INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

DESARROLLO DE SOFTWARE

Instrumento para facilitar el proceso de enseñanza-aprendizaje de la asignatura

GUÍA GENERAL DE ESTUDIO DE LA ASIGNATURA 20230021

ANÁLISIS Y DISEÑO DE SISTEMAS

Período académico
Primero

2023 IIS

ING.FERNANDO NÚÑEZ COLLANTES, Mg.

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA – ANÁLISIS Y DISEÑO DE SISTEMAS
INSTITUTO SUPERIOR TECNOLÓGICO TENA

Carrera Desarrollo de Software

ISTT DSW Primera Edición – Tena, agosto 2023

SIN ISBN

Instituto Superior Tecnológico Tena
Km. 1 1/2 Vía Tena - Archidona
Tena, Ecuador

AUTOR(ES) - REDACCIÓN Y FORMULACIÓN DE CONTENIDOS
Ing. Fernando Núñez Collantes, Mg.

Profesor del Instituto Superior Tecnológico Tena

REVISIÓN DE PARES

Lcdo. Segundo Calisto Rochina Chileno
Mg. Alvaro Santiago Toalombo Díaz
Mg. Henry Fabian Chango Chango
Ing. Agustín Gonzalo Guanipatin Ramirez

Comisión de revisión técnica de guías de estudio del Instituto Superior Tecnológico Tena

APROBACIÓN

Mg. Danilo Alexander Zamora Núñez
Coordinador de Investigación, Desarrollo Tecnológico e Innovación

Impreso y hecho en Ecuador.

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA DE ANÁLISIS Y DISEÑO DE SISTEMAS

DATOS GENERALES DE LA ASIGNATURA							
Carrera	Desarrollo de Software			**Nombre asignatura**	Análisis y Diseño de Sistemas		
Modalidad	Presencial			**Campo de Formación**	S/N		
Jornada	Matutina y Nocturna			**Unidad de Organización Curricular**	Profesional		
Período académico	Primero A y B			**Código de la asignatura**	DSW104		
Distribución de horas en las actividades de aprendizaje				**N° Total de horas de la asignatura**	192		
N° de horas Docencia	**64**	**N° de horas Aprendizaje Práctico Experimental**				**N° de horas Autónomo**	**40**
		En contacto con docente	**48**	**Autónomo**	**40**		

PRERREQUISITOS Y CORREQUISITOS			
Prerrequisitos de la asignatura		**Correquisitos de la asignatura**	
Asignatura	**Código**	**Asignatura**	**Código**
		Fundamentos de Programación	DSW103

DESCRIPCIÓN DE LA ASIGNATURA

La asignatura de Análisis y Diseño de Sistemas da los Fundamentos acerca de cómo usar las computadoras dentro de la organización, para mejorar la productividad y el alcance de los objetivos de la misma. A través de ésta, el alumno conocerá la teoría necesaria para entender al usuario potencial y a la tecnología, además de conseguir integrar un mejor diseño de los sistemas de información.

OBJETIVO GENERAL

Analizar la situación actual de un problema y aplicar alternativas de solución informática.

CONTRIBUCIÓN DE LOS RESULTADOS DE APRENDIZAJE DE LA ASIGNATURA AL PERFIL DE EGRESO DE LA CARRERA

Resultados de aprendizaje de la asignatura	**Resultados de aprendizaje del perfil de egreso de la carrera**	**Contribución** (alta – media – baja)
Conoce los procesos principales de una empresa y sus interrelaciones	Aplica técnicas de investigación en la búsqueda de nuevas formas de aplicación del desarrollo de software en los sectores industriales.	Alta Alta

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

Conoce estructuras empresariales, Ilustra la estructuración de relaciones de la empresa con sus proveedores	Determina los recursos necesarios para el desarrollo de un proyecto software, considerando el hardware, el software y las redes.	Alta
Aplica alternativas de solución a problemas identificados	Realiza procesos de análisis y verificación de consistencia de datos extraídos de diversas fuentes, que permitan generar reportes relevantes para una organización. Aplica habilidades de Tics, trabajo en equipo, gestión de proyectos, liderazgo y creatividad, para trabajar en ambientes colaborativos con profesionalismo y responsabilidad social.	Media

CONTENIDOS DE LA ASIGNATURA (descripción mínima de contenidos de la asignatura)

Unidad 1: INTRODUCCIÓN A ANÁLISIS DE SISTEMAS

1.1 Sistemas de Información
1.2 Definición de Sistemas de Información
1.3 Clasificación de Sistemas de Información
1.4 Elementos de los Sistemas de Información
1.5 Análisis de sistemas
1.6 Conceptos de Análisis de sistemas
1.7 Técnicas de recopilación de información
1.8 Encuesta
1.9 Entrevista
1.10 Observación
1.11 Determinación de los requerimientos
1.12 Requerimientos Funcionales
1.13 Estudio de facilidad
1.14 Ciclo de vida del software
1.15 Tipos de ciclos de vidas del software

Unidad 2: DISEÑO DE DIAGRAMA UML

2.1 Conceptos y características de UML
2.2 Clasificación del UML
2.3 Diagrama de con texto
2.4 Diagrama cero
2.5 Diagrama lógico y físico
2.6 Ejemplos de Diagrama lógico y físico
2.7 FLISOL
2.8 Ejemplo de Diagrama de caso de uso
2.9 Diagrama de clase
2.10 Ejemplos con Diagrama de clase
2.11 Diagrama de actividad
2.12 Ejemplos de Diagrama de actividades.
2.13 Modelo Conceptual (entidades, atributos)
2.14 Entidades
2.15 Ejercicios de aplicación con Entidades
2.16 Atributos
2.17Ejercicio de aplicación atributos

Unidad 3: EL PROCESO DE DISEÑO DE UN SISTEMA INFORMÁTICO

3.1 Estándares de diseño de interfaces
3.2 Diseño de procedimientos precisos para la entrada de datos
3.3 Diseño de Interfaz
3.4 Diseño de salida

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

3.5 Diseño de Diccionario de datos

ESTRATEGIAS METODOLÓGICAS Y RECURSOS DIDÁCTICOS		
ESTRATEGIAS METODOLÓGICAS	**HABILIDADES BLANDAS**	**FINALIDAD**
Activas para la enseñanza y aprendizaje	**Valores vinculados a la autonomía del sujeto**: confianza, crítica y autocrítica, honestidad, integridad	• Generar confianza/ Promover el pensamiento crítico. • Permite a los estudiantes cumplir un rol activo dentro de su formación. • Construye una sociedad participante.
Aprendizaje y trabajo cooperativo	**Valores elementales de convivencia y civilidad:** crítica y autocrítica, tolerancia, empatía, respeto, justicia, lealtad, paciencia	• Promover un ambiente de colaboración/ trabajo en equipo/ Saber escuchar/Promover el pensamiento crítico/ fomentar el liderazgo/ adaptabilidad. • Mantener una comunicación abierta con el equipo/ tolerancia a los errores, aceptar y aprender de las críticas. • Fomentar el sentido de pertenencia
Aprendizaje individual	**Valores vinculados a la autonomía del sujeto**: responsabilidad, honestidad, integridad, efectividad, autonomía	• Facilitar la asimilación del contenido por parte del estudiante/ Plantear preguntas para promover la comunicación efectiva /Promover el pensamiento crítico • Lectura comprensiva para fijar contenidos/ Promover el pensamiento crítico

