diferentes deben poder trabajar en partes de la solución con mínimas dependencias entre diferentes equipos y deben poder desarrollar contra interfaces bien definidos.
- Los componentes individuales deben ser cohesivos
- Los componentes no relacionados directamente deben estar débilmente acoplados

- Los diferentes componentes de una solución de poder ser desplegados de una forma independiente, e incluso mantenidos y actualizados en diferentes momentos.
- Para asegurar una estabilidad, la solución debe poder probarse de forma autónoma cada capa.

Las capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o el servicio. Nos ayudan a diferenciar entre los diferentes tipos de tareas a ser realizadas por los componentes, ofreciendo un diseño que maximiza la reutilización y especialmente la mantenibilidad.

En definitiva se trata de aplicar el principio de 'Separación de Responsabilidades' (Soc.- Separation of Concerns principle) dentro de una Arquitectura.

Cada capa lógica de primer nivel puede tener un número concreto de componentes agrupados en sub-capas. Dichas sub-capas realizan a su vez un tipo específico de tareas.

Al identificar tipos genéricos de componentes que existen en la mayoría de las soluciones, podemos construir un patrón o mapa de una aplicación o servicio y usar dicho mapa como modelo de nuestro diseño.

El dividir una aplicación en capas separadas que desempeñan diferentes roles y funcionalidades, nos ayuda a mejorar el mantenimiento del código, nos permite también diferentes tipos de despliegue y sobre todo nos proporciona una clara delimitación y

situación de donde debe estar cada tipo de componente funcional e incluso cada tipo de tecnología

4.2.4 Diseño básico de Capas

La clave de una aplicación en N- Capas está en la gestión de dependencias, los componentes de una capa pueden interactuar solo con componentes de la misma capa o bien con otros componentes de capas inferiores. Esto se da por los componentes de la solución en capas.

Los componentes de cada capa deben ser cohesivos y aproximadamente el mismo nivel de abstracción.

Cada capa de primer nivel debe de estar débilmente acoplada con el resto de capas de primer nivel. Comenzar en el nivel más bajo de abstracción, por ejemplo Layer 1. Este es la base del sistema. Se continúa esta escalera abstracta con otras capas (Layer J, Layer J-1) hasta el último nivel (Layer- N).

4.2.5 N – Capas a pruebas

Una aplicación en N- Capas mejora considerablemente la capacidad de implementar pruebas de una formar apropiada:

- Debido a que cada capa interactúan con otras capas solo mediante interfaces bien definidos, es fácil añadir implementaciones alternativas a cada capa (Mock y Stubs). Esto permite realizar pruebas unitarias de una capa incluso cuando las capas de las que depende no están finalizadas o incluso porque se quiera poder ejecutar mucho más rápido un conjunto muy grande de pruebas unitarias que al acceder a las

capas de las que depende se ejecutan mucho más lentamente. Esta capacidad se ve muy mejorada si se hace uso de clases base (Patron 'Layer Supertype') interfaces (Patron 'Abstract Interface'), porque limitan aún más las dependencias entre las capas. Especialmente es importante el uso de interfaces pues nos dará la posibilidad de utilizar incluso técnicas más avanzadas de desacoplamiento, que exponemos más adelante en esta guía.

- Es más fácil realizar pruebas sobre componentes individuales por que las dependencias entre componentes están limitadas de forma que los componentes de capas de alto nivel solo pueden interaccionar con componentes en niveles inferiores esto ayuda a aislar componentes individuales para poder probarlos adecuadamente y nos facilita el poder cambiar unos componentes de capas inferiores por otros diferentes con un impacto muy pequeño en la aplicación siempre y cuando cumplamos los mismos interfaces

4.2.6 Beneficios de usos de capas

- El mantenimiento de mejoras en una solución será mucho más fácil porque las funciones están localizadas y además las capas deben estar débilmente acopladas entre ellas y con alta cohesión internamente lo cual posibilita variar de una forma sencilla diferentes implementaciones o combinaciones de capas

