**CURRÍCULOS EXPLORATORIOS**

**DESARROLLO E IMPLEMENTACIÓN DE APLICACIONES WEB.**

**UNIDAD 1: REQUISITOS DE SOFTWARE.**

**Lección 1: Fundamentos**

Antes de comenzar debemos tener claro que es un requerimiento. Si vamos al diccionario de la lengua española y buscamos requisito este se define como: *Circunstancia o condición necesaria para algo.* Pero esta definición se queda muy corta y además muy resumida porque en el campo del desarrollo web un requerimiento es mucho más que eso.

Para entender de forma clara supongamos que vamos a comenzar a generar un desarrollo, lo primero que pensamos es: ¿qué lenguaje vamos a usar?, ¿qué tipo de motor de base de datos usaremos?, ¿qué estilos de *CSS* usaremos? ¿qué es lo que queremos desarrollar? y así sucesivamente a nuestra cabeza comienzan a llegar ideas, solicitudes, preguntas, condiciones como por ejemplo queremos que sea azul, queremos que tenga 3 secciones, queremos que tenga un tipo de letra arial y sin percatarlo nosotros mismos estamos realizando requerimientos.

A continuación estableceremos como definición de requisito:

*Un requisito de software es una característica que se debe exhibir para solucionar un cierto problema en el mundo real. Se convierte en una combinación compleja de requisitos entregados por parte de los usuarios implicados dentro del desarrollo de la solución , teniendo en cuenta que pueden corresponder a diferentes niveles jerárquicos, ambientes e intereses. Es importante también que cada requisito sea comprobable, pensando también en las implicaciones que esto puede conllevar.*

Teniendo claro que un requisito es una característica que debe tenerse para solucionar cierto problema en el mundo real debemos tener en cuenta que hay distintos tipos de requisitos de los cuales hablaremos brevemente, los tipos de requisitos no son excluyentes entre sí, es decir 1 requerimiento puede ser de más de un tipo.

*Requisito de Producto:*  Se refiere a los parámetros que tenemos que buscar que solucionar por ejemplo el software debe validar si el número de cédula y la contraseña son validos para entrar al sistema.

*Requisito de Proceso:*  Estos tipos de requisitos se refiere más a la parte técnica o las herramientas que usaremos para desarrollar el aplicativo, por ejemplo el aplicativo web deberá ser desarrollado en el motor de base de datos Oracle y usando el lenguaje Web Php.

*Requisitos funcionales:* Hacen referencia a lo que va poder o no hacer el software.

*Requisitos no funcionales:* Este tipo de requerimiento no hace parte fundamental del software pero son los que obligan a que se actúen para llegar a la solución del mismo, por ejemplo botones de confirmación, maquetado web entre otros.

**Solicitud de recursos educativos**

|  |  |
| --- | --- |
| Titulo del recurso educativo | Diferencia entre requerimientos funcional y no funcional |
| Descripción: | Video que relata las diferencias entre los requerimientos funcionales y no funcionales entre otros aspectos. |
| Unidad | 1 |
| Lección | 1 |
| Tipo de recurso: | Video |
| Instrucciones para el estudiante | Tome atenta nota de las diferencias entre los requisitos funcionales y no funcionales con tal de afianzar el conocimiento. |
| URLs de ejemplo o sugerencias | <https://www.youtube.com/watch?v=tF88eNhNSb4>  El video tiene una duración de 3:52 pero del minuto 2:09 al minuto 2:20 no es necesario esa información al igual que en el minuto 2:54 al 3:12 por lo cual debería editarse esta información. |

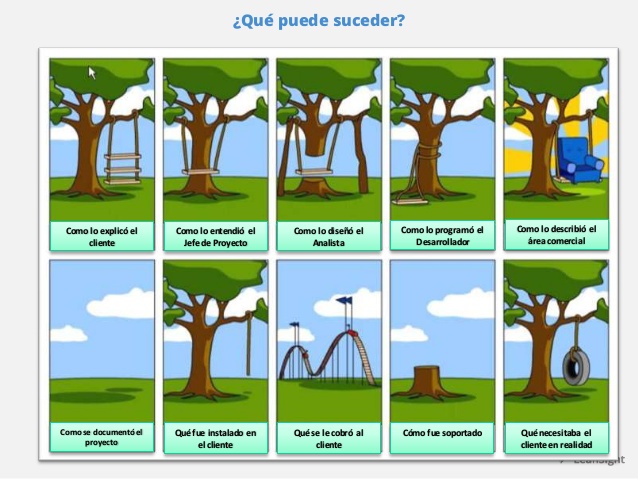
*Características inesperadas:* Este tipo de requerimiento hace referencia aquellos que no pueden ser tratados por un solo componente del software, un ejemplo de este tipo puede ser el software tiene que estar online las 24 horas del día, los 365 días de la semana, para que se logre el ejemplo anterior se tiene que tener en cuenta que se debe tener respaldo de servidor por si se va la luz en donde este esta ubicado, se debe tener doble fluido de internet como de luz, se debe organizar dentro del código que las peticiones se hagan a 2 sitios dependiendo de una condición entre muchas otras cosas.

*Requisitos cuantificables:* Este tipo de requisito se refiere a que en la medida de lo posible todo requisito que se recolecte debe ser medible y se debe poder cuantificar, por ejemplo para un desarrollo web se solicita que el sistema debe ser capaz de manejar 500 solicitudes de compra al mismo tiempo.

*Requisitos del sistema y requisitos del software:*  Los requisitos del sistema hacen referencia a lo mínimo con lo que se puede trabajar el aplicativo, por ejemplo se necesita que se tenga instalado el sistema operativo Windows 7, base de datos Oracle, 4gb de RAM y 500 de disco duro como mínimo. En cambio los requisitos del software hacen referencia al conocimiento que debe tener las personas, las condiciones que debe tener las instalaciones donde funcione y por último los servicios que debe prestar.

**Lección 2:** Pertinencia del levantamiento de requisitos y agentes adecuados.

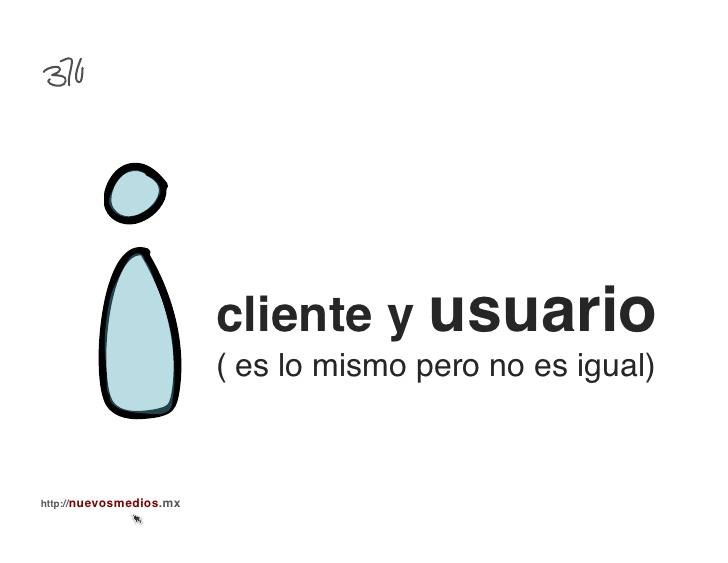
En la lección anterior aprendimos la importancia de los requisitos y los tipos de requisitos que pueden haber pero el verdadero éxito de un desarrollo de software o de un aplicativo web esta en que los requisitos se levanten adecuadamente y que cumplan con las expectativas de todos los agentes que intervienen en el proceso esto con el fin de que no ocurran sorpresas o no cumplan con las expectativas esperadas.



Como se ve en la imagen anterior no entender claramente lo que el cliente solicita puede generar errores, insatisfacción y frustración a los agentes participantes.

Entre los agentes podemos encontrar:

*Cliente:* Es la persona o empresa que nos contrata para desarrollar el software, esto no quiere decir que sea propiamente el que interactúa con el software. Su visión de requerimientos en general es distinta a la de otros agentes, con el cliente podremos obtener requisitos funcionales, requisitos del sistema y del software entre otros.



*Usuario:* Es la persona o grupos de personas que interactúan directamente con el software y al cual le debemos prestar una atención especial ya que nos brindará principalmente información de requisitos no funcionales, de características inesperadas, requisitos cuantificables entre otros.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Diferencia entre usuario y cliente. |
| Descripción: | Imagen que representa un usuario informatico |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Imagen en gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | http://image.slidesharecdn.com/376ladiferenciaentreclienteyusuario-111031210609-phpapp02/95/376-la-diferencia-entre-cliente-y-usuario-1-728.jpg?cb=1320095204 |

*Reguladores:* Generalmente algunas empresas o personas son reguladas por algún ente de control es importante en la medida de lo posible levantar requisitos también a ellos ya que nos permitirán tener una visión más global del desarrollo de software y además asegurarnos que nuestro aplicativo cumplirá con la ley y la normatividad legal vigente.

Luego de tener plenamente identificados a los agentes que participan en el desarrollo del aplicativo web debemos asegurarnos que haya una relación clara y de entendimiento entre las partes, también se debe tener claro que no va ser posible cumplir con todos los requisitos de los distintos agentes y que se deberá negociar ciertos aspectos.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Actividad de reconocer que tipo de requisito es |
| Descripción: | Imagen que resalta la importancia de los requisitos adecuados |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Imagen en gif |
| Instrucciones para el estudiante | El estudiante deberá mirar la imagen para resaltar la importancia de los requisitos ajustados con el agente adecuado |
| URLs de ejemplo o sugerencias | http://image.slidesharecdn.com/leansight-scrum-queesycriteriosdeexito2-140905114538-phpapp02/95/las-sincuenta-sombras-de-scrum-13-638.jpg?cb=1409917972 |

|  |  |
| --- | --- |
| Titulo del recurso educativo | Sopa de letras con conocimientos establecidos |
| Descripción: | Encontrar los conocimientos técnicos mencionados en las lecciones 1 y 2 con el fin de afianzar los conocimientos. |
| Unidad | 1 |
| Lección | 2 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | El estudiante deberá encontrar en una sopa de letras distinas palabras que ayuden afianzar el conocimiento brindado en las unidades 1 y 2. |
| URLs de ejemplo o sugerencias | Palabras a encontrar:  Funcional, requisito, usuario, cliente, reguladores, nofuncionales, sistema, software, lenguaje. |

**Lección 3:** Captura de requisitos

La captura de requisitos se refiere de donde vienen los requisitos y como se realizará la obtención de los mismos, en este proceso el desarrollador por primera vez podrá tener una comprensión adecuada del problema que se requiere solucionar, en esta etapa es importante la participación de todos los agentes mencionados en la lección anterior. Es la etapa más temprana en el ciclo de vida del desarrollo del aplicativo web

Para capturar los requisitos el primer paso que se debe realizar es identificar sus fuentes y evaluar el impacto que tendrán en el mismo.