RECURSOS DIDÁCTICOS	
MATERIALES CONVENCIONALES	*Material impreso: libros, folletos, fotocopias, periódicos, etc.*
	Tableros didácticos: pizarra
MATERIALES AUDIOVISUALES	*Imágenes fijas proyectables (fotos): diapositivas y fotografías.*
	Materiales audiovisuales (vídeo): películas y videos
NUEVAS TECNOLOGÍAS	*Programas informáticos: procesador de palabras, presentaciones. Aplicaciones de Diseño : Mockflow*
	Servicios telemáticos: páginas web, plataforma EVA, correo electrónico

BIBLIOGRAFÍA		
Bibliografía Básica de la Asignatura:	**Físico**	**Digital**
• Guillermo P., Ludmila R. (2015). *Ingeniería de Software*. Argentina: Alfaomega Grupo Editor. ISBN: 978-987-1609-78-9. **Código de inventario en Biblioteca ISTT-DS-0040.**	**X**	
	X	

• José B. (2022) – *Alfabetización y competencias Digitales*, Colombia: Ediciones Rama Editorial, ISBN: 978-958-792-438-1. **Código de inventario en Biblioteca ISTT-DS-0252**		
Bibliografía de consulta de la Asignatura:	**Físico**	**Digital**
• Roger P. (2006). *Ingeniería del Software. Un Enfoque Práctico*, México: McGraw-Hill. ISBN: 970-10-5473-3. **Código de inventario en Biblioteca ISTT-DS-0032**	X	
• Kenneth K. y Julie K. (2011). *Análisis y Diseño de Sistemas*, México: Pearson Educación, **ISBN: 978-607-32-0577-1.**	X	

DESCRIPTIVA DE LAS COMPETENCIAS DE LA GUÍA DE ANÁLISIS Y DISEÑO DE SISTEMA

La presente guía está diseñada para desarrollar competencias clave en el estudiante, orientadas a la comprensión y aplicación de técnicas de análisis y diseño de sistemas de información. Estas competencias abarcan habilidades técnicas, metodológicas y transversales que permiten identificar problemas, recopilar requisitos, modelar soluciones y diseñar sistemas eficientes que respondan a las necesidades de una organización. A través de estas competencias, se busca formar profesionales capaces de abordar los retos tecnológicos con un enfoque crítico, colaborativo y creativo, garantizando soluciones alineadas con los objetivos del entorno actual.

Competencias Específicas

Unidad 1: Introducción a Análisis de Sistemas

- Identificar los conceptos fundamentales de sistemas y análisis de sistemas, comprendiendo su importancia en el desarrollo de soluciones tecnológicas.
- Reconocer los elementos y etapas del análisis de sistemas dentro del ciclo de vida de desarrollo de sistemas.
- Aplicar técnicas básicas para la recopilación de requisitos, como entrevistas, cuestionarios y observación.
- Analizar y documentar las necesidades del usuario y del sistema de forma estructurada.

Unidad 2: Diseño de Diagramas UML

- Comprender los principios básicos del Lenguaje Unificado de Modelado (UML) y su utilidad en el análisis y diseño de sistemas.
- Diseñar diagramas UML fundamentales, como diagramas de casos de uso, diagramas de clases y diagramas de secuencia, para modelar sistemas informáticos.
- Interpretar diagramas UML para comunicar de manera clara y efectiva la estructura y funcionalidad de un sistema.

- Integrar diagramas UML en el proceso de análisis y diseño para garantizar una representación precisa de los requisitos del sistema.

Unidad 3: El Proceso de Diseño de un Sistema Informático

- Planificar el diseño de un sistema informático basándose en los resultados del análisis de sistemas.
- Seleccionar herramientas y metodologías adecuadas para diseñar soluciones alineadas con las necesidades del usuario.
- Crear prototipos funcionales que representen la solución propuesta, priorizando la usabilidad y la eficiencia.
- Evaluar el diseño del sistema para garantizar su viabilidad técnica, económica y funcional.

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

UNIDAD 1: INTRODUCCIÓN A ANÁLISIS DE SISTEMAS

1.1 Sistemas de Información
1.2 Definición de Sistemas de Información
1.3 Clasificación de Sistemas de Información
1.4 Elementos de los Sistemas de Información
1.5 Análisis de sistemas
1.6 Conceptos de Análisis de sistemas
1.7 Técnicas de recopilación de información
1.8 Encuesta
1.9 Entrevista
1.10 Observación
1.11 Determinación de los requerimientos
1.12 Requerimientos Funcionales
1.13 Estudio de facilidad
1.14 Ciclo de vida del software
1.15 Tipos de ciclos de vidas del software

Resultado de Aprendizaje

El estudiante será capaz de conocer los conceptos básicos referentes al Análisis y Diseño de Sistemas

DIAGRAMA DE APRENDIZAJE

SÍNTESIS

La asignatura de Análisis y Diseño de Sistemas se centra en identificar, comprender y estructurar las necesidades de un sistema para proponer soluciones tecnológicas eficientes. A través del análisis, se descomponen los problemas en componentes manejables, y mediante el diseño, se crean modelos y estructuras que permiten su implementación. Esta disciplina utiliza herramientas como diagramas de flujo, casos de uso y diagramas de clases, combinando metodologías tradicionales y ágiles para garantizar que los sistemas desarrollados sean funcionales, escalables y alineados con los objetivos organizacionales.

UNIDAD 1: INTRODUCCIÓN A ANÁLISIS DE SISTEMAS

1.1. Sistemas de Información

El análisis de sistemas es una etapa crucial en el desarrollo de soluciones tecnológicas, que consiste en estudiar, comprender y documentar el funcionamiento de un sistema, ya sea existente o en fase de planificación. Este proceso permite identificar las necesidades y problemas de los usuarios, así como definir los requisitos que el sistema debe cumplir. A través de herramientas y técnicas específicas, como diagramas de flujo, casos de uso y entrevistas, el análisis de sistemas busca descomponer procesos complejos en elementos más simples para facilitar su evaluación y mejoramiento. Esta etapa no solo sirve de base para el diseño y la implementación de sistemas efectivos, sino que también ayuda a garantizar que las soluciones estén alineadas con los objetivos organizacionales y las expectativas de los usuarios.

1.2. Definición de Sistemas de Información

Los **Sistemas de Información (SI)** son herramientas fundamentales en el entorno empresarial y tecnológico, ya que permiten gestionar datos de manera organizada para transformarlos en información útil. Su definición abarca múltiples perspectivas, ya que no solo se enfocan en la tecnología, sino también en las personas y procesos que los utilizan para alcanzar objetivos específicos.

Un sistema de información integra componentes como hardware, software, datos, personas y procesos para facilitar la toma de decisiones, mejorar la eficiencia operativa y generar ventajas

competitivas. Desde aplicaciones simples hasta infraestructuras complejas, los sistemas de información son el corazón de la gestión moderna en cualquier ámbito.

1.3. Clasificación de Sistemas de Información

Los Sistemas de Información se clasifican según su propósito y funciones dentro de una organización. Esta clasificación ayuda a entender cómo cada tipo de sistema puede ser aplicado para resolver problemas específicos o apoyar procesos organizativos.

1.3.1Tipos de Sistemas de Información:

- **Sistemas transaccionales (TPS):** Manejan operaciones diarias, como compras y ventas
- **Sistemas de información gerencial (MIS):** Generan repostes para la toma de decisiones
- **Sistemas de apoyo a decisiones (DSS):** Ayudan en problemas complejos y decisiones estratégicos.
- **Sistemas expertos:** Simulan la toma de decisiones humanas usando inteligencia artificial.

