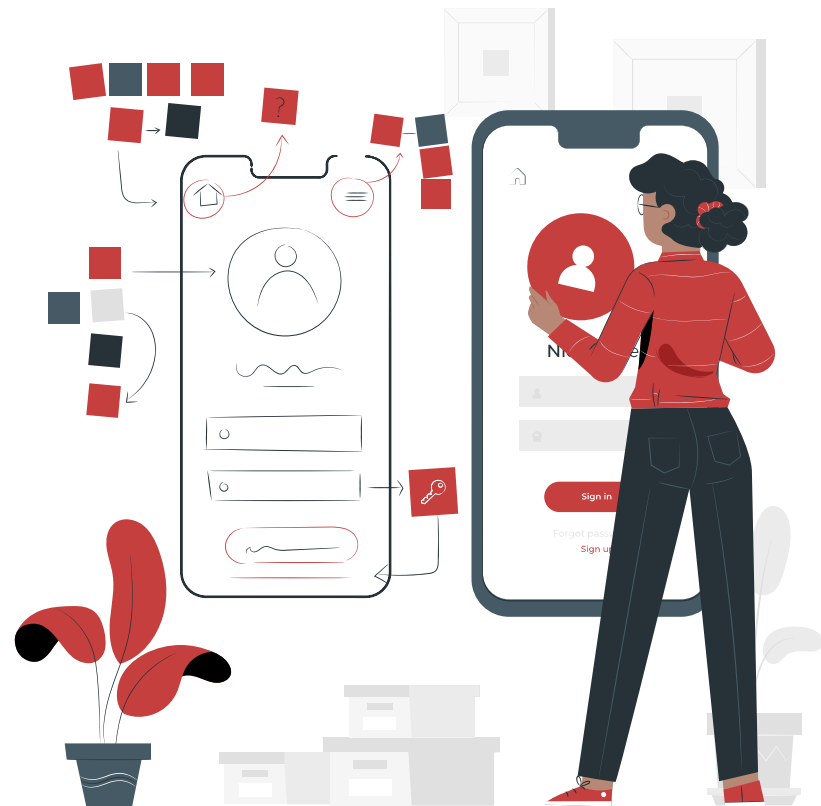


EL PROCESO DEL DESARROLLO DE SOFTWARE

Desarrollo de aplicaciones web en el
entorno servidor



CONTENIDO

01

Modelos del ciclo de vida del software

Análisis y especificación de requisitos

02

Diseño

Implementación

03

Validación y verificación de sistemas

Pruebas y calidad del software

04

Herramientas para el desarrollo de Software

Proyectos de desarrollo de software



01

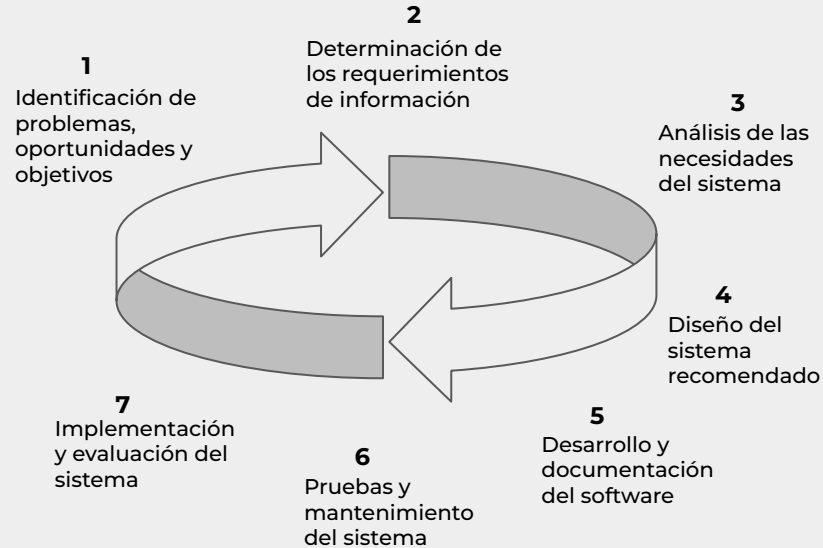
Ciclo de vida del software

Modelos

¿Qué es?

“El ciclo de vida de vida del desarrollo de sistemas es un **enfoque** por fases para el **análisis y el diseño** cuya premisa principal consiste en que los sistemas se desarrollan mejor utilizando un **ciclo específico de actividades del analista y el usuario**”

Kendall & Kendall



Siete fases del ciclo de vida del desarrollo de sistemas

IDENTIFICACIÓN DE PROBLEMAS Y OBJETIVOS

- Observación directa
- Síntesis y organización de la información
- Identificar si existe una necesidad, un problema o una oportunidad argumentada
- Documentar resultados
- Estimar el alcance del proyecto y estudiar los riesgos
- Determinación de viabilidad



DETERMINACIÓN DE LOS REQUERIMIENTOS



¿La Meta?

Determinación de los objetivos General y específicos



¿Cómo?

Descripción detallada de los procedimientos



¿Quién y qué?

Personas involucradas, actividades desempeñadas, entorno, funciones, procedimientos actuales ...



¿Qué debe hacer?

Requerimientos funcionales y no funcionales



Herramientas

Entrevistas, muestreos, investigación de datos impresos y aplicación de cuestionarios

ANÁLISIS DE LAS NECESIDADES DEL SISTEMA





DISEÑO DEL SISTEMA RECOMENDADO

Diseño lógico del sistema

Procedimientos de captura de datos

Bases de datos (E-R)

Interfaz de usuario

Controles y procedimientos de respaldo

Especificaciones de programa para los programadores

DESARROLLO Y DOCUMENTACIÓN DEL SOFTWARE



PRUEBAS Y MANTENIMIENTO



Casos Borde
Rectificaciones
necesarias



Programación de
las pruebas del
sistema



Planificación de las
horas del
mantenimiento del
sistema



IMPLEMENTACIÓN Y EVALUACIÓN



Resistencia al cambio

Medir la adaptabilidad de los usuarios

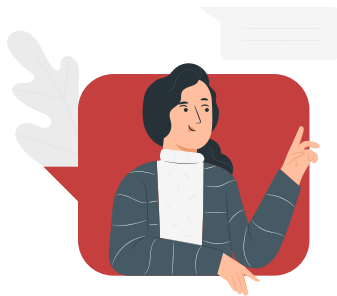
Instalación

Instalación y puesta en marcha

Adiestramiento

Enseñar al usuario el correcto uso del software

MODELOS DEL CICLO DE VIDA DEL SOFTWARE



El Modelo

Intenta determinar el orden de las etapas involucradas en el desarrollo de software

En cascada (Waterfall)

No muy complejos, sin entregas parciales. Más utilizado

Espiral

Es más costoso

Iterativo y creciente

Favorece la creación de prototipos