Entre las principales fuentes pero no las únicas tenemos:

*Metas:* Son los objetivos que tienen que cumplir el aplicativo web que desarrollaremos y nos permitirán ir conociendo si vamos en el camino adecuado para cumplir con la entrega del software, también se debe manejar la capacidad para saber que metas son realizables y cuales no y dejarlas claras desde un inicio con el cliente para no generar malestares posteriores.

*Conocimiento del dominio:* Tener conocimiento en el área en el cual se va desarrollar una solución es de suma importancia para tratar de entender lo que para el agente es difícil de comunicar en términos entendibles. Por ejemplo si vamos a desarrollar un aplicativo web contable y nos hablan de PYG, superávit, cuenta de orden entre otros muchos conocimientos propios de contabilidad no estaremos en la capacidad de desarrollar la solución ya que no contamos con los conocimientos básicos del mismo.

*Stakeholders ( agentes):* Como se menciono en la lección anterior debemos estar en la capacidad de generar un punto de encuentro entre todos los requisitos de los distintos agentes para no generar insatisfacción en los mismos.

Ya sabemos que es un requisito, como categorizarlos, de donde obtenerlos, ponerlos como objetivos; ahora debemos tener técnicas para capturarlos.

Entre las distintas técnicas existentes tenemos:

*Entrevistas:* Se debe realizar con los distintos agentes, con un objetivo claro en cada entrevista, escuchar las necesidades, lo que espera y lo que quiere del software y en la medida de lo posible grabarlas y levantar actas posterior al fin de la entrevista.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Entrevistas |
| Descripción: | Por medio de las entrevistas se baja la tensión y el usuario se relaja fluyendo un poco más el levantamiento de requisitos. |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Imagen en gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | *http://contalento.es/wp-content/uploads/2014/11/entrevista-trabajo.jpg* |

*Reuniones:* Se realizan con grupos del mismo tipo de agente, o con grupos de distintos agentes esto con el fin de lograr refinar las ideas, requisitos de forma tal que sea un punto neutro para todos los participantes en el desarrollo web.

*Observación:* Simplemente observar el proceso que hace normalmente el agente nos dará una visión más particular y real de la situación que hace la persona, esto con el fin de lograr entender algunos procesos complejos que hace implícitamente algún agente y no es capaz de transmitirlo como tal, por ejemplo supongamos un cocinero que hace una torta, el nos da los pasos para hacer la torta pero resulta que cuando mete al horno la torta hace una mezcla adicional o aplica alguna técnica en especial y que no la transmitió pero que por medio de la observación la pudimos entender.

*Prototipos:* Es una técnica usada para capturar requisitos cuando los agentes no son claros con la información que nos suministran, esto nos permite darle un contexto al usuario para que se ubique y le sea más familiar el ambiente y nos ayude a conceptualizar mejor su información, por ejemplo una ventana de inicio donde el usuario vaya construyendo como le gustaría que el aplicativo web interactuará con nosotros.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Captura de requisitos. |
| Descripción: | Para conocer los conocimientos de la captura de requisitos el estudiante deberá dar clic en el menú interno donde encontrará la definición de cada término al interior de un recuadro. |
| Unidad | 1 |
| Lección | 3 |
| Tipo de recurso: | Recurso HTML |
| Instrucciones para el estudiante | De clic para ahondar en las fuentes de conocimientos que puede tener en la captura de requisitos. |
| URLs de ejemplo o sugerencias | http://contenidos.sucerman.com/nivel3/redes/unidad1/leccion4.html  El menú debe ser como se utiliza en los ejemplos de topologias físicas en redes WAN con el termino inicial como fuente y el final como prototipo. |

**Lección 4: Análisis y Validación de los requisitos.**

Al realizar el levantamiento de requisitos nos vamos a encontrar con una gran cantidad de requerimientos que tendremos en la medida de lo posible unificarlos, descartarlos y realizarlos, pero como saber cual es cual. A través del análisis de requisitos podremos realizar este proceso.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Análisis |
| Descripción: | Un adecuado análisis de los requerimientos nos evitará dolores de cabeza en el futuro. |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Imagen en gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | http://revistaitnow.com/wp-content/uploads/2012/07/an%C3%A1lisis1.jpg |

El primer paso es clasificarlos tal cual como lo vimos en la lección 1 de esta unidad, por medio de esta clasificación podemos comenzar a ver cuales son los realmente importantes y cuales de pronto son de forma solamente, podremos ver las características que vamos a tener en el sistema. De pronto podremos pensar pues todos los requisitos que sean del tipo no funcional los puedo reducir pero esto no sería lo más adecuado supongamos que un agente quiere que el software sea de color azul con morado y sin eso no dará la aprobación del mismo, por eso es muy importante analizar y negociar que irá y que no.

El segundo paso es realizar una nueva clasificación a través de la prioridad, ver cual de todos los requisitos tiene una prioridad mayor por ejemplo para un cliente que tiene un negocio de comidas rápidas puede decir que para el lo más importante es el módulo de facturación pero otro cliente del mismo tipo de negocio de comidas rápidas dice que para el lo más importante es el despacho de pedidos. Por eso internamente en el desarrollo del aplicativo Web debemos establecer prioridades para los requisitos y establecernos en un orden lógico que nos permita cumplir con los objetivos trazados.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Ciclo de vida en general del levantamiento de requisitos. |
| Descripción: | A través del ciclo de vida del levantamiento de requisitos nos podemos asegurar un desarrollo web exitoso. |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Imagen en gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | <http://image.slidesharecdn.com/diapositivasexpo2-110629094337-phpapp01/95/ingenieria-de-requerimientos-13-728.jpg?cb=1309340716> |

Como hemos mencionado en varias oportunidades también debemos poner un alcance a los requisitos, en la medida de lo posible todo requisito debe tener una unidad de medida con el fin de demostrarnos que ya conseguimos cumplir con este requisito.

También dentro de los requisitos se debe prever el crecimiento y la estabilidad del software, por ejemplo supongamos que diseñamos un software que permite manejar todo un negocio de comidas rápidas para un establecimiento pero el cliente en algún momento nos indico que tiene previsto abrir 2 establecimientos más en un futuro cercano, por lo cual dentro de nuestra construcción debemos ver como incluiremos estos 2 establecimientos en el futuro y como podrán tener toda la información unificada en algún momento.

Luego que hemos clasificado por tipo y por nivel de prioridad los requisitos se define una reunión con el cliente y los usuarios para enseñar los requisitos aprobados en el desarrollo del software donde se busca la aprobación por parte del cliente, esto se conoce como la validación de los requisitos. Es importante que al finalizar esta etapa estén todos los agentes que participan en el desarrollo del proyecto y se genere un acta de aprobación por parte de los agentes (stakeholders) con el fin de evitar complicaciones en un futuro.

Plantilla de requisitos

|  |  |
| --- | --- |
| **Nombre** | Nombre del requerimiento |
| **Autor** | Nombre de la persona que levanta el requerimiento |
| **Fecha** | Dd/mm/aaaa |
| **Descripción** | |
| Breve descripción del requerimiento. | |
| **Actores** | |
| Agentes que participan en el requerimiento. | |
| **Precondiciones** | |
| Situaciones que se deben tener en cuenta antes del requerimiento. | |
| **Flujo normal** | |
| Lo que se espera que haga normalmente el requerimiento. | |
| **Flujo alternativo** | |
| Como actúa el requerimiento en alguna situación especial. | |
| **Nivel de importancia.** | |
| **Alto ( ) Medio ( ) Bajo ( )** | |
| **Postcondiciones** | |
| Donde conecta el requerimiento o si con el requerimiento acaba el proceso. | |

|  |  |
| --- | --- |
| Titulo del recurso educativo | Captura de requisitos. |
| Descripción: | Plantilla para llenar los requisitos de tal manera que permita su análisis de forma adecuado y posterior validación. |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Recurso HTML |
| Instrucciones para el estudiante | El estudiante deberá realizar el llenado del formulario de forma adecuado para cada uno de los requisitos que surgan dentro de la etapa de levantamiento de requisitos. En este espacio se usará solo como espacio netamente de ejemeplo. |
| URLs de ejemplo o sugerencias | Debe ser un formulario que permita el manejo del documento a través de la interacción del mismo. Basandose en la plantilla de requisitos ubicada en la página 10 del presente documento. |

|  |  |
| --- | --- |
| Titulo del recurso educativo | Crucigrama del contenido de la lección. |
| Descripción: | El estudiante deberá completar el crucigrama a partir de los conocimientos generados en toda la unidad, reforzando asi los conocimientos adquiridos. |
| Unidad | 1 |
| Lección | 4 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | El estudiante deberá encontrar en una sopa de letras distinas palabras que ayuden afianzar el conocimiento brindado en las unidades 1 y 2. |
| URLs de ejemplo o sugerencias | Palabras del crucigrama con su relación.  Horizontales:  **Validación:** Se refiere al proceso que se realiza posterior a la captura de los requisitos y el análisis de los mismos.  **Stakeholders:** Hace referencia a todas las entidades, usuarios o clientes que participan en el levantamiento de requisitos.  **Entrevistas:** Proceso por el cual se escucha de una forma directa y sin interrupciones a los usuarios del sistema, donde se toma atenta nota de cada uno de los requisitos que pide cada usuario.  **Usuario:** Persona, grupo de persona o sistemas que interactuan de forma directa con el desarrollo web.  Verticales:  **Requisito:** característica que se debe exhibir para solucionar un cierto problema en el mundo real  **Cliente:** Persona que nos contrata para desarrollar una solución web.  **Clasificación:** Proceso por el cual categorizamos los requerimientos obtenidos en la captura de requerimientos.  **Prioridad:** Proceso por el cual generamos un nivel de importancia entre requerimientos. |

**UNIDAD 2: DISEÑO DE SOFTWARE**

**Lección 1: Fundamentos**

El diseño de software se define como el proceso donde en el ciclo de vida del desarrollo web se definen los componentes, las interfaces, la arquitectura y otros aspectos necesarios para completar los requerimientos realizados por el cliente en el ciclo de vida de levantamiento de requerimientos. En esta etapa del ciclo de vida se realiza un bosquejo muy aterrizado de todo lo que necesita para alcanzar los requerimientos y como serán las interacciones de los Stakeholders con el software a través de distintos diagramas tales como los diagramas de casos de uso, los diagramas de secuencia y los diagramas de distribución, entre muchos incluidos en los Diagramas UML (Unified Modeling Language), los diagramas UML es un lenguaje creado precisamente para estandarizar la representación gráfica de las distintas formas de plasmar las interacciones de un desarrollo de software.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Diagramas UML |
| Descripción: | Distintos diagramas de UML |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Recurso de Interacción HTML |
| Instrucciones para el estudiante | Revisar con cuidado la cantidad de recursos que se tiene para poder representar las interacciones con el sistema. |
| URLs de ejemplo o sugerencias | http://es.oocities.com/monsalvelaura/fase2/images/diagramasuml.jpg  Si hay la posibilidad de que al pasar el cursor sobre por ejemplo diagrama de casos de uso, diagrama de secuencia y diagrama de distribución se muevan o resalten con un borde un poco más grueso para mostrar que esos son los 3 diagramas de los que se verán en la unidad. |

Un diagrama de caso de uso no es más que una descripción gráfica de un proceso desde su inicio a su fin, tal cual como cuando programamos utilizábamos los diagramas de flujo para describir lo que haría nuestro algoritmo, los diagramas de caso de uso nos permiten describir la forma como los usuarios a través de los requerimientos van a realizar acciones en el software.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Definiciones de Casos de uso |
| Descripción: | Resumén de las distintas definiciones de casos de uso. |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Leer con atento cuidado las definiciones para obtener el conocimiento sobre el diagrama de caso de uso. |
| URLs de ejemplo o sugerencias | http://www.codecompiling.net/files/slides/UML\_clase\_02\_UML\_casos\_de\_uso.pdf  Convertir la página 3 del sitio en mención en imagen. |

Dentro de las partes de acciones de caso de uso tenemos.