- Otras soluciones debería poder reutilizar funcionalidad expuesta por las diferentes capas especialmente si se han diseñado para ello
- Los desarrollos distribuidos son mucho más sencillos de implementar si el trabajo se ha distribuido previamente en diferentes capas lógicas
- La distribución de capas en diferentes niveles físicos pueden en algunos casos mejorar la escalabilidad aunque este punto hay que evaluarlo con cuidado pues puede impactar negativamente en el rendimiento

4.2.7 Proceso n-capas

En la forma de realizar un proceso en n-capas mientras más se efectúa un proceso operativo esto provoca muchas necesidades las cuales crecen haciendo desbordar el soporte de la maquina por lo que se separa la estructura de un programa en varias capas dando una mejor distribución, mejor manejo y disponibilidad del programa, por lo que en la actualidad la programación en capas es un estilo de programación en la que el objetivo principal es separar la lógica de negocios de la lógica de diseño

Para mejor entendimiento un ejemplo en el funcionamiento de un programa que emplee usuarios dividiríamos por capas ya sea la capa de presentación al usuario separada de la capa de datos y así misma de la capa de negocios dando así un diseño muy empleado en la actualidad

4.2.8 La evolución

Las arquitecturas basadas en n-capas son el siguiente paso lógico en un proceso de evolución, el cual está basado en las arquitecturas convencionales cliente-servidor (2 y 3 capas) más la convergencia de dos tecnologías tremendamente potentes:

- Desarrollo de aplicaciones basadas en componentes - relacionado directamente con la Programación Orientada a Objetos (Lenguajes y Técnicas)
- Internet - primer ejemplo de un sistema complejo de n-capas cliente-servidor.

Los sistemas de n-capas utilizan técnicas de desarrollo basadas en componentes combinados con los estándares abiertos de Internet, para crear aplicaciones multiplataforma muy potentes con bajos costes, fáciles de mantener y con gran efectividad. Lo que realmente es nuevo en el modelo de n-capas es la posibilidad de distribuir objetos independientes sobre el número de capas que sean necesarias y enlazarlas dinámicamente, cuando sea necesario, para proporcionar una flexibilidad ilimitada a la aplicación.

4.2.9 Arquitectura en n-capas: Un sistema adoptivo

N-Tier forma parte también de un revolucionario proceso, actualmente en desarrollo, basado en la aplicación de estas nuevas tecnologías (componentes y estándares de Internet). Estas tecnologías son los bloques para crear Software de Negocio y Sistemas de Información adaptables que ayuden a las empresas a integrar todos sus sistemas de

Tecnologías de la Información, así como las inversiones realizadas en éstos, mientras que obtienen una ventaja clara en el uso de Internet.

Las empresas exitosas del futuro serán aquellas que se adapten mejor a un mundo conectado. Los framework de n-capas utilizan herramientas basadas en Internet que proporcionan a los clientes la adopción de las últimas y más potentes tecnologías que proporcionarán claros avances competitivos. Las empresas hoy en día (no importa dónde estén, qué tamaño tengan o en qué industria se encuentren) deben ser capaces de implementar las últimas prácticas de negocio, ventas y estrategias de distribución, procesos de fabricación, logística de la cadena de suministro, etc.

Por eso, los sistemas basados en n-capas ayudan rápidamente a cambiar los negocios para experimentar la compartición sin restricciones de datas a lo largo de aplicaciones o fuentes de datos en la empresa, incluyendo Enterprise Resource Planning (ERP), aplicaciones hechas a medida, empaquetadas, heredadas o bases de datos.

4.3 Ventajas de aplicaciones en N - Capas

- Escalabilidad de las aplicaciones: permiten distribuir la carga esto es el momento que realiza muchas peticiones pero lo hace con el mismo rendimiento por medio de varios servidores, lo que además permite hacer un balance de carga.
- Simplifican el acceso a sistemas heterogéneos: existen servicios especializados que hacen posible la comunicación e integración de aplicaciones.

- Desarrollo de aplicaciones Web: están orientadas al desarrollo de aplicaciones sobre Internet.
- Soporte de varios lenguajes de programación: en la arquitectura de las aplicaciones distribuidas cada componente que la forma puede ser creado independientemente de los otros.
- Facilidad de administración de la aplicación: los servicios están centralizados facilitando el desarrollo y mantenimiento de la aplicación.
- Administración eficiente de recursos: están orientadas a administrar los recursos de forma que las aplicaciones puedan ser escalables.
- Implementación de seguridad: como la aplicación se comunica por medio de la red la seguridad es un aspecto muy importante y puede ser configurada según las necesidades de la aplicación.