1.3.2 Por tipo de actividad:

- Sistemas de gestión empresarial.
- Sistemas de soporte a la toma de decisiones.

1.4 Elementos de los Sistema de Información

Un sistema de información se compone de cinco elementos clave:

1. **Hardware:** Dispositivos físicos utilizados para procesar información.
2. **Software:** Programas y aplicaciones que gestionan datos.
3. **Datos:** Información estructurada o no estructurada.
4. **Personas:** Usuarios y administradores que interactúan con el sistema.
5. **Procesos:** Métodos y procedimientos para gestionar información.

1.5 Análisis del Sistema

El análisis de sistemas es el estudio detallado de los componentes y procesos de un sistema con el fin de entender cómo funciona, identificar problemas y proponer soluciones. Es el primer paso en el desarrollo de un sistema eficiente y efectivo.

1.6 Conceptos de Análisis de Sistemas

El análisis de sistemas involucra términos como entradas, salidas, retroalimentación, y subsistemas. También incluye enfoques como el análisis estructurado y el orientado a objetos.

1.7 Técnicas de Recopilación de Información

Las técnicas de recopilación de información son esenciales para comprender las necesidades de los usuarios y los requisitos del sistema.

1.8 Encuesta

Técnica basada en un formulario de preguntas prestablecidas, mayoritariamente cerradas, que cuenta con una categoría de respuesta limitadas.

- Instrumento: Se utiliza para obtener la misma información de un número.
- Utilidad: la encuesta se utiliza para obtener la misma in formación de un número, pequeño pero representativo de individuos(muestra) de un colectivo o grupo(población). Para ello se utiliza herramientas estadísticas.

  1.8.1 Tipos de encuestas

  - ✓ Descriptiva: es la encuesta que tiene como objetivo establecer la distribución de una realidad concreta en una determinada muestra de población.
  - ✓ Explicativa: este tipo de encuetas intenta dar a conocer las causas por las que se reproduce un fenómeno determinado.

1.9 Entrevista

La entrevista es un método cualitativo que permite obtener información detallada y personalizada directamente de los usuarios.

1.9.1 Ventajas: Permite el conocimiento de ciertos aspectos de la realidad que pertenecen al ámbito interno, subjetivo y mental de las personas, como pensamientos, sentimientos, creencias, impresiones o intenciones de la persona, describirlas interpretarlos si es necesario.

1.9.2 Tipos de entrevistas

- ✓ Entrevista no estructurada:
  - Secuencia o se caracteriza porque es mas abierta y flexible. Este tipo de entrevistas parte de preguntas abiertas que generalmente no responden a ningún estándar o secuencia formal previa.
  - Tipo: a entrevista focalizada, la la entrevista no dirigida
- ✓ Entrevista Estructurada:
  - Entrevista abierta: las preguntas son abiertas, lo que permite que la persona entrevistada pueda expresarse con sus propias palabras

- Entrevista estructurada no secuenciada: puede alterar la secuencias de las preguntas en función como discurra la entrevista, facilita la naturalidad y la flexibilidad.

1.10 Observación

Consiste en analizar las actividades y procesos en su contexto natural, identificando dinámicas y problemas.

- Técnica: es el conjunto de instrumentos y medios a través de los cual se efectua el método y solo aplica a una ciencia.
- Técnicas de la investigación: la técnica es indispensable en el proceso de la investigación científico, ya que integra la estructura por medio de la cual se organiza la investigación, la técnica pretende los siguientes objetivos:
  - Ordenar las etapas de la investigación.
  - Aportar instrumentos para manejas la información.
  - Llevar un control de los datos.
  - Orientar la obtención de conocimiento

1.10.1 La Observación científica puede ser:

- Dirección directa o Indirecta.
- Participante o no Participante.
- Estructurada o no Estructurada
- De Campo o de Laboratorio.
- Individual o de Equipo.

1.10.2 Observación Directa y la Indirecta

Es **directa** cuando el investigador se pone en contacto personalmente con el hecho o fenómeno o que trata de investigar.

Es **indirecta** cuando el investigador entra en conocimiento del hecho o fenómeno observando a través de las observaciones realizadas anteriormente por otra persona.

1.10.3 Observación participante y no Participante:

Es participante cuando para obtener los datos el investigador se incluye en el grupo, hecho o fenómeno, observando para conseguir la información "desde adentro".

La observación no participante es aquella en la cual se recoge la información desde afuera, sin intervenir para nada en el grupo social, hecho o fenómeno investigado.

1.11 Determinación de los Requerimientos

La determinación de requerimientos busca identificar qué espera el usuario del sistema, incluyendo funcionalidades y restricciones.

1.12 Requerimientos Funcionales

Describen qué debe hacer el sistema, incluyendo entradas, procesos y salidas específicas.

1.12.1 Importancia de definir los requerimientos funcionales

Los problemas y errores en la gestión de requerimientos funcionales son citados como una de las causas más frecuentes que ocasionan insatisfacción de las expectativas de los interesados en proyectos de software.

Profundizando en las causas de estos problemas, las situaciones observadas con mayor frecuencia son:

- Requerimientos funcionales con descripciones muy ambiguas, produciendo interpretaciones inadecuadas por parte del equipo de desarrollo.
- El requerimiento funcional no fue entendido adecuadamente cuando fue levantado con el interesado, pasando información incorrecta al equipo de desarrollo.
- En su forma original, el requerimiento funcional no era **factible técnicamente** y el equipo de desarrollo realizó modificaciones que no fueron aprobadas por los interesados.

1.13 Estudio de Factibilidad

Evalúa la viabilidad técnica, económica y operativa de implementar un sistema.

1.13.1 Tipos de Factibilidad

- Factibilidad **Técnica**:
  - Evalúa si la tecnología disponible puede satisfacer los requerimientos del sistema.
  - Incluye análisis de hardware, software, redes y recursos técnicos necesarios.

- Factibilidad **Económica**:
  - Analiza los costos del proyecto en relación con los beneficios esperados.
  - Utiliza métodos como análisis de costo-beneficio o retorno de inversión (ROI)

- Factibilidad **Operativa**:
  - Determina si el sistema propuesto será aceptado y utilizado por los usuarios.
  - Considera la integración con procesos existentes y la capacitación necesaria.

- Factibilidad **Legal**:
  - Verifica el cumplimiento de regulaciones, leyes y normativas aplicables al sistema.

- Factibilidad **de Cronograma**:
  - Estudia si el proyecto puede completarse en el tiempo disponible, considerando recursos y etapas.