**Actor ó actores:** Hace referencia al usuario o usuarios que interactúan con el diagrama de caso de uso. Generalmente se representa con una persona en caso de ser un usuario o en su defecto con el sistema informático que relacione. Por ejemplo un diagrama de caso de uso puede representar la interacción que debe hacer una base de datos para consultar si tienen un dato en su información y esta a su vez es solicitada por un usuario.

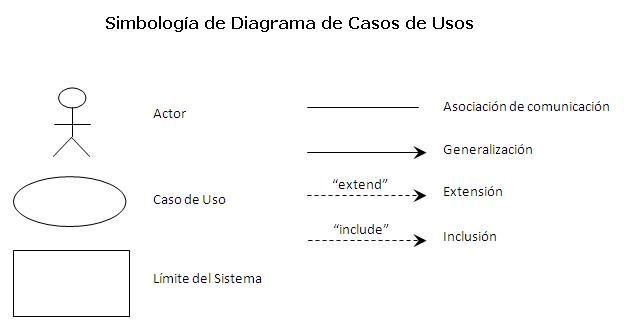
**Caso de uso:** Hace referencia a la acción o a los juegos de acciones que se realizan dentro del requerimiento, supongamos el ejemplo anterior, cuando entramos a Facebook nosotros interactuamos ingresando nuestro usuario y contraseña ( caso de uso de logeo), luego de esto el sistema debe ir a otro caso de uso que denominaremos verificación, el cual realizará la verificación del caso de uso logeo a través de la base de datos que sería otro actor dentro del sistema. Por último el sistema nos daría una respuesta ya sea negativa o positiva que sería el último paso y llamaremos a este caso de uso como inicio.

**Limite de Sistema:** Muestra el limite del sistema que estamos representando, recordemos que en un requerimiento pueden interactuar distintos agentes y distintos casos de uso.

**Línea de asociación:** Representa la relación entre un actor y un caso de uso, la relación entre actores y casos de uso puede ser de 1 a 1, de 1 a muchos y de muchos a 1.

**Extensión:** Hace referencia cuando un caso de uso se extiende a otro caso de uso, en el caso del ejemplo que usábamos en caso de uso haría referencia al extends entre verificación y validación. También conocido como extends.

**Incluye:** Este proceso hace referencia cuando un caso de uso incluye todos los atributos de otro caso de uso. Por ejemplo cuando nos suscribimos a una revista deportiva, hay un caso de uso que se llamará inscripción revista y este caso de uso incluirá un caso de uso que se llaman revistas deportivas.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Partes de un diagrama de caso de uso. |
| Descripción: | Se distinguen los distintos componentes de un diagrama de caso de uso. |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Identificar de forma gráfica como se representa cada uno de los componentes de un diagrama de caso de uso. |
| URLs de ejemplo o sugerencias | http://es.oocities.com/monsalvelaura/fase2/images/simbologiacasosusos.JPG  Convertir la página 3 del sitio en mención en imagen. |

Un diagrama de secuencia permite mostrar la forma en que los componentes u objetos del software se comunican entre sí al transcurrir el tiempo.

En un diagrama de secuencia podemos encontrar, los objetos que participan en la interacción y la secuencia de mensajes que se intercambian entre los objetos.

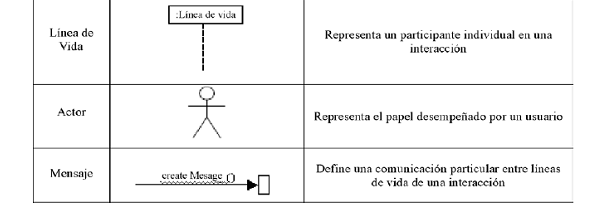
Dentro de las partes de los diagramas de secuencia tenemos:

**Objetos***:* Pueden ser stakeholders o componentes propios del sistema, son los que realizan como tal las peticiones y esperan respuestas de otros objetos.

**Mensajes:** Son la información que se transportan entre objetos y se representan de forma horizontal, pueden ser de 3 tipos, el mensaje simple, que ocurre cuando se hace la transferencia de control de un objeto a otro, los mensajes síncronos que se utilizan cuando un objeto solicita a otro una información y queda a espera de su respuesta y por último tenemos los mensajes asíncronos, este tipo de mensaje a diferencia del síncrono solicita una respuesta pero mientras llega la respuesta del destinatario va realizando otros procesos.

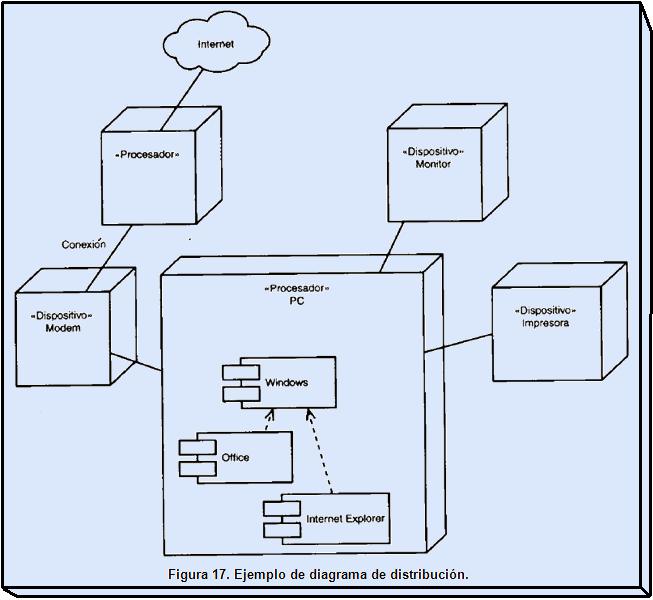
**Línea de tiempo**: Se representa de forma vertical e indica el tiempo transcurrido desde que se inicio el objeto, un mensaje que se ubique en la parte superior de la línea de tiempo indica que va antes de un mensaje que se encuentre en la parte inferior de la línea de tiempo.

Los diagramas de Secuencias se usan continuamente como modelos explicativos para los escenarios de los casos de uso. Mediante la creación de un diagrama de secuencias con un actor y los elementos involucrados en el caso de uso, puede modelar la secuencia de pasos que toman el usuario y el sistema para completar las tareas requeridas.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Partes de un diagrama de secuencia |
| Descripción: | Se distinguen los distintos componentes de un diagrama de secuencia |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Identificar de forma gráfica como se representa cada uno de los componentes de un diagrama de secuencia |
| URLs de ejemplo o sugerencias | <http://www.scielo.org.co/img/revistas/eia/n10/n10a08t1.gif>  En la medida de lo posible recortar la imagen para mostrar solo los 3 componentes que estan en la imagen inmediatamente superior. |

El Diagrama de distribución representa la estructura de hardware necesaria para llevar a cabo el desarrollo web. Este diagrama es uno de los más fáciles de manejar ya que solo tiene un componente para representarse y es a través de un Nodo, un nodo es un cubo que representa cualquier estructura de hardware que participe en nuestro sistema y ejecuta o procesa alguna acción, dentro de los nodos se pueden incluir otros diagramas UML, representándolos completos o solo haciendo relación al nombre con el que lo creamos, también hay nodos que participan en nuestro sistema pero no ejecutan ni procesan, tales como impresoras o pantallas de computador.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Ejemplo de un diagrama de distribución |
| Descripción: | Se distingue la representación de los nodos y la inclusión de otros diagramas dentro de los nodos. |
| Unidad | 2 |
| Lección | 1 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Identificar de forma gráfica como interactuan los distintos diagramas dentro de un nodo en el diagrama de distribución |
| URLs de ejemplo o sugerencias | <http://www.monografias.com/trabajos67/diagramas-uml/image017.png> |

**Lección 2: Diseño de datos.**

El activo más importante de una organización es la información por lo tanto en esta lección aprenderemos como realizar un diseño adecuado para que los datos no sean solo datos sino que se conviertan en información que pueda ser consultado por todos y usada de forma adecuada.

El diseño de datos consiste en descubrir y definir completamente los proceso y características de los datos de la aplicación. Es un proceso de perfeccionamiento gradual que abarca las situaciones más simples como por ejemplo ¿Qué datos requieren la aplicación? ¿ Cómo se accederán a esos datos? ¿Cómo se almacenarán los datos?. Si se logra un diseño de datos bueno el acceso a los datos de la aplicación será rápido y fácil de mantener y podrá aceptar sin problemas las futuras mejoras de los datos.

Un ejemplo de diseño de datos adecuado ocurrió con el sitio web Youtube.com y el mega éxito “Gangnam Style” ya que el contador de visitas al video que maneja YouTube estaba destinado para manejar una capacidad máxima de 32 bit es decir 2.147.483.647 una cifra que tiempo atrás se creía inalcanzable pero el Surcoreano hizo cambiar esta situación y los desarrolladores de YouTube gracias a un buen diseño de datos cambiaron el tipo de dato de 32 bits a 64 bits alcanzando una cifra de 4.294.967.296

|  |  |
| --- | --- |
| Titulo del recurso educativo | Éxito Gangnam Style |
| Descripción: | Permite ver el diseño de datos en el contador de “views” de Youtube, |
| Unidad | 2 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación - Video |
| Instrucciones para el estudiante | Identificar de forma gráfica como se representa cada uno de los componentes de un diagrama de caso de uso. |
| URLs de ejemplo o sugerencias | <https://www.youtube.com/watch?v=9bZkp7q19f0>  Es importante que se vea el contador de visitas del video para ver como aumenta cada vez que se recarga la página. |

El diseño de datos se centra más que todo en el diseño de la estructura de la base de datos y los archivos que van a ser usaos por el sistema de información en construcción.