4.4 Desventajas de las aplicaciones en N – Capas

- Conocimiento en diversas tecnologías: requiere que el equipo de desarrollo de la aplicación tenga conocimientos en varias ramas. Por ejemplo, la integración de una arquitectura mainframe y una aplicación Web requiere de un experto en el desarrollo de aplicaciones mainframe, un experto en la conexión para la integración entre la aplicación Web y el sistema mainframe y, además, un experto en herramientas de desarrollo de aplicaciones Web.
- Evaluación de varias plataformas: se necesitan realizar evaluaciones en cada caso para determinar cuál es la mejor plataforma para aplicaciones distribuidas entrando en un proceso de investigación para conocer a profundidad los detalles y escoger la mejor opción según las necesidades que se tengan.

CAPITULO

V

5. Conclusiones

- Al momento de especificar qué aplicaciones pueden ser ejecutadas en n-capas se puede decir que son las aplicaciones de despliegue, esto quiere decir que ésta puede ser usada de distintos lados y cada módulo dando un soporte indistintamente ya que la arquitectura n capas permite realizarla. Dentro de estas aplicaciones están: las aplicaciones ricas en internet o también llamadas RIA(Rich Internet Applications), las aplicaciones orientadas a objetos y aplicaciones web. Esta arquitectura es una nueva base para el desarrollo de aplicaciones, es la mejor y más utilizada. En el momento se dice decir que ha emergido como la principal arquitectura para la construcción de aplicaciones multiplataforma de misión crítica y ofrecen la única arquitectura funcional para la siguiente generación de soluciones informáticas distribuidas basadas en Internet.
- En las ventajas de las aplicaciones n-capas la más notable es la escalabilidad ya que permite repartir los pedidos que realiza la aplicación, esto no quiere decir que el rendimiento de la aplicación varíe ya que se ocupa la estructura n capas en la que se estructura por varios servidores y esto permite realizar el balance de la carga. Con la ayuda de n-capas se hace más fácil el desarrollo de aplicaciones en la web, muchas de las arquitecturas trabajan en un mismo lenguaje de programación pero ese no es el caso de n capas ya que cada componente puede ser desarrollado en un distinto lenguaje el cual le hace independiente de cada uno, esto otorga una fácil administración ya sea en seguridad, en mantenimiento y eficiencia, siendo una arquitectura donde una aplicación puede ser la mejor desempeñada.

- Como desventajas: en el desarrollo de aplicaciones con base en la arquitectura n-capas se debería tener personas especialistas en desarrollo que cuenten con conocimiento de varias ramas, es decir al menos saber desarrollar software en diversas tecnologías, a pesar de esto la variación más notable será el escoger la plataforma en la que se va a situar la aplicación, esto se escogerá mediante una investigación de la arquitectura para cada aplicación buscando la más óptima para ello.

6. Recomendaciones

- Un buen ejecuta miento al momento de la implementación para así obtener el mejor funcionamiento ya sea en la escalabilidad y su mejor mantenimiento
- Contar con personas idóneas al momento del diseño del software para evitar dificultades al desarrollar la aplicación y más en su implementación.

7. Glosario

A

Agentes o usuarios de sistema: Los usuarios del sistema o usuarios cliente son personas que se conectan al sistema para hacer uso de los servicios que este les proporciona.

Arquitectura: Modo en que están organizados e interconectados los diversos elementos que constituyen un sistema de computación

Aplicaciones: Cada uno de los programas que una vez ejecutados permiten trabajar con el ordenador. Son aplicaciones los procesadores de textos, hojas de cálculo, bases de datos, programas de dibujo, etc.

ASP.NET: Paginas Activas del Servidor.

B

Back-end: Procesador que se utiliza para determinada función muy especializada, como por ejemplo, administrar una base de datos

Back-up: Copia de seguridad de los ficheros o aplicaciones disponibles en un soporte magnético (generalmente disquetes), con el fin de poder recuperar la información y las aplicaciones en caso de una avería en el disco duro, un borrado accidental o un accidente imprevisto.