1.14 Ciclo de Vida del Software

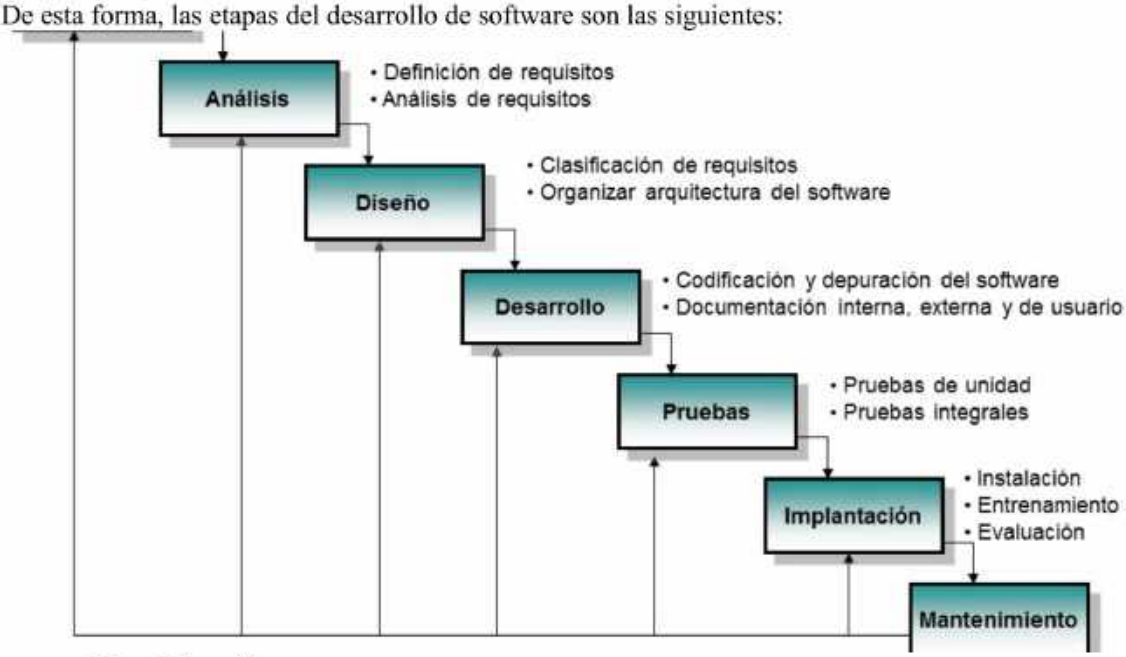
Es un marco estructurado que define las fases para desarrollar y mantener un software, como el análisis, diseño, implementación y mantenimiento.

➢ **Fases de desarrollo de software**

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con grandes posibilidades de éxito. Esta sistematización indica cómo se divide un proyecto en módulos más pequeños para normalizar cómo se administra el mismo.

Así, una metodología para el desarrollo de software son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

De esta forma, las etapas del desarrollo de software son las siguientes:



1. Planificación

Antes de empezar un proyecto de desarrollo de un sistema de información, es necesario hacer ciertas tareas que influirán decisivamente en el éxito del mismo. Dichas tareas son conocidas como el *fuzzy front-end* del proyecto, puesto que no están sujetas a plazos.

Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

2. Análisis

Por supuesto, hay que averiguar qué es exactamente lo que tiene que hacer el software. Por eso, la etapa de análisis en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

3. Diseño

En esta fase se estudian posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa.

Es posible que la solución inicial no sea la más adecuada, por lo que en tal caso hay que refinarla. No obstante, hay catálogos de patrones de diseño muy útiles que recogen errores que otros han cometido para no caer en la misma trampa.

4. Implementación

En esta fase hay que elegir las herramientas adecuadas, un entorno de desarrollo que facilite el trabajo y un lenguaje de programación apropiado para el tipo de software a construir. Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.
Al programar, hay que intentar que el código no sea indescifrable siguiendo distintas pautas como las siguientes:

- Evitar bloques de control no estructurados.
- Identificar correctamente las variables y su alcance.
- Elegir algoritmos y estructuras de datos adecuadas para el problema.
- Mantener la lógica de la aplicación lo más sencilla posible.
- Documentar y comentar adecuadamente el código de los programas.
- Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.

También hay que tener en cuenta la adquisición de recursos necesarios para que el software funcione, además de desarrollar casos de prueba para comprobar el funcionamiento del mismo según se vaya programando.

5. Pruebas

Como errar es humano, la fase de pruebas del ciclo de vida del software busca detectar los fallos cometidos en las etapas anteriores para corregirlos. Por supuesto, lo ideal es hacerlo antes de que el usuario final se los encuentre. Se dice que una prueba es un éxito si se detecta algún error.

6. Instalación o despliegue

La siguiente fase es poner el software en funcionamiento, por lo que hay que planificar el entorno teniendo en cuenta las dependencias existentes entre los diferentes componentes del mismo.
Es posible que haya componentes que funcionen correctamente por separado, pero que al combinarlos provoquen problemas. Por ello, hay que usar combinaciones conocidas que no causen problemas de compatibilidad.

7. Uso y mantenimiento

Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

- Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
- Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
- Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Aunque suene contradictorio, cuanto mejor es el software más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.

1.15 Tipos de Ciclo de Vida de Software

- **Modelo Cascadas:**

  El desarrollo en cascada (en inglés, waterfall model) es un **procedimiento lineal** que se caracteriza por dividir los procesos de desarrollo en sucesivas fases de proyecto. Al contrario que en los modelos iterativos, cada una de estas fases se ejecuta tan solo una vez. Los resultados de cada una de las fases sirven como hipótesis de partida para la siguiente. El waterfall model se utiliza, especialmente, en el desarrollo de software.

secretaria.general@itstena.edu.ec km 1 1/2 vía Tena- Archidona ISTT

✓ **Fases de modelos Cascada**

Podemos dividir la modelo cascada o waterfall en varias fases. Es posible que diferentes expertos unan las fases o que les denominan de otra manera, pero el proceso siempre va a ser el mismo. Nosotros las dividimos en 6 fases:

Requisitos

Como ya hemos mencionado es la fase más importante. Durante esta fase normalmente se realizan entrevistas, reuniones e intercambio de opiniones para definir los requisitos para el proceso de desarrollo y el resultado final del proyecto. Se analizan los requisitos recopilados y documentados. Después se decide qué tareas habrá que completar para llegar al resultado final, se establece el plan de proyecto con los costes y cesionarios para cada tarea.

Diseño y construcción

Esta etapa puede contener procesos de implementación, desarrollo y codificación. Cabe mencionar, que la implementación aquí no significa que empezamos a utilizar el resultado, sino que empezamos a trabajar en el desarrollo del producto a base de requerimientos y diseño.

Fase de prueba

En esta etapa, los especialistas responsables prueban el software (u otro producto que se desarrolla en el proyecto) y detectan errores. Aquí es fundamental asegurarse de que el producto cumpla con todos los requisitos del cliente.

Instalación / implantación

Es una fase en la que el producto sale para el uso de acuerdo con todos los requisitos. Algunos procesos de prueba pueden tener lugar en esta etapa.

Soporte y mantenimiento

El producto final se entrega al cliente. Dependiendo del tipo de proyecto, se pone en marcha el mantenimiento y el soporte. Si todo está bien, el producto sigue funcionando según lo diseñado. Para algunos proyectos, por ejemplo, un software, se necesita el mantenimiento continuo.

Las ventajas de la metodología de cascada

Las principales ventajas de la metodología cascada de gestión de proyectos son las siguientes:

- El modelo es simple y fácil de usar.
- Como la metodología es bastante rígida, es fácil de administrar porque cada fase consta de entregables específicos.
- El proceso es bastante predecible, todos tienen una idea con anterioridad cómo se evolucionará el proyecto. Los clientes saben qué esperar en cuanto a los costes, el cronograma, y el resultado final de su proyecto desde el principio. El equipo sabe bien cómo son y cuando tienen que hacer sus tareas.
- Las fases no se superponen. Se ejecutan y se completan una a la vez.
- Las metodologías de desarrollo de software en cascada son buenas para proyectos que contienen requisitos claros.
- Si la rotación de empleados en su empresa es bastante frecuente, al estar todo bien definido y documentado, eso impactará mínimamente el proyecto.