Aunque en el diseñado de datos trataremos de cubrir todos los aspectos fundamentales y necesarios para el diseño adecuado para almacenar, mantener y recuperar datos es imposible precisar al inicio de un diseño como, cuando y donde se van almacenar los datos, generalmente solo se puede prever el uso de un motor de base de datos relacional.

Para un diseño de datos adecuado se debe incluir la identificación de datos, la definición de datos, la integridad de datos, precauciones que se deben adoptar al diseñar datos.

**Identificación de datos:** Los datos nos permiten describir cosas, personas, productos, elementos, clientes entre otras muchas cosas más que nos permiten realizar tareas de clasificación por categorías, organización y mantenimiento.

La identificación de datos es un proceso iterativo que a medida que va iterando va siendo más preciso y de alto nivel, esto permite adquirir más conocimiento sobre los procesos previstos de la aplicación y hacerlos más fiables. Cuando identificamos un dato generalmente se debe documentar la siguiente información: nombre, descripción general (qué es), propiedad (quién es el responsable), características (como se mide, que magnitud puede tener), relaciones, procesos y eventos lógicos ( cómo y cuándo se crea, modifica y utiliza).

Es importante resaltar los datos tienen muchas características diferentes y es importante cuantificar cada uno de los datos con atributos medibles. Algunas características típicas de los son las siguientes:

* Atributos de ubicación (dirección, país, lugar de almacenamiento).
* Atributos físicos (peso, dimensiones, volumen, color, material, textura).
* Atributos conceptuales (nombre, categoría, número de serie).
* Atributos relacionales (conjuntos formados por subconjuntos, escritores de varios libros).
* Atributos de valor (moneda, disposición, consideración).

Es importante resaltar que no todos los datos deben ser incluidos en el desarrollo web y esta en nuestra habilidad como desarrolladores saber que vamos a documentar y que características van a tener estos datos.

**Definición de datos:** A medida que vamos identificando los datos y su estructura general a través de las características de los datos podemos crear relaciones entre ellos y diseñar un poco más nuestra base de datos aplicando lo que hemos visto en cursos anteriores. Para definir los datos se debe realizar las siguientes acciones:

* Definir tablas, filas y columnas

Independientemente de la forma en que se almacenan los datos en la aplicación estos generalmente deben estar establecidos en tablas donde por medio de conjuntos de atributos identificaremos al objeto o dato que hacemos relación.

Por ejemplo un dato persona, contendrá los campos Nombre, Dirección y Teléfono y un id numérico.

Tabla Personas

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Nombre** | **Dirección** | **Teléfono** |
| 108825212 | Andrés Del Valle | xxxxx xxxxxxxx | (xxx) xxx-xxxx |
| 112313566 | Carlos García | xxxxx xxxxxxxx | (xxx) xxx-xxxx |
| 212425667 | Isabel Díaz | xxxxx xxxxxxxx | (xxx) xxx-xxxx |
| 4464574577 | Bernardo Ramírez | xxxxx xxxxxxxx | (xxx) xxx-xxxx |

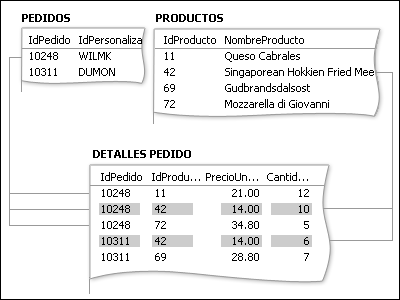
* Insertar claves de índice (llaves primarias)

Para el ejemplo anterior la llave primaria o clave de indice sería el atributo o campo Id el cual nos permite crear una relación con el objeto único pesona, ya sea bernardo ramírez o isabel díaz.

Una clave es un campo especial que proporciona un índice para la recuperación de datos de forma rápida.

* Insertar relaciones entre tablas.

Atributos de un mismo dato se relaciona en una tabla y a su vez atributos de otras tablas se pueden relacionar con un atributo que sea único de una tabla por ejemplo para la tabla personas la clave de indice o llave primaría es Id, esto nos permitirá relacionar por ejemplo con otra tabla que fuera notas de la persona. Tal cual como vimos en el curso de base de datos.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Relaciones entre tablas |
| Descripción: | Permite ver una relación entre 2 tablas |
| Unidad | 2 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación - Imagen |
| Instrucciones para el estudiante | Recordar el modelo entidad – relación como la forma más adeucada para optimizar el diseño de datos. |
| URLs de ejemplo o sugerencias | <http://techinerd.com/wp-content/uploads/2014/05/difference.png> |

https://osiprodeusodcspstoa01.blob.core.windows.net/es-es/media/2dabb02b-c7cb-452f-99f7-ddc6accdb05b.gif

* Asignar tipos de datos:

Por último nos queda solo asignar de forma adecuada os tipos de datos para cada atributo del dato, por ejemplo el atributo nombre de la tabla persona debe ser de tipo texto ya que nadie se llama 1418523578. Estos tipos de datos generalmente van muy relacionado al tipo de base de datos que usemos y pueden cambiar sus caracteristicas de un motor de base de datos a otro. Lo importante a resaltar en esta etapa de diseño es recordar que el espacio de almacenamiento es un recurso finito y que asignar el tipo de dato adecuada en la fase de diseño nos permitirá optimizar y mantener de una forma más adecuada nuestro software.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Tipos de motores de bases de datos |
| Descripción: | Reconocer la importancia de los tipos de datos independiente del motor de base de datos que se utilice |
| Unidad | 2 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación - Imagen |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | <http://techinerd.com/wp-content/uploads/2014/05/difference.png> |

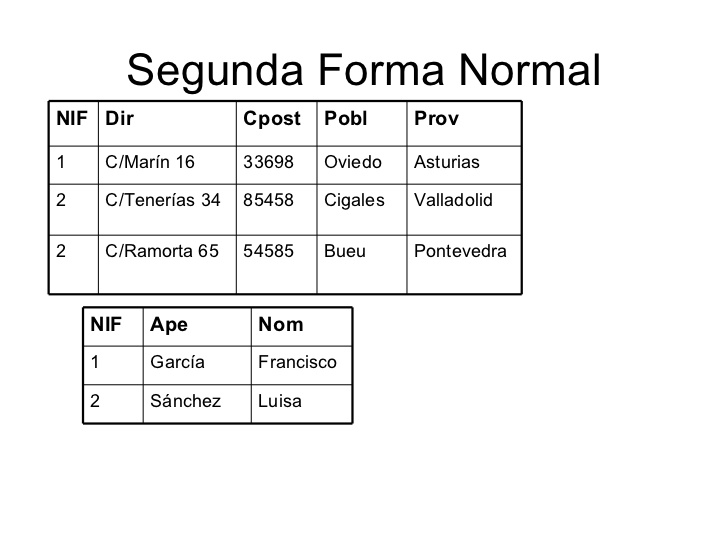
**Integridad de datos:** Con la integridad de datos nos referimos a los valores reales que almacenamos y utilizamos en la base de datos del desarrollo web, debemos garantizar que el desarrollo web solo acepte, almacene y mantenga los datos adecuados generando control sobre los mismos y generando los mecanismos para la corrección de los mismos, hay distintas formas de garantizar la integridad de los datos como lo son:

* Normalizar datos.

Explica el proceso que consiste en perfeccionar las definiciones de datos para eliminar grupos de dependencias innecesarias.

Este proceso consiste en tratar de minimizar las dependencias entre claves y que exista una mayor cohesion de los datos supongamos que tenemos un dato en el cual existen más de una llave única, la normalización nos enseña que lo ideal es separar estas en datos más pequeños y minimizar riesgos.

Hay 3 niveles de normalizar los datos y es un proceso que deben recordar de las bases de datos



|  |  |
| --- | --- |
| Titulo del recurso educativo | Tipos de motores de bases de datos |
| Descripción: | La normalización permite que una base de datos este más estandarizada y baja la complejidad de los datos al realizarse busquedeas muy grandes. |
| Unidad | 2 |
| Lección | 2 |
| Tipo de recurso: | Recurso de observación - Imagen |
| Instrucciones para el estudiante | Observe con cuidado el ejemplo de normalización de una tabla. |
| URLs de ejemplo o sugerencias | <http://image.slidesharecdn.com/formasnormales-110503005120-phpapp02/95/formas-normales-10-728.jpg?cb=1304383914> |

* Definir reglas de empresa.

Por medio de las reglas de la empresa establecemos el control coherente y correcto para el acceso a datos de la aplicación esto incluye la inserción, actualización, eliminación y vista de los dtos, valdiación de datos, controlar la seguridad de los datos, controlar el acceso a datos de varios archivos.

* Proporcionar integridad referencial.

Es el mecánismo por el cual garantizamos que se dañen los datos.

* Validar los datos.

Son los proceso por los cuales comprobamos que los datos almacenados sean validos, este proceso puede ser antes, durante y después de que el dato este en nuestra base de datos, por ejemplo cuando la persona este llenando los apellidos, podemos crear un filtro en el código de nuestro sistema para asegurarnos que solo se suban caracteres de letras y no números. Que en el campo teléfono solo se permitan valores numéricos, ya cuando los datos esten almacenados en nuestra base de datos podemos convertir los datos de un tipo de dato a otro y por último si deeamos eliminar los datos y no dejar cobertura debemos asegurarnos que estos datos no puedan ser legibles.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Enunciados de verdadero y falso. |
| Descripción: | A partir de los conocimientos de la lección 1 y 2 contestar verdadero o falso los siguientes enunciados |
| Unidad | 2 |
| Lección | 2 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | Por medio de los conocimientos adquiridos en la lección 1 y 2 por favor conteste verdadero o falso según sea el caso los siguientes enunciados. |
| URLs de ejemplo o sugerencias | 1. La normalización de una tabla de datos busca aumentar la complejidad de la tabla de datos para disminuir el rendimiento de la base de datos (Respuesta FALSO).  2. Con la integridad de datos buscamos garantizar que el desarrollo web solo acepte, almacene y mantenga los datos adecuados generando control sobre los mismos y generando los mecanismos para la corrección (Respuesta VERDADERO).  3. Los mensajes es la información que se transportan entre objetos y se representan de forma horizontal, pueden ser de 3 tipos, el mensaje simple, que ocurre cuando se hace la transferencia de control de un objeto a otro, los mensajes síncronos que se utilizan cuando un objeto solicita a otro una información y queda a espera de su respuesta y por último tenemos los mensajes asíncronos (Respuesta VERDADERO).  4. El diagrama de componentes esta compuesto por actores, casos de uso, limite del sistema, incluides y extendes (Respuesta FALSO).  5. Un total de 20 tipos de diagramas son los existentes en el lenguaje de modelado unificado ó sus siglas en ingles UML (Respuesta FALSO).  6. La identificación de los datos nos permiten describir cosas, personas, productos, elementos, clientes entre otras muchas cosas más que nos permiten realizar tareas de clasificación por categorías, organización y mantenimiento. (Respuesta VERDADERO).  7. Un stakeholders es una parte fisica de hardware que permite optimizar el rendimiento de nuestro aplicativo web (Respuesta FALSO).  8. El Diagrama de caso de uso representa la estructura de hardware necesaria para llevar a cabo el desarrollo web. (Respuesta FALSO).  9. Un diagrama de caso de uso no es más que una descripción gráfica de un proceso desde su inicio a su fin (Respuesta Verdadero).  10. Entre los activos más importante de una empresa esta el activo de la información (Resputa VERDADERO). |