Base de datos: Conjunto de datos organizados de modo tal que resulte fácil acceder a ellos, gestionarlos y actualizarlos.

C

Clúster: En la tecnología de las computadoras, un cluster es la unidad de almacenamiento en el disco rígido. Un archivo está compuesto por varios clusters, que pueden estar almacenados en diversos lugares del disco.

Cliente-Servidor: (Client/Server Network). Red de comunicaciones que utiliza servidores dedicados para todos los clientes en la red. Nótese la diferencia con peer-to-peer network, que permite que cualquier cliente sea también un servidor.

D

Desbordar: Salir del contenido de un recipiente por los bordes, Exceder los límites o la capacidad de una persona o cosa

Dotnet: Es una "plataforma de Software", en este sentido se define como un ambiente donde pueden interrelacionar diversos componentes independientemente del lenguaje

E

ERP (Enterprise resource planning): Los sistemas de planificación de recursos empresariales (en inglés ERP, Enterprise Resource Planning) son sistemas de gestión de información que automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa.

F

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado

Freeware: Software de distribución gratuita. Programas que se distribuyen a través de Internet de forma gratuita.

H

HTML: Hyper Text Mark-up Language. Lenguaje de programación para armar páginas web.

Hardware: Conjunto de componentes materiales de un sistema informático. Cada una de las partes físicas que forman un ordenador, incluidos sus periféricos. Maquinaria y equipos (CPU, discos, cintas, modem, cables, etc.

I

Infraestructura: Conjunto de elementos o servicios que se consideran necesarios para el funcionamiento de una organización o para el desarrollo de una actividad.

Interfaz gráfica: Es la que permiten una mejor y más fácil interacción con el ordenador. Los interfaces gráficos -Windows es el ejemplo típico- permiten el aprendizaje intuitivo de los programas, facilitando y reduciendo el tiempo de formación y aumentando la productividad.

L

Layer supertypeo capas súpertype: Un tipo que actúa como el súper tipo de todos los tipos en su capa. No es raro que todos los objetos de una capa de contar con métodos que no desea que se han duplicado en todo el sistema. Puede mover todo este comportamiento en un súper tipo común Capa.

Layer -n o n-capas: Se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver.

LAN: (Local Area Network). Red de área local. El término LAN define la conexión física y lógica de ordenadores en un entorno generalmente de oficina. Su objetivo es compartir recursos (como acceder a una misma impresora o base de datos) y permite el intercambio de ficheros entre los ordenadores que componen la red. Los servidores son máquinas de alta velocidad que contienen programas y datos que comparten todos los usuarios de redes. Las estaciones de trabajo, o clientes, son los computadores personales de los usuarios, que realizan procesamiento autónomo y tienen acceso a los servidores de la red según se requiera. El software de control en una LAN es el sistema operativo de la red como NetWare, UNIX y Appletalk que reside en el servidor. En cada cliente reside una parte componente del software y permite que la aplicación lea y escriba datos del servidor como si estuviera en la máquina local. La transferencia de mensajes es administrada por un protocolo de transporte como IPX, SPX y TCP/*IP*. La transmisión física de datos es realizada por el método de acceso (Ethernet, Token Ring, etc.) que se implementa en los adaptadores de la red y que conectan a las máquinas. La vía de acceso real de las comunicaciones es el cable (par trenzado, cable coaxial, fibra óptica) que interconecta cada adaptador de red.

M

Modelo vista controlador(MVC): Es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.

Mock: Es un objeto dentro de la programación se los llama objetos de imitación son objetos simulados que imitan el comportamiento de los objetos reales de forma controlada.

Middleware: Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones

Monolíticos: Son aquellos en los que su centro es un grupo de estructuras fijas, las cuales funcionan entre sí.