Las desventajas de la metodología de cascada

A pesar de todos los puntos fuertes enumerados anteriormente, también hay algunas desventajas:

- Si encuentra un error de requisito o necesita cambiar algo, su proyecto debe iniciarse desde el principio con un nuevo código.
- Cuando su producto está en la etapa de prueba, no es fácil volver atrás y cambiar algo que no está claro o no se ha formulado bien en la fase inicial.
- No puede resolver algunos problemas esenciales utilizando Waterfall para proyectos complejos y orientados a objetos. Tampoco es una buena idea usarlo para proyectos largos con requisitos complejos e imprecisos.
- El método no es apropiado para los proyectos en los que se sabe desde inicio que hay muchas probabilidades que los requisitos cambien.
- Los clientes pueden no estar satisfechos con el producto entregado. Como todas las tareas y los entregables se basan en requisitos documentados, es posible que los clientes no vean lo que se entregará hasta que esté casi terminado. Puede ser difícil cambiar algo en ese momento.

- **Modelo Iterativo**

El **Modelo Iterativo** es un enfoque para el desarrollo de software que permite construir el sistema en ciclos repetitivos o iteraciones, en lugar de completarlo en una sola fase continua. Cada iteración incluye el desarrollo de una parte funcional del sistema, que se mejora y refina en las siguientes iteraciones.

Características Clave:

1. **División en Iteraciones**:
   - El proyecto se divide en ciclos más pequeños y manejables.

   - Cada iteración entrega una versión funcional y mejorada del producto.

2. **Repetición y Mejora Continua**:
   - Se pueden realizar ajustes y correcciones después de cada iteración basándose en retroalimentación.
3. **Entrega Parcial y Progresiva**:
   - Las primeras iteraciones desarrollan las funcionalidades básicas.
   - Las siguientes iteraciones agregan características más complejas.
4. **Adaptabilidad**:
   - Permite ajustes según los cambios en los requisitos del cliente o descubrimientos técnicos.

Ventajas:

- **Flexibilidad**: Los cambios pueden integrarse de manera continua.
- **Riesgos Reducidos**: Los problemas se detectan más temprano.
- **Retroalimentación Temprana**: Los usuarios pueden probar prototipos iniciales.
- **Entrega Incremental**: Funcionalidades esenciales pueden implementarse rápidamente.

Desventajas:

- Puede requerir más recursos de gestión por el desarrollo repetitivo.
- Si no se controla, las iteraciones pueden prolongarse indefinidamente.
- Requiere comunicación constante entre los equipos y el cliente.

Fases en Cada Iteración:

1. **Planificación**: Determinar objetivos específicos para la iteración.
2. **Diseño**: Crear o ajustar el diseño según los objetivos.
3. **Implementación**: Codificar y desarrollar funcionalidades.
4. **Pruebas y Evaluación**: Verificar que el producto cumple con los requisitos establecidos para la iteración.
5. **Retroalimentación y Ajustes**: Incorporar comentarios de usuarios y mejorar el producto.

- **Modelo Ágil**

La metodología ágil es un concepto de gestión de proyectos que implica dividir el proyecto en fases y hace hincapié en la colaboración y la mejora continuas. Los equipos siguen un ciclo de planificación, ejecución y evaluación.

✓ **Fases de la Metodología Ágil:**

Aunque las metodologías ágiles no tienen fases rígidas, un proyecto ágil suele seguir este flujo:

1. **Concepto y Planeación**:
   - Se identifican los objetivos del proyecto y las características principales.
   - Se priorizan las funcionalidades en un **backlog** (lista de tareas por hacer).
2. **Diseño**:
   - Se crean diseños básicos y ajustables, sin documentación extensa.
   - Se asegura que el diseño sea suficiente para soportar la implementación.
3. **Implementación**:

- El equipo desarrolla el producto en ciclos cortos (sprints).
- Cada sprint entrega un incremento funcional del software.

4. **Pruebas**:
   - Las funcionalidades se prueban de forma continua durante y después de cada sprint.
   - Los usuarios participan para validar los resultados.
5. **Entrega**:
   - Se entrega al cliente un producto funcional al final de cada iteración.
   - El software es desplegado y puede usarse en producción.
6. **Retroalimentación**:
   - Se recopilan comentarios de los clientes y usuarios para ajustar el desarrollo en iteraciones futuras.

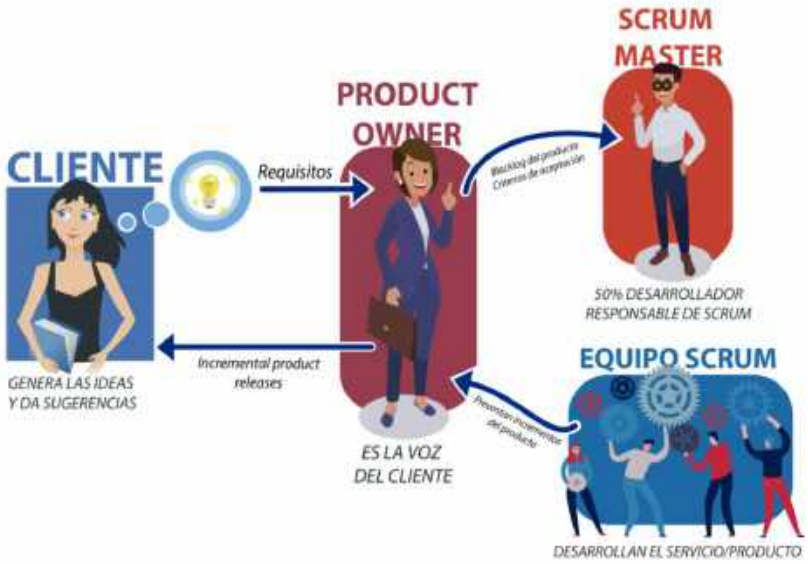
✓ **Roles en Metodologías Ágiles:**

Los roles clave varían ligeramente según la metodología específica (Scrum, Kanban, etc.), pero incluyen:

1. **Product Owner**:
   - Representa al cliente y define las prioridades.
   - Es responsable de gestionar el backlog del producto.
2. **Scrum Master** (en Scrum):
   - Facilita el proceso ágil y elimina obstáculos que afecten al equipo.
   - Garantiza que el equipo siga los principios ágiles.
3. **Equipo de Desarrollo**:
   - Grupo multidisciplinario que incluye desarrolladores, diseñadores, testers, entre otros.
   - Son responsables de entregar incrementos funcionales del producto.
4. **Stakeholders**:
   - Clientes, usuarios o partes interesadas que proporcionan requisitos y retroalimentación.
5. **Usuario Final**:
   - La persona que utilizará el producto y cuya satisfacción define el éxito del proyecto.

✓ Ventajas de la Metodología Ágil:

- **Flexibilidad** para adaptarse a cambios.
- Mayor **colaboración** entre equipos y partes interesadas.
- **Entrega continua** de valor.
- Reducción de riesgos gracias al enfoque en iteraciones pequeñas.