**Lección 3: Estructura y Arquitecura de software**

En este momento estamos en la mitad de la segunda etapa del ciclo de vida de un desarrollo web, hemos aprendido sobre que es un requerimiento, como levantarlos, como realizar su aprobación; posterior a esto aprendimos como plasmar estos requerimientos en diagramas gráficos para entender su complejidad y luego de esto aprendimos que los requerimientos van acompañados por datos los cuales establecemos en una base de datos y permiten consolidar información. Pero ahora aprenderemos sobre la arquitectura de software y su estructura, haciendo una analogía con la construcción de un edificio en este momento solo hemos comprado el lote donde construiremos el edificio, sabemos de cuantos pisos va ser el mismo, para que va ser usado, hemos comprado algunos que otros materiales y contratado algunos obreros, pero ahora tenemos que cimentar nuestro edificio y para esto tenemos que realizar calculos, análisis de que tipo de cemento vamos a usar, cuantas vigas necesitamos y muchas cosas más que solo una buena arquitectura nos garantiza que al culminar nuestro edificio este no se vaya a caer.

Lo mismo ocurre con el software, no importa que tan bien hagamos el levantamiento de requisitos y el diseño de los datos sino escogemos la arquitectura adecuada el software se convertirá en un efecto bola de nieve que a la final lo más fácil será volver a construirlo todo de nuevo.

Por lo anterior la arquitectura de software se define como la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán y los principios que orientan su diseño y evolución.

**Componentes:** Esta compuesta por clientes y servidores, bases de datos y nivelen en sistemas jerárquico.

**Interacciones:** Existens conjuntos de interacciones entre los que encontramos lladas a procedimientos, comportamientos variables y transmisión asincrona de eventos.

La arquitectura de software tiene distintos patrones de diseño arquitectónicos los cuales provee un esquema para refinar los elementos de un sistema de software o las relaciones entre ellos, existentes distintos tipos de patrones de diseño pero en el presente curso solo trataremos el modelo cliente servidor y el modelo de capas

**Modelo Cliente Servidor:** Un servidor es una aplicación que ofrece un servicio a usuarios de una red, el servidor puede estar instalada en la misma máquina o puede estar instalada en otra máquina. Es decir un cliente puede ser al mismo tiempo un servidor si el programa esta instalado en la misma máquina. El funcionamiento del modelo cliente/ servidor es sencillo el servidor recibe solicitudes por parte de 1 cliente o distintos clientes, a partir de estas solicitudes, el servidor realiza lo que le solicitan y le devuelve un resultado al cliente.

Por favor recurrir al video para entender el modelo.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Modelo cliente / servidor |
| Descripción: | El modelo cliente servidor es uno de los servicios más usados en la actualidad. |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación - Video |
| Instrucciones para el estudiante | Observe cuidadosamente el video para entender un poco más el modelo cliente servidor. |
| URLs de ejemplo o sugerencias | [**https://www.youtube.com/watch?v=ea2vy1mQUmA**](https://www.youtube.com/watch?v=ea2vy1mQUmA)  **Mostrar el video solo hasta el minuto 1,15.** |

El cliente es el entte que se encarga en el modelo cliente / servidor de generar las solicitudes, demandar peticiones y obtener información que le envia el servidor. Hay 3 tipos de clientes, cliente pesado, cliente liviano y cliente hibrido.

**Cliente liviano:** Un cliente liviano hace referencia a un dispositivo de hardware o software que no usa sus recursos propios para ejecutar funciones sino que realiza una conexión a otro dispositivo de hardware o software para que ejecute y realice las peticiones. Generalmente un cliente liviano no requiere tener demasiadas características de hardware ya que no tiene que ejecutar demasiados procesos ya que el desgaste lo realiza el servidor.

**Cliente Pesado:** El cliente pesado es todo lo contrario a un cliente liviano ya que las peticiones, solicitudes y requerimientos que genera el cliente los realiza el mismo y solo se comunica con el servidor para actualizar la información, insertar datos, eliminar datos entre otras cosas.

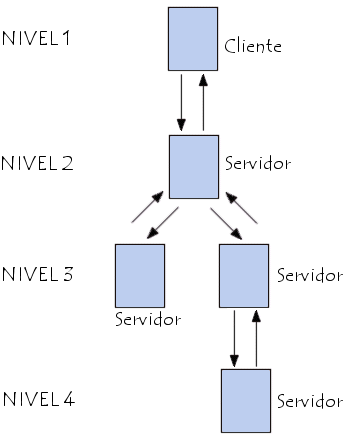
**Cliente Hibrido:** Este tipo de cliente hace referencia a estaciones de trabajo que no tienen disco y por lo tanto no almacenan información, para poder realizar su función se basa en una conexión por red para realizar las funciones.

Los 3 tipos de clientes tienen sus ventajas y sus desventajas pero esta en nuestra pericia como desarrolladores web tener la capacidad de que tipo de cliente usar y en que momento.

**Modelo de n Capas:** El modelo de capas propone una forma distinta de trabajar el desarrollo web, donde se busca es separar la lógica de negocios de la lógica de diseño, un ejemplo claro puede ser buscar separar la capa de datos de la capa de presnetación del usuario.

Lo que conocemos como capas no es más que un estilo de programación donde utilizamos la máxima “divida y vencerás” y donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento, etc.

Esto en resumida permitirá una mejor usabilidad del código del desarrollo web, un mantenimiento más fácil de administrar y un mayor rendimiento del servidor.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Modelo N Capas. |
| Descripción: | En un modelo N capas cada nivel hace referencia a una parte del entorno del desarrollo web. |
| Unidad | 2 |
| Lección | 3 |
| Tipo de recurso: | Recurso de observación - Imagen |
| Instrucciones para el estudiante | Observe con cuidado el ejemplo de Modelo n Capas |
| URLs de ejemplo o sugerencias | <http://3.bp.blogspot.com/-7woRty8tU18/T7_59Xjc5fI/AAAAAAAAAK8/2a3NFXTjBPY/s1600/n-tier.gif> |

**Lección 4: Diseño de interfaz.**

El diseño de interfaz de usuario es la construcción del contenido con el cual el usuario final va interactuar con el sistema. Esto incluye el software, el hardware y todo lo necesario para ser lo más amigable el entorno con el que el usuario se va encontrar.

Para entender un poco más este ejemplo, supongamos que vamos a comprar un vehículo y tenemos 2 opciones, un carro color rosado, con rines de lujo, último modelo y la otra opción es una camioneta 4x4 color negro, con rines de lujo. Dado lo anterior algunos preferimos la camioneta y otros preferiremos el carro color rosado porque sentimos un poco más de afinidad con el uno y con el otro. Lo mismo sucede con una interfaz, si es amigable, si luce bonita y además si es simple su manejo a un usuario le va gustar mucho más que otra interfaz.

Por esto el diseño de interfaz se convierte en un paso que no podemos dejar de lado, y en la medida de lo posible tenemos que realizar bosquejos que le permita al usuario y a nosotros mismo ver como se va ver el software ya que a la final el hardware y software se convierten en simples herramientas.

El entorno dentro del cual se inscribe el diseño de una interfaz y su usabilidad edado por 3 factores, persona, tarea y contexto.

Para lograr el objetivo principal del diseño de ser más atractivas se deben tener los siguientes principios en cuenta:

* **Familiaridad del usuario:** Utilizar lenguaje y terminos reconocidos por el usuario.
* **Consistencia:** Todos los procesos similares deben cumplir el mismo proceso.
* **Minima sorpresa:** No debe haber comportamientos distintos al que espera el usuario en el uso del mismo.
* **Recuperabilidad:** Se debe proveer de mecánismos para proteger al usuario de errores o deshacer acciones.
* **Guia al usuario:** Debe existir un manual de usuario que permita al usuario retroalimentarse y ayudar en ciertos conextos.
* **Diversidad de usuarios:** Se debe proveer características de interacción apropiada para los diferntes tipos de usuarios.

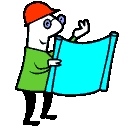


|  |  |
| --- | --- |
| Titulo del recurso educativo | Interfaz de usuario |
| Descripción: | Se debe plantear bien la interfaz de usuario |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Recurso de observación - Imagen |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | http://d3fdgtuepgkuxk.cloudfront.net/wp-content/uploads/2015/04/23-21475076951-655x470.jpg |
| Titulo del recurso educativo | Empareje las palabras de acuerdo al conocimiento |
| Descripción: | A partir de los conocimientos de la lección 3 y 4 contestar de forma correcta |
| Unidad | 2 |
| Lección | 4 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | Por medio de los conocimientos adquiridos en la lección 1 y 2 por favor conteste verdadero o falso según sea el caso los siguientes enunciados. |
| URLs de ejemplo o sugerencias | 1. **Familiaridad del usuario** Utilizar lenguaje y terminos reconocidos por el usuario.  2. **El diseño de interfaz de usuario** es la construcción del contenido con el cual el usuario final va interactuar con el sistema.  3. **Cliente Hibrido** Este tipo de cliente hace referencia a estaciones de trabajo que no tienen disco y por lo tanto no almacenan información, para poder realizar su función se basa en una conexión por red para realizar las funciones.  4. **Consistencia** Todos los procesos similares deben cumplir el mismo proceso.  5. **Un cliente liviano** hace referencia a un dispositivo de hardware o software que no usa sus recursos propios para ejecutar funciones sino que realiza una conexión a otro dispositivo de hardware o software para que ejecute y realice las peticiones.  6. **Servidor** es una aplicación que ofrece un servicio a usuarios de una red, el servidor puede estar instalada en la misma máquina o puede estar instalada en otra máquina.  7. **arquitectura de software** es laorganización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán y los principios que orientan su diseño y evolución. |

**UNIDAD 3: CONSTRUCCIÓN DE SOFTWARE**

**Lección 1: Fundamentos**

En la tercera etapa del ciclo de vida del software nos encontramos con la construcción del software. Esta etapa hace referencia a la creación detallada del software basados en los requisitos que recolectamos en la primera etapa, basándonos en el diseño que construimos en la segunda etapa



|  |  |
| --- | --- |
| Titulo del recurso educativo | Gif Animado |
| Descripción: |  |
| Unidad | 3 |
| Lección | 1 |
| Tipo de recurso: | Gif Animado |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | <http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/Complemento%20Material%20Didactico/Maest-Ing-Soft-Sergio/Imagenesgrales/Construccion.gif> |

La etapa de construcción esta muy conectada a la etapa del diseño de software y a una etapa que trataremos en la próxima unidad que son las pruebas. Esto se debe a que el mismo proceso nos obliga a que revisemos constante el diseño establecido para no salirnos de la ruta o el plan previamente establecido y por otra parte por medio de las pruebas nos aseguramos que lo que construimos sea realmente lo que propusimos en la etapa de diseño.