Multiplataforma: Es la mescla de plataformas o soporta la ejecución de distintos programas realizados en otras plataformas

N

N niveles: Son aplicaciones centradas en datos separadas en varias capas lógicas (o niveles). Es decir, una aplicación de datos con n niveles es una aplicación que se separa en varios proyectos

O

Outsourcing; Es un tipo de relación contractual que permite a una organización cualquiera el uso de las instalaciones físicas de otra que, además, provee a la primera de mantenimiento y desarrollo de aplicaciones, proceso de datos, gestión de comunicaciones, cte. Así, una empresa (la segunda, según la definición anterior), aprovecha, por ejemplo, el sobrante de potencia de proceso de sus máquinas, o rentabiliza su experiencia en la gestión de sus propias aplicaciones, o comparte su estructura de comunicaciones, etc. La primera, en estas circunstancias, no necesita acometer costosísimas inversiones para resolver sus necesidades de operación.

P

PDA: (Personal Digital Assitant). Asistentes digitales personales. Es un pequeño ordenador que cabe en el bolsillo. Se utilizan como agenda y como bloc de notas.

Plataforma: Es un término de carácter genérico que designa normalmente una arquitectura de hardware, aunque también se usa a veces para sistemas operativos o para el conjunto de ambos. Los ordenadores VAX de la firma Digital, por ejemplo, serían una plataforma en la que se pueden soportar aplicaciones que, a su vez, corren en otras plataformas.

R

RIA rich internet application: O aplicaciones de Internet enriquecidas son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales.

Rup: El Proceso Unificado de Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software

S

Stubs: Es un objeto dentro de la programación se los llama objetos de imitación son objetos simulados que imitan el comportamiento de los objetos reales de forma controlada.

Scrum: Es un marco de trabajo para la gestión y desarrollo de software basada en un proceso iterativo e incremental utilizado comúnmente en entornos basados en el desarrollo ágil de software.

Soa: Una arquitectura orientada a servicios (SOA) es un conjunto de principios y metodologías para el diseño y desarrollo de software en forma de interoperabilidad de servicios.

Sunone: Es una marca con la que Sun Microsystems utiliza para comercializar productos de software de servidor.

Software: Es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

Shareware: Las versiones de programas que reciben esta denominación permiten probar sus capacidades sin realizar el desembolso mucho mayor que representaría comprar el programa convencional completo.

T

Topología: Se define como una familia de comunicación usada por los computadores que conforman una red para intercambiar datos. El concepto de red puede definirse como conjunto de nodos interconectados

W

Wcf: Wondows communication foundation (WFC) es un marco de trabajo para la creación de aplicaciones orientadas a servicios

Win32: El win32 se refiere a la carpeta donde se alhoja tu sistema operativo y los controladores de tu computadora

WAN: Cualquier red pública es de este tipo. Su característica definitoria es que no tiene límites en cuanto a su amplitud. Existen redes privadas de gran cobertura soportadas en estructuras físicas que son propiedad de operadores nacionales o internacionales.

8. Bibliografía

Software y Aplicaciones web

Fuente: http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx

Conceptos básicos

Autor: Mark Wagner

Obtenido 28 de octubre del 20012

Blog ángeles

Fuente: http://angeleser.blogspot.com/2011/02/113-aplicaciones-23-y-n-capas.html

Imagines y conceptos

Obtenido 29 de octubre del 2012

Foros web

Fuente:http://www.forosdelweb.com/f14/modelo-3-capas-arquitectura-n-capas-122898/

Autor: Ing. Javier Ugalde

Estructura conceptos

Obtenido 2 de noviembre del 2012

Rincón del vago

Conocimiento del desarrollo en n capas

Fuente: http://html.rincondelvago.com/desarrollo-de-n-capas.html

Obtenido 4 de noviembre del 20012

Desarrollo en .NET

Fuente: http://geeks.ms/blogs/dcerredelo/archive/2012/04/02/gu-237-a-de-arquitectura-n-capas-ddd-net-4-0.aspx

Arquitectura N Capas

Obtenido 4 de noviembre del 20012

Autor: Darío Cerredelo

Arquitectura n capas

http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx

Obtenido 4 de noviembre del 2012

Autor: Juan Peláez

. NET

Autor: NET

Guía de n capas

http://www.etnassoft.com/biblioteca/guia-de-arquitectura-n-capas-ddd-net-4-0/

Obtenido 4 de noviembre del 2012

Buenas Tareas

Arquitectura n capas

http://www.buenastareas.com/ensayos/Arquitectura-n-Capas/2752158.html

Obtenido 5 de noviembre del 2012

ANEXOS