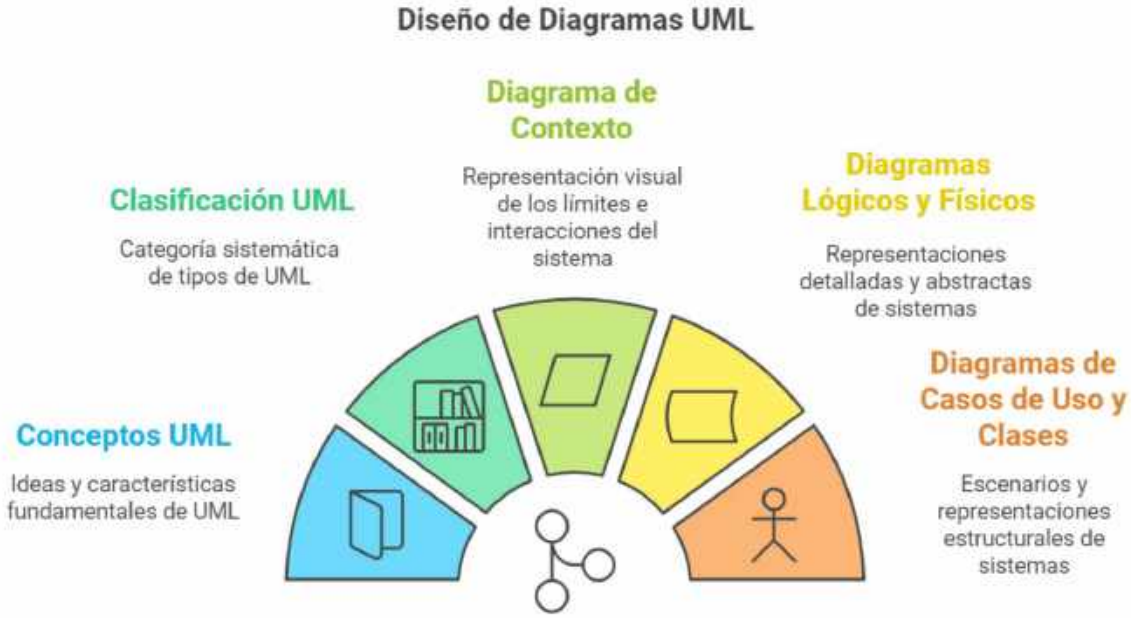
UNIDAD 2: DISEÑO DE DIAGRAMA DE UML

2.1 Conceptos y características de UML
2.2 Clasificación del UML
2.3 Diagrama de con texto
2.4 Diagrama cero
2.5 Diagrama lógico y físico
2.6 Ejemplos de Diagrama lógico y físico
2.7 FLISOL
2.8 Ejemplo de Diagrama de caso de uso
2.9 Diagrama de clase
2.10 Ejemplos con Diagrama de clase
2.11 Diagrama de actividad
2.12 Ejemplos de Diagrama de actividades.
2.13 Modelo Conceptual (entidades, atributos)
2.14 Entidades
2.15 Atributos

Resultado de Aprendizaje

El estudiante adquieran habilidades para conceptualizar, representar y comunicar de manera efectiva las estructuras y comportamientos de sistemas utilizando los diagramas estándar definidos en UML (Lenguaje de Modelado Unificado).

DIAGRAMA DE APRENDIZAJE



SINTESIS

El **diseño de diagramas UML (Unified Modeling Language)** es una herramienta esencial en el análisis y diseño de sistemas de software. UML proporciona un conjunto de diagramas estandarizados que permiten modelar y visualizar tanto la estructura como el comportamiento de un sistema. Su uso facilita la comunicación entre los miembros del equipo de desarrollo y con los clientes, asegurando que todos comprendan la arquitectura del sistema de manera clara.

UNIDAD 2: DISEÑO DE DIAGRAMA DE UML

2.1 Conceptos y características de UML

UML (Unified Modeling Language) es un lenguaje estándar de modelado visual utilizado para describir, especificar, diseñar y documentar sistemas de software. UML ayuda a representar tanto la estructura estática como el comportamiento dinámico del sistema.

Características clave de UML:

- **Visualización**: Ofrece una representación gráfica de los sistemas.
- **Estándar**: Está reconocido internacionalmente como un estándar de modelado.
- **Versatilidad**: Se utiliza en diversos tipos de sistemas y arquitecturas.
- **Comprensibilidad**: Facilita la comunicación entre los miembros del equipo y el cliente.

2.2 Clasificación del UML

UML se clasifica en dos tipos principales de diagramas:

- ✓ **Diagramas Estructurales**: Definen la estructura estática del sistema (ej. Diagrama de clases, de objetos, de componentes).
- ✓ **Diagramas de Comportamiento**: Describen las interacciones y el comportamiento dinámico (ej. Diagrama de casos de uso, de actividades, de secuencia).

2.3 Diagrama de Con Texto

Un **diagrama de con texto** es un tipo de diagrama que incluye tanto una representación gráfica como una descripción textual para explicar el funcionamiento o las relaciones entre los componentes de un sistema.

Características:

- **Claridad**: El texto ayuda a explicar detalles que no se pueden mostrar gráficamente.
- **Contextualización**: Proporciona el contexto de las relaciones o procesos que se ilustran.

2.4 Diagrama Cero

El **Diagrama Cero** es una representación preliminar de un sistema, creada en las primeras fases del desarrollo. Generalmente, es un diagrama simple que esboza las entidades principales y sus relaciones, sin entrar en detalles de implementación.

Propósito:

- Ayudar a visualizar la estructura básica.
- Establecer un punto de partida para el desarrollo de diagramas más complejos.

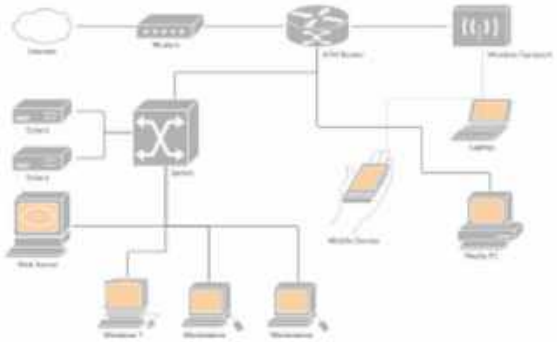
2.5 Diagrama Lógico y Físico

• Diagrama **Lógico**: Representa la estructura del sistema desde un punto de vista abstracto y conceptual, sin considerar detalles de hardware o software específicos.

- Ejemplo: Relación entre clases y objetos en un sistema.

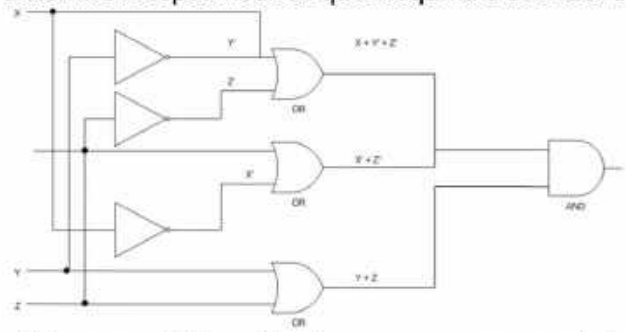
• Diagrama **Físico**: Muestra cómo se implementa el sistema en términos de hardware, bases de datos y redes.

- Ejemplo: Diagrama de despliegue que describe los nodos físicos de una red y sus conexiones.