La construcción del software debe tener como objetivo:

* Minimizar la complejidad
* Anticiparse a los cambios
* Construir para verificar
* Aplicar estándares en la construcción
* Documentar adecuadamente

|  |  |
| --- | --- |
| Titulo del recurso educativo | Casa de construcción |
| Descripción: | Cada punto es un componente de una casa por favor construyala |
| Unidad | 3 |
| Lección | 1 |
| Tipo de recurso: | Recurso de interacción – HTML5 |
| Instrucciones para el estudiante | Por medio de la construcción de la casa reforzamos el punto principal de que la construcción de un software es la unión de varios objetivos. |
| URLs de ejemplo o sugerencias | Me gustaría que cada uno de los puntos mencionados anteriormente fueran parte de una casa, y que el estudiante tenga que ir arrastrando cada punto y que a medida que se arrastre construya la casa con el fin de demostrar al estudiante que todo esos son componentes de una casa. |

1. Preparación del Terreno (Configuración de frameworks y librerias)
2. Cimentaciones (Creación de base de datos, tablas, archivos)
3. Instalacion de tuberías (Creación de mecanismos de acceso a datos y/o archivos)
4. Columnas y Vigas (Creación de lógica de negocio – Implementación casos de uso)
5. Muros y Techo (Creacíon de la interfaz de usuario)
6. Instalaciones Electricas (Comunicación entre componentes internos)
7. Instalación de puertas y ventanas (Implementación Servicios de Comunicación con otros sistemas e implementación de mecanismos de seguridad)
8. Acabados (Implementación de Presentación, Usabilidad y Experiencia de Usuario)

La construcción de software causa el volumen más alto de elementos de configuración que tienen que ser dirigidos en un proyecto de software tales como archivos de fuente de código, módulos de pruebas entre otros más.

**Lección 2: Gestión de la Construcción**

En la unidad anterior vimos que había un modelo cliente / servidor y un modelo de n capas pero igualmente hicimos énfasis en que no eran los únicos. En la gestión de la construcción de software pasa un proceso similar ya que tenemos formas distintas de construir, pero para el presente curso explicaremos el modelo de entrega por etapas y el modelo iterativo “SCRUM”.

**Modelo de entrega por etapas:** Consiste en dividir el software en etapas y tener mini ciclo de vidas, es decir que el software se va construyendo y entregando de forma tal que el cliente no tiene que esperar a que todo el desarrollo web este terminado sino que puede ir usando una parte mientras la siguiente esta en construcción.

Al tener mini ciclos de vida quiere decir que cada vez que se arranca una etapa se debe realizar todo el proceso que llevamos hasta el momento para cada etapa, es decir levantar requisitos, diseñar la arquitectura, construir el código y por último realizar pruebas.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Modelo de entrega por etapas |
| Descripción: | Muestra la forma en como se documenta la entega por etapas |
| Unidad | 3 |
| Lección | 2 |
| Tipo de recurso: | Imagen |
| Instrucciones para el estudiante | Analice con cuidado el modelo de entrega por etapas o incremental |
| URLs de ejemplo o sugerencias | http://image.slidesharecdn.com/metodologiasdedesarrollodesoftware-090813101707-phpapp01/95/metodologias-de-desarrollo-de-software-13-728.jpg?cb=1250159058 |

Este modelo tiene ventajas y desventajas:

Ventaja:

Permite que el usuario tenga una experiencia anticipada con el software y pueda ir interactuando con el y no tenga que esperar hasta el final.

Al realizar la primera etapa esto sirve a el usuario como resultado de cómo va ser el software.

Permite manejar los requisitos críticos de primero y los menos importantes posteriormente.

Desventaja:

Se puede caer en ciclos sin fin al cumplir con las expectativas del cliente solicitará más requisitos por lo cual es importante tener bien delimitado el proyecto.

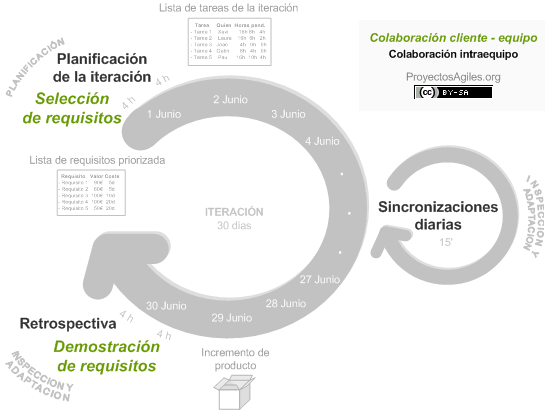
Puede ser difícil ajustar los requerimientos.

**Modelo “SCRUM”:** Es un modelo orientado a realizar proyectos agiles donde el usuario final recibe entregas del código en un tiempo menor y donde la tendencia al error es menor ya que se basa en buenas prácticas de trabajo y en trabajo colaborativo.

Por medio de SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Esta metodología de trabajo es ideal aplicarla en entornos de trabajo complejos donde el cliente necesita generalmente soluciones rápidas e inmediatas de problemas realmente complejos pero que se pueden ir solucionando en medidas pequeñas.

Con esta metodología nos aseguraremos de que el cliente reciba lo que realmente espera y así reducimos costo y aumentamos la calidad de nuestro proyecto.

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de 2 semanas, si así se necesita). Cada iteración genera un entregable y la siguiente iteración genera un incremento del producto final sin necesitar demasiado esfuerzo.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Modelo “SCRUM” |
| Descripción: | Muestra la forma en como se documenta la entega por etapas |
| Unidad | 3 |
| Lección | 2 |
| Tipo de recurso: | Recurso HTML - Interactivo |
| Instrucciones para el estudiante | Por favor organice la imagen de forma tal que pueda analizar el ciclo de vida de un proyecto por medio del modelo SCRUM. |
| URLs de ejemplo o sugerencias | El recurso interactivo sería dividr la imagen en parte de un rompecabezas de forma tal que el estudiante tenga una versión miniatura de la imagen y pueda ir construyendo el ciclo de vida.  https://scrumenespanol.files.wordpress.com/2015/09/diagrama-proceso-scrum.gif?w=553&h=414 |

Al finalizar cada iteración del modelo ágil SCRUM se realiza una actividad de retroalimentación con el cliente de forma tal que se pueda mejorar lo más que se pueda el software en la siguiente iteración

**Planificación de la construcción:** Cuando estemos construyendo el software o el desarrollo web debemos tener una planificación adecuada donde definamos la ruta crítica del proyecto y no sufra retrasos, luego de definir la ruta crítica debemos definir el modelo de construcción. El modelo de ciclo de vida que escojamos estará muy acorde a los requerimientos realizados en la etapa 1 de nuestro ciclo de vida.

**Medición de la Construcción:** Se deben generar medidas a lo largo del proyecto sobre la eficiencia y la eficacia de mismo, por ejemplo la extensión, código reutilizado, código destruido, complejidad del código, tasa de errores presentados, resultados de las pruebas alfa, pruebas beta para generar altos niveles de calidad.

ACTIVIDAD DE RETROALIMENTACIÓN

|  |  |
| --- | --- |
| Titulo del recurso educativo | Quiz única respuesta. |
| Descripción: | Por favor seleccione la respuesta adecuada. |
| Unidad | 3 |
| Lección | 2 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | Por medio de los conocimientos adquiridos en la lección 1 y 2 por favor seleccione una única respuesta. |
| URLs de ejemplo o sugerencias | 1. ¿Qué se busca en la medicicón de la construcción?    1. Generar medidas de eficiencia y eficacia.    2. Aumentar la complejidad del código.    3. Reducir costos.    4. Generar más ingresos. 2. ¿Cuánto es el tiempo máximo para el desarrollo de un proyecto con el modelo SCRUM?    1. 1 día    2. 1 mes    3. 1 año    4. 6 meses. 3. Una ventaja de la metodología incremental es    1. Generar ciclos sin fin que no permite que el proyecto se pueda cerrar en algún momento.    2. Ciclos de trabajo de 2 semanas.    3. Permitir manejar los requisitos críticos de primero y los menos importantes posteriormente. 4. Un objetivo de la construcción de software es.    1. Medir la construcción    2. No generar ningún tipo de documentación    3. Minimizar la complejidad    4. No planificar la construcción. 5. En SCRUM cuando se termina una iteración cual es el último paso que se realiza.    1. Actividad de retroalimentación    2. Cerrar el proyecto    3. Entregar los manuales de usuario.    4. Borrar los archivos fuentes. |

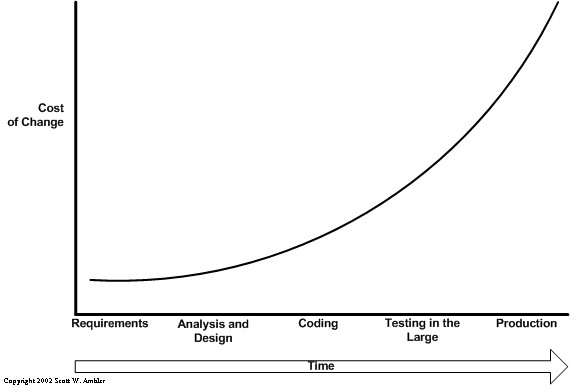
**Lección 3: Consideraciones prácticas.**

Para el desarrollo web necesariamente nos tenemos que basar en una mezcla de lenguajes, tanto de etiquetado web como de programación para poder dar solución al problema de la vida real.

Por lo anterior es básico e importante definir el lenguaje de programación que más se acomode a las necesidades de nuestro desarrollo web, seleccionar el lenguaje no adecuada nos puede significar retrasos y aumentos en los tiempos de entrega del software.