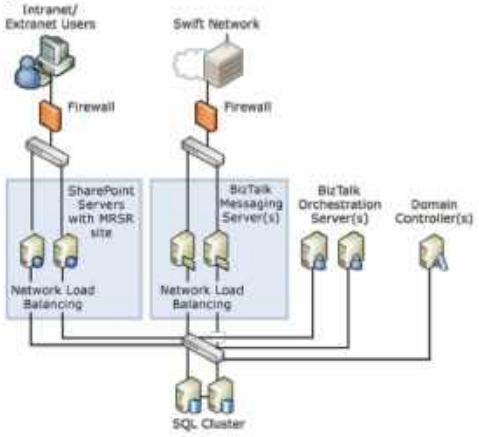


2.6 Ejemplos de Diagrama Lógico y Físico

• **Diagrama Lógico**: Un diagrama que muestra las relaciones entre los módulos de un sistema sin especificar en qué máquina o servidor se ejecuta cada módulo.



• **Diagrama Físico**: Un diagrama que representa la distribución de servidores, bases de datos y otros recursos físicos en la infraestructura de TI.

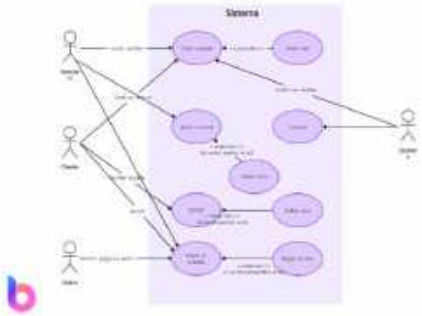


2.7 FLISOL

FLISOL (Festival Latinoamericano de Instalación de Software Libre) es un evento anual donde se promueve el uso de software libre en la comunidad. Se realizan instalaciones y charlas sobre sistemas basados en software libre, incluyendo herramientas de modelado como UML.

2.8 Ejemplo de Diagrama de Caso de Uso

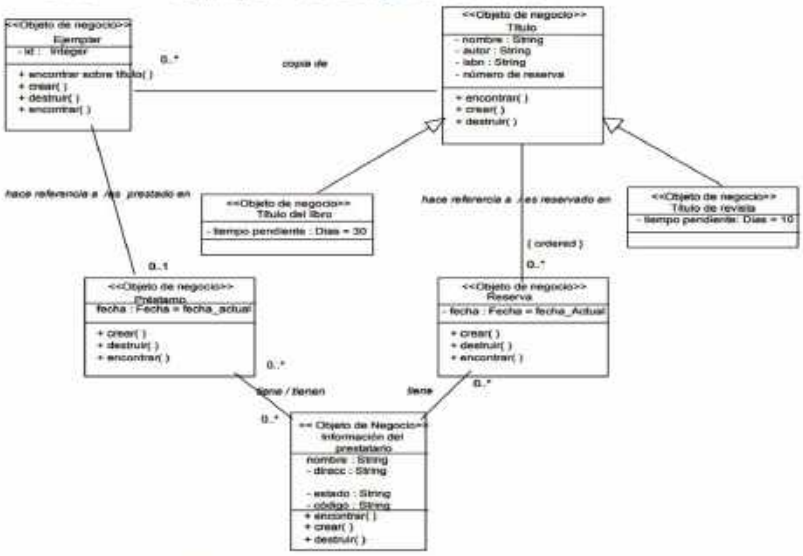
Un **diagrama de caso de uso** describe las interacciones entre los actores (usuarios o sistemas externos) y el sistema que se está desarrollando



2.9 Diagrama de Clase

El **diagrama de clase** describe las clases del sistema, sus atributos, métodos y las relaciones entre ellas (como asociaciones, herencia y dependencia). Es una de las representaciones más comunes en UML.

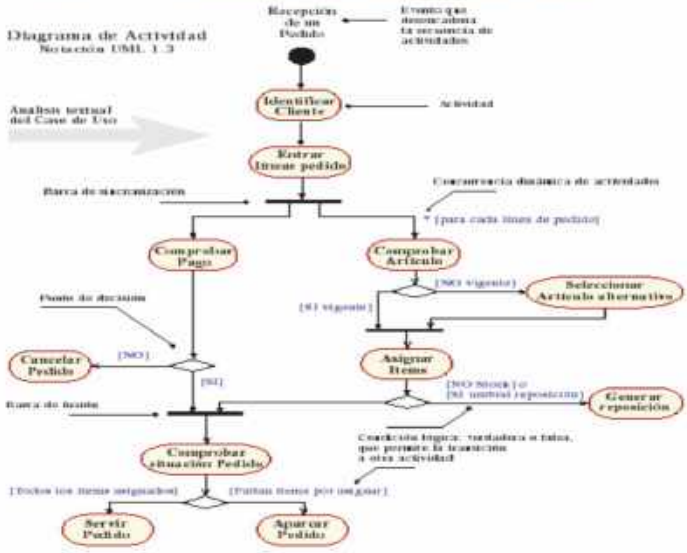
2.10 Ejemplos con Diagrama de Clase



2.11 Diagrama de Actividad

- ✓ El diagrama de actividades es una clase especial del diagrama de estados y muestra el flujo desde una actividad a otra dentro del sistema y sirven para modelar las funciones del mismo.
- ✓ Un diagrama de actividades es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes.

2.12 Ejemplos de Diagrama de Actividades

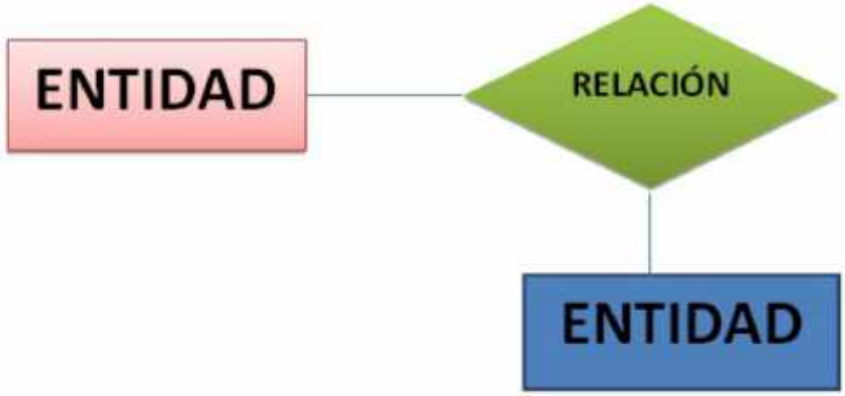


2.13 Modelo Conceptual (Entidades, Atributos)

El **modelo conceptual** es una representación abstracta de los elementos clave del sistema, como entidades y atributos. Estas entidades representan objetos o conceptos que son importantes para el dominio del problema, mientras que los atributos describen las características de esas entidades.

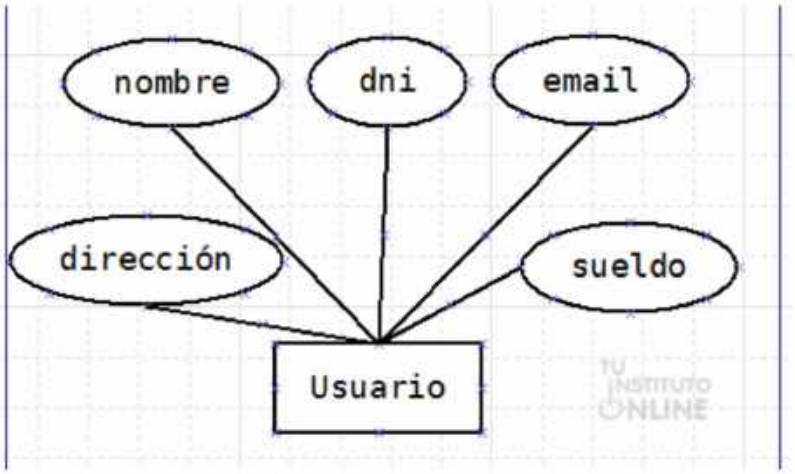
2.14 Entidades

Una entidad es una cosa u objeto del mundo real, también puede ser un concepto abstracto y es distinguible de todos los demás objetos. Una entidad tiene un conjunto de propiedades o atributos que la caracterizan. Ejemplos: Personas, Animales, Casas, Autos, etc.



2.15 Atributos

Los atributos son las características o propiedades de una entidad. Cada uno de los elementos de la entidad poseen los mismos atributos y a cada atributo se le asigna un valor único por cada elemento. Tomando la entidad "Persona" como ejemplo, identificamos algunas propiedades en ella como son: identificación, nombres, apellidos, fecha de nacimiento, sexo, etc.



Unidad 3: EL PROCESO DE DISEÑO DE UN SISTEMA INFORMÁTICO

3.1 Estándares de diseño de interfaces
3.2 Diseño de procedimientos precisos para la entrada de datos
3.3 Diseño de Interfaz
3.4 Diseño de salida
3.5 Diseño de Diccionario de datos

Resultado de Aprendizaje

Se enfoca en que los estudiantes comprendan, apliquen y analicen las distintas fases y metodologías utilizadas en el diseño de un sistema informático. El objetivo es desarrollar habilidades que permitan a los estudiantes identificar los requerimientos, planificar el diseño y realizar un sistema eficiente, funcional y alineado con las necesidades del cliente.

DIAGRAMA DE APRENDIZAJE

SINTESIS

El **proceso de diseño de un sistema informático** es un conjunto de actividades y fases que transforman los requisitos y necesidades de un sistema en una solución funcional y eficiente. Este proceso abarca desde

la planificación inicial hasta la implementación, y se basa en varios enfoques metodológicos que guían a los desarrolladores a lo largo de su evolución.

Unidad 3: EL PROCESO DE DISEÑO DE UN SISTEMA INFORMÁTICO

3.1 Estándares de Diseño de Interfaces

Los estándares de diseño de interfaces son directrices que aseguran que las interfaces de usuario sean consistentes, accesibles, y fáciles de usar. Estos estándares buscan crear experiencias coherentes y agradables para los usuarios.

3.2.1 Puntos clave:

- Consistencia: Utilización de patrones comunes en el diseño de interfaz para que los usuarios no tengan que aprender a usar el sistema desde cero cada vez que interactúan con una nueva pantalla.
- Accesibilidad: La interfaz debe ser accesible para personas con discapacidades, lo que puede incluir la compatibilidad con lectores de pantalla o el uso de colores y contrastes apropiados.
- Claridad: Las interfaces deben ser claras y concisas, facilitando que el usuario entienda rápidamente lo que debe hacer.
- Responsividad: La interfaz debe adaptarse a diferentes tamaños de pantalla y dispositivos, garantizando que los usuarios tengan una experiencia fluida sin importar el dispositivo.

3.2 Diseño de Procedimientos Precisos para la Entrada de Datos

El diseño de procedimientos precisos para la entrada de datos implica la creación de formularios y campos de entrada que faciliten la captura correcta y eficiente de la información por parte de los usuarios.

3.2.1 Puntos clave:

- ✓ **Validación de datos**: Asegurarse de que los datos ingresados sean correctos y consistentes (por ejemplo, validar que un correo electrónico tenga el formato correcto).
- ✓ **Facilidad de uso**: Los formularios deben ser fáciles de completar, con campos bien etiquetados y agrupados lógicamente.
- ✓ **Sugerencias de entrada**: Utilizar sugerencias o autocompletar para facilitar la entrada de datos repetitivos o largos (como direcciones o nombres).
- ✓ **Manejo de errores**: Incluir mensajes claros de error cuando los datos ingresados no sean correctos, indicando cómo solucionarlo.

3.3 Diseño de la Interfaz

El diseño de la interfaz se refiere a la creación visual y funcional de las pantallas que interactúan con el usuario. Un buen diseño de interfaz facilita la navegación y el uso del sistema.

3.3.1 **Puntos clave**:

- ✓ **Diseño visual atractivo**: La interfaz debe ser agradable y visualmente coherente, utilizando tipografías, colores y elementos gráficos que sean fáciles de leer y atractivos.
- ✓ **Navegación intuitiva**: Los usuarios deben poder navegar por la aplicación sin dificultad. Los menús deben ser claros y los botones deben estar colocados en lugares lógicos.
- ✓ **Interacción efectiva**: La interfaz debe permitir una interacción fluida y eficiente, sin tiempos de espera innecesarios o pasos complicados.

3.4 Diseño de Salida

El diseño de salida se refiere a cómo se presentan los resultados o la información al usuario. Un buen diseño de salida es crucial para la comprensión rápida de los datos procesados.

3.4.1 Puntos Clves:

- ✓ **Claridad y simplicidad**: La información debe ser presentada de forma clara y fácil de entender. Los resultados deben destacarse para que el usuario pueda interpretarlos rápidamente.
- ✓ **Formato adecuado**: La salida debe ser presentada en el formato que mejor se adapte al tipo de información, como texto, tablas, gráficos, etc.
- ✓ **Retroalimentación al usuario**: Es importante proporcionar mensajes de retroalimentación para confirmar acciones o mostrar el estado de un proceso (por ejemplo, "Compra completada exitosamente").

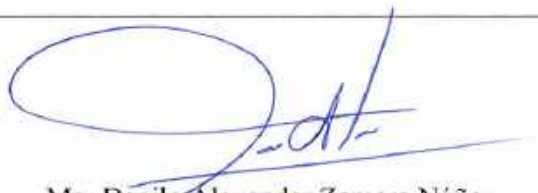
3.5 Diseño de Diccionario de Datos

El diccionario de datos es una herramienta utilizada para definir todos los elementos de datos dentro de un sistema, describiendo su estructura, tipo, posibles valores y cómo interactúan dentro del sistema.

3.5.1 Puntos clave:

- ✓ **Definición clara**: Cada elemento de datos debe tener una descripción precisa para evitar ambigüedades.
- ✓ **Estandarización**: Usar un formato estándar para describir los datos, asegurando que todas las partes interesadas comprendan su significado y uso.
- ✓ **Relaciones de datos**: Definir las relaciones entre diferentes entidades de datos dentro del sistema (por ejemplo, una relación entre "cliente" y "pedido" en un sistema de ventas).

.

ELABORACIÓN, REVISIÓN Y APROBACIÓN DE PARES	
Profesor(a)	
 Ing. Darwin Fernando Núñez C. Mg.	
Fecha de elaboración: 01/8/2023	
Comisión de revisión de pares de guías de estudio del Instituto Superior Tecnológico Tena	
 Lcdo. Segundo Calisto Rochina Chileno	 Mg. Alvaro Santiago Toalombo Díaz
 Mg. Henry Fabian Chango Chango	 Ing. Agustin Gonzalo Guanipatin Ramirez
Fecha de revisión: 04/09/2023	
Coordinador de Investigación, Desarrollo Tecnológico e Innovación	
 Mg. Danilo Alexander Zamora Núñez	
Fecha de aprobación: 12/10/2023	