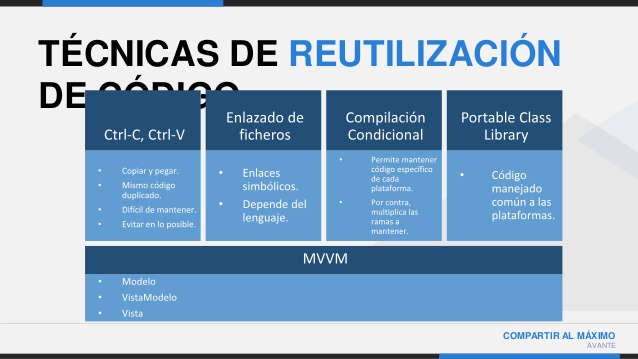
Dentro de las consideración prácticas a tener en cuenta en la construcción de software tenemos, los lenguajes de construcción, las pruebas de construcción, la reutilización y la calidad de construcción.

* **Lenguajes de construcción:** Son la forma como nos comunicamos con la máquina y le codificamos lo que necesitamos que realice de forma normal para realizar una solución ejecutable. Dentro de los lenguajes de construcción pueden existir de configuración (archivos .dll) de programación (php, java, python entre otros), de herramientas ( para diagramar casos de usos entre otras cosas) y entre otros.
* **Pruebas de construcción:** Cuando comenzamos a desarrollar el código generamos distintos tipos de prueba de la infinidad de pruebas que hay, pero estas las veremos en una unidad posterior. El propósito de realizar pruebas es el de disminuir considerablemente los errores dentro del código. Si las pruebas de código no se realizan en el momento adecuado esto contraerá gastos adicionales y aumentarán cada vez que se este más cercano a finalizar el desarrollo web.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Costo de no implementar pruebas en la etapa de construcción de software. |
| Descripción: | El gráfico anterior muestra como va aumentando considerablemente el costo de corregir errores detectados en pruebas a medida que se avanza en el ciclo de vida del software. |
| Unidad | 3 |
| Lección | 3 |
| Tipo de recurso: | Imagen |
| Instrucciones para el estudiante | A partir de la gráfica anterior imagine como sería el comportamiento de la gráfica si se realiza las pruebas en el momento adecuado. |
| URLs de ejemplo o sugerencias | http://www.initcoms.com/blog/wp-content/uploads/2014/06/Foro-para-la-calidad-del-software-2014-coste-cambio.jpg |

* **Reutilización:** Es importante generar una documentación adecuada de todo lo que se realice en el desarrollo web y sobre todo en la etapa de construcción y codificación de tal forma que se pueda optimizar lo más que pueda nuestro desarrollo, por ejemplo si en algún punto se tiene que generar un formulario, este código de formulario debería quedar como plantilla para que dentro del mismo proyecto o en futuros proyectos no tengamos que volver a realizar el esfuerzo de codificar el formulario sino solamente de editarlo. Acá realizariamos un proceso por así decirlo de RECICLAJE pero de código.

****

|  |  |
| --- | --- |
| Titulo del recurso educativo | La reutilización es como el reciclaje |
| Descripción: | Una adecuada documentación nos va permitir reusar y reciclar el código generado y reducir tiempo de trabajo |
| Unidad | 3 |
| Lección | 3 |
| Tipo de recurso: | Recurso de animación |
| Instrucciones para el estudiante | De clic para ver las técnias de reutilización de código. |
| URLs de ejemplo o sugerencias | Generar un intercambio de imágenes entre las 2 propuestas cada vez que el usuario de clic a ella.  <http://image.slidesharecdn.com/03-150608185628-lva1-app6891/95/crear-apps-mviles-multiplataforma-con-xamarin-compartiendo-la-mayor-cantidad-de-cdigo-9-638.jpg?cb=1433789847>  <https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcRhOYjMBPaNwtnqIT_eMiOjkaGCFs2TAEnNMePhsn7qUateszJc1Q> |

**Calidad de Construcción:** La calidad es fundamental en el desarrollo de código pero en el presente curso no abarcaremos este punto ya que como tal ocuparía todo un curso, por el momento diremos que un software debe cumplir con los estándares de calidad y procesos como tal y que hay distintas certificaciones que permiten sustentar el desarrollo de un código como de calidad. Existente también varias técnicas que permiten evaluar la calidad.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Rellenar espacio |
| Descripción: | Por favor rellene el espacio con la palabra correcta |
| Unidad | 3 |
| Lección | 3 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | Con los conocimientos adquiridos por favor llene el espacio con la palabra correcta. |
| URLs de ejemplo o sugerencias | 1. El lenguaje de construcción nos permite especificar una solución ejecutable para un problema de un ordenador.  2. El reuso de código nos permite reducir los tiempos de entrega de los componentes de software.  3. Entre más tiempo pase un software sin realizarse pruebas, mas aumentará su costo.  4. Como se denomina a los lenguajes como java, python entre otros: lenguaje de programación  5. Una técnica de reutilización de código es el enlazado de ficheros. |

**UNIDAD 4: PRUEBAS.**

**Lección 1: Fundamentos de prueba.**

Comenzamos diciendo que la etapa de prueba no es una etapa que venga antes de una o después de etapa, sino que es un proceso intrínseco que se realiza a lo largo de todas las etapas del ciclo de vida de software.

Pero ¿qué es una prueba?. Una prueba es un proceso por el cual ejecutamos un proceso o un sistema con unos datos y esperamos los resultado que arroje el sistema o el proceso de software y los evaluamos con los resultados que previamente esperábamos nos resultarán a partir de los datos ingresados.

Dependiendo del resultado que nos arroje la prueba, esta puede ser exitosa o no exitosa, una prueba es exitosa si se detectan errores en la misma, por el contrario no es exitosa sino se encuentran errores.

En los resultados de prueba exitosos podemos encontrar 3 tipos de casos que son:

|  |  |
| --- | --- |
| Titulo del recurso educativo | Un error que puede desencadenar en un defecto y luego en un fallo |
| Descripción: | Por medio de la linea de tiempo podemos ver como un error se puede convertir en un defecto y este a su vez termina siendo un fallo. |
| Unidad | 4 |
| Lección | 1 |
| Tipo de recurso: | Recurso de Interacción HTML Linea de tiempo |
| Instrucciones para el estudiante | Revise la diferencia entre error, defecto y fallo. |
| URLs de ejemplo o sugerencias | En esta linea de tiempo me gustaría que hubiera 3 hitos o circulos, el primer circulo sería el de error, el segundo el de defecto y el tercero el de fallo. Cuando el usuario pase por cada punto a través de una burbuja debería aparecer la definición de cada uno por ejemplo.  Error: Un error generalmente es una equivocación de codificación, una mala interpretación del requerimiento por parte del desarrollado. Por ejemplo un error es que el programador ponga que 2+2 = 5 ó que en un if de comparación ponga “<” en vez “<=”  Defecto: Cuando se genera un error este puede desencadenar un defecto que se define como una diferencia entre el funcionamiento normal del software y el funcionamiento erroneo, por ejemplo colocar un < en vez de <= en una iteración de un for significará que se hará una operación menor.  Falla: Es la discrepancia visible que se produce al ejecutar un programa con un defecto.  Macintosh HD:private:var:folders:ns:wqwmg9y93yz3hv9cxtlky1f40000gq:T:TemporaryItems:relacion.jpg  <http://3.bp.blogspot.com/_CtuqUh5fx0c/S-t16KtJjvI/AAAAAAAAAA0/y4IAYnrycxU/s320/relacion.jpg>  Ojala la imagen saliera inmediatamente debajo de la linea de tiempo y que esta fuera una ventana nueva algo así como lo que desarrollarón para la unidad 1 lección 3 de este mismo contenido digital. |

Es imposible en un desarrollo web probar todo el software, en la medida de lo posible sería lo ideal pero eso aumentaría tanto el costo como el tiempo de entrega del proyecto, esta en nuestra habilidad como desarrolladores determinar cuales son los módulos o componentes de software que vamos a desarrollar y que debemos probar para comprobar su adecuada funcionalidad.

Entonces se preguntarán ¿por qué enredarnos la cabeza con que probar y qué no y mejor no realizar pruebas? Esto tampoco puede ocurrir ya que no realizar ningún tipo de prueba aumentará considerablemente los defectos y fallos de software y por lo tanto aumentará los costos de tiempo y mantenimiento de software y entre más adelante estemos en el proyecto más costos será reparar los errores que conducen a fallos y a defectos.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Gráfica de Costo contra prevención a travé de fallas. |
| Descripción: | La gráfica muestra los resultados de implementar y no implementar pruebas y como este afecta los costos del proyecto. |
| Unidad | 4 |
| Lección | 1 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Denote la importancia de implementar y no implementar pruebas. |
| URLs de ejemplo o sugerencias | http://image.slidesharecdn.com/calidadeneldesarrollodesoftware-150615113647-lva1-app6892/95/calidad-en-el-desarrollo-de-software-26-638.jpg?cb=1434368260 |

Para generar pruebas exitosas lo más recomendable es que cada caso de uso tenga un resultado posible con el fin de poder comparar los resultados de la prueba.

El encontrar muchas pruebas exitosas no demuestra que seamos malos profesionales o desarrolladores o que nuestro software sea malo, lo único que demuestra esto es que estamos generando un código de alta calidad que nos permitirá tener clientes contentos y felices con su solución.

**Lección 2: Técnicas de Pruebas.**

Esta lección la dividiremos en 2 partes, por una parte explicaremos brevemente las técnicas de pruebas y en la segunda parte explicaremos los niveles de las pruebas.

Las técnicas de pruebas se categorizan en más de 7 técnicas pero al no ser un curso muy extenso en el presente curso solo trataremos 3 técnicas y explicaremos brevemente algunas de ellas.

Pero que es una técnica, es un conjunto de recursos o procedimientos que nos permite llegar a una solución.

|  |  |
| --- | --- |
| Titulo del recurso educativo | Técnicas |
| Descripción: |  |
| Unidad | 4 |
| Lección | 2 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | http://www.konradlorenz.edu.co/images/stories/home/proyectos\_software.jpg |

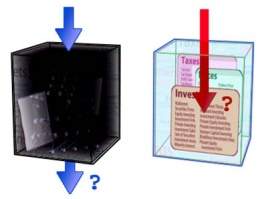
|  |  |
| --- | --- |
| Titulo del recurso educativo | Técnias y Niveles de Prueba |
| Descripción: | Recorra el menú con el fin de que conozca algunas técnicas de pruebas y los niveles de prueba |
| Unidad | 4 |
| Lección | 2 |
| Tipo de recurso: | Recurso de Interacción HTML Menú |
| Instrucciones para el estudiante | Por favor revise con cuidado el siguiente cuadro con el fin de que distinga las técnicas y niveles de prueba. |
| URLs de ejemplo o sugerencias | Pestaña 1 del Menú :  Técnicas de Pruebas:  Basadas en la experiencia o intuición: Esta técnica usada pr desarrolladores con experiencia se usan cuando no es tan claro y es mejor usar la práctica aplicada en proyectos anteriores para aplicar pruebas.  Entre estas tenemos:  **Menú Lateral:**  **Prueba ad hoc,** por medio del conocimiento previo, la persona que realiza la prueba escoge a que segmentos del código se le debe realizar ya que son procesos críticos.  **Prueba de exploración,** Arranca desde los requerimientos y pasa por el diseño para seleccionar que fragmentos se debe realizar, pero es un proceso de investigación de todas las partes del código.  **Pruebas aleatorias,** Esta técnica se basa precisamente en el azar es decir no se tiene un comportamiento claro para seleccionar que fragmento de código o componentes del software se van a trabajar con las pruebas.  **Basadas en el perfil operativo,** estas pruebas hacen referencia a las pruebas que se realizan como si el desarrollador fuera un usuario, esto con el fin de descartar cualquier comportamiento anormal y ademas de comprobar que el desarrollo web tenga todo en el funcionamiento adecuado.  **Basadas en la naturaleza de la programación,** esta técnica varía dependiendo del tipo de programación que se usa en el desarrollo del proyecto; orientada a objetos, funcional, recursivo entre otros, tambíen aplican las aplicaciones tipos web y con unidad de interfaz gráfica.  Pestaña 2 del menú:  Niveles de pruebas  Al ejecutar pruebas hemos visto que hay distintas técnicas y estas técnicas han dejado ver que hay distintos niveles de aplicación, algunos solo aplicaban para fragmentos de códigos, otros para el tipo de programación. En realidad las pruebas se categorizan en 3 grandes grupos, de unidad, de integración y de sistema.  **MENÚ LATERAL:**  **De Unidad:** Este nivel agrupa las pruebas que se ejecutan para probar fragmentos de códigos o componentes pequeños del desarrollo web.  **De integración:** Este nivel de prueba consiste en probar la interacción entre distintos componentes de software, entre ellas estan las que se realizan de arriba hacia abajo o de atrás para adelante.  **De sistema:** Este nivel ocupa todas las pruebas que evaluan el software como un todo y comprueba que el desarrollo web se ejecute como debe ser, que cumpla con todos los objetivos planteados al inicio del software y que tenga un flujo normal de trabajo. |

**Lección 3: Tipos de prueba.**

Hay varios tipos de pruebas y son demasiadas para mencionar en el presente curso además que la intención no es atiborrarlos de información que por la misma complejidad de las mismas no van a poder entender, por ejemplo hay pruebas de software que solo están enfocadas en mejorar el rendimiento del software a través del manejo adecuado de los recursos de hardware.

Por lo anterior en esta lección hablaremos de 2 tipos de pruebas.

La primera de la que hablaremos es la prueba de caja blanca esta se basa en como diseñamos los casos de uso esta prueba también es conocida como caja de cristal. En la prueba de caja blanca el código de programación es fundamental y se centraliza más en identificar que los bucles se ejecuten en sus limites operacionales, realizar los caminos de decisión tales como los if por las ramificaciones posibles con el fin de comprobar que en todas se consiga el resultado esperado.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Caja Blanca |
| Descripción: |  |
| Unidad | 4 |
| Lección | 3 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante |  |
| URLs de ejemplo o sugerencias | http://www.ecured.cu/images/thumb/7/76/Caja-blanca-caja-negra.jpg/260px-Caja-blanca-caja-negra.jpg |

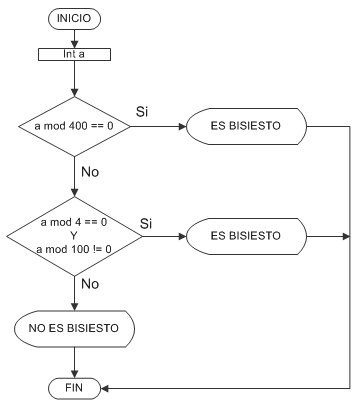
A partir del código o del caso de uso se dibuja un diagrama de flujo como si estuviera realizándose antes de programar pero el objetivo de este diagrama de flujo es el de ayudarnos a realizar las pruebas de escritorio con el objetivo de comprobar el funcionamiento adecuado.

Realizar una prueba de caja blanca parece relativamente fácil cuando son códigos cortos y sin muchas bifurcaciones pero supongamos que estamos ante un código con más de 900 líneas y con varios if anidados y varios ciclos ¿se acomplejiza la situación, cierto? Para esto tenemos el proceso de Complejidad Ciclomática

La complejidad ciclomática se define como una medición cuantitativa de complejidad lógica de un programo, en otras palabras nos dice la cantidad de caminos diferentes que tiene un código de software por medio de la siguiente formula.

v(G) = e – n + 2, donde e representa el número de aristas y n el número de nodos.

Un nodo es el circulo o cuadrado dentro de un diagrama de flujo, y una arista es la flecha o unión entre cada nodo.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Ejemplo de Caja Blanca |
| Descripción: | Revisar con cuidado el ejemplo de caja blanca. |
| Unidad | 4 |
| Lección | 3 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Es importante que entienda como funciona la complejidad ciclomática |
| URLs de ejemplo o sugerencias | http://joaquinoriente.com/wp-content/uploads/2012/11/Graf\_algo\_bisiesto.png |

Para el ejemplo anterior el calculo de la complejidad ciclomática sería la siguiente

v(G) = 8 -7 + 2 = 3, siendo e = 8; n = 7

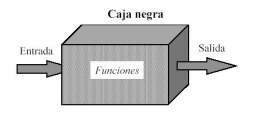
Entre mayor sea el resultado, mayor será el riesgo de que el código se vuelva inestable.

**Pruebas de caja negra:** Son complementarías a las pruebas de caja blanca y están diseñadas para tratar de descubrir diferentes tipos de errores a los encontrados en la caja blanca.

Las pruebas de caja negra permiten probar la funcionalidad del código a partir de unos datos y ver el resultado que estos arrojan y comprobar que el resultado arrojado sea el correcto, como su nombre lo dice, a la prueba no le importa lo que hay dentro de la caja, es decir no se enfoca en como se realiza el resultado sino en el resultado mismo y la interacción para llegar a este resultado.

Para la ejecución de la prueba de caja negra solo es necesario tener un dato de entrada y esperar la ejecución.

Supongamos que realizamos un desarrollo web de una calculadora y vamos a realizar la suma de 2 + 2, el resultado esperado es el 4 pero al realizar la prueba da un defecto y nos arroja por error 5. La caja negra no nos permitiría saber que esta mal con el código solo nos mostraría que hay algo malo con el código.



|  |  |
| --- | --- |
| Titulo del recurso educativo | Caja negra |
| Descripción: | Muestra de cómo funciona la caja negra |
| Unidad | 4 |
| Lección | 3 |
| Tipo de recurso: | Imagen gif |
| Instrucciones para el estudiante | Es importante que entienda como funciona la complejidad ciclomática |
| URLs de ejemplo o sugerencias | http://www.ecured.cu/images/thumb/8/8e/Cb2.jpg/260px-Cb2.jpg |

|  |  |
| --- | --- |
| Titulo del recurso educativo | Preguntas de selección multiple con multiple respuestas. |
| Descripción: | Seleccione 2 respuestas verdaderas para los siguientes enunciados. |
| Unidad | 4 |
| Lección | 3 |
| Tipo de recurso: | Actividad de retroalimentación. |
| Instrucciones para el estudiante | Por medio de los conocimientos adquiridos en la lección 1, 2 y 3 por favor conteste de forma adecuada los enunciados |
| URLs de ejemplo o sugerencias | 1. Cuando estamos realizando pruebas hay:    1. Técnias    2. Niveles    3. Categorías    4. Pruebas 2. ¿Cuál de las siguientes hacen parte de los niveles de pruebas?.    1. AD hOC    2. Experiencia    3. Unitarias    4. Integrales 3. Un error es:    1. Cuando el usuario codifica de forma erronea    2. Cuando se confunden las variables    3. Cuando hay fallas en la programación    4. No se compila el código 4. Para encontrar la complejidad ciclomática requerimos de:    1. Nodos    2. Simbolos    3. Aristas    4. Caja negra 5. Entre los tipos de prueba podemos encontrar:    1. Prueba de caja blanca    2. Pruebas de ad hoc    3. Pruebas de resistencia    4. Prueba de caja negra 6. La complejidad ciclomática es mas maleable cuando:    1. El valor es menor    2. Hay menor cantidad de aristas y nodos    3. Se presentan más aristas y nodos    4. El valor es mayor |

Bibliografia

<http://es.slideshare.net/juanpabloov18/resumen-swebok-original>

<http://es.scribd.com/doc/190297682/SWEBOK-en-Espanol-Completo#scribd>

<http://es.slideshare.net/lcastillo110/diseo-del-software-e-ingeniera-del-software?next_slideshow=1>

<http://www.codecompiling.net/files/slides/UML_clase_02_UML_casos_de_uso.pdf>

<http://www.oocities.org/es/monsalvelaura/fase2/analisis.html>

<http://es.slideshare.net/FABIANGARCIA/diagramas-de-secuencia-presentation>

<http://es.slideshare.net/buster_blader99/diagrama-de-secuencia-uml>

<http://www.sparxsystems.com.ar/download/ayuda/index.html?sequenceelements.htm>

<http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372008000200008>

<http://www.solvetic.com/tutoriales/article/555-uml-diagrama-de-distribucion/>

<http://www.monografias.com/trabajos67/diagramas-uml/diagramas-uml2.shtml>

<http://www.abc.es/tecnologia/redes/20141203/abci-gangnam-style-rompe-youtube-201412031647.html>

<https://msdn.microsoft.com/es-es/library/aa290752(v=vs.71).aspx>

<http://isab07.blogspot.com.co/2007/06/diseo-de-datos.html>

<http://www.desarrolloweb.com/articulos/1622.php>

<http://www.ecured.cu/Arquitectura_de_software>

<https://www.fing.edu.uy/tecnoinf/mvd/cursos/ingsoft/material/teorico/is05-ArquitecturaDeSoftware.pdf>

<http://es.ccm.net/contents/144-redes-cliente-liviano>

<http://blog.netsein.com/que-son-los-clientes-ligeros-thin-client/>

<http://www.academia.edu/10102692/Arquitectura_de_n_capas>

<http://es.slideshare.net/adrianazamora/diseo-de-interfaz-importancia-y-proceso-24058181>

<http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/Complemento%20Material%20Didactico/Maest-Ing-Soft-Sergio/Cuerpoconocimiento/Construcci%C3%B3n%20del%20software.htm>

<http://proyectosagiles.org/que-es-scrum/>

<http://modeloentregaporetapas.blogspot.com.co/>

<http://equipomaquila.blogspot.com.co/2010/05/relacion-entre-defecto-falla-y-error.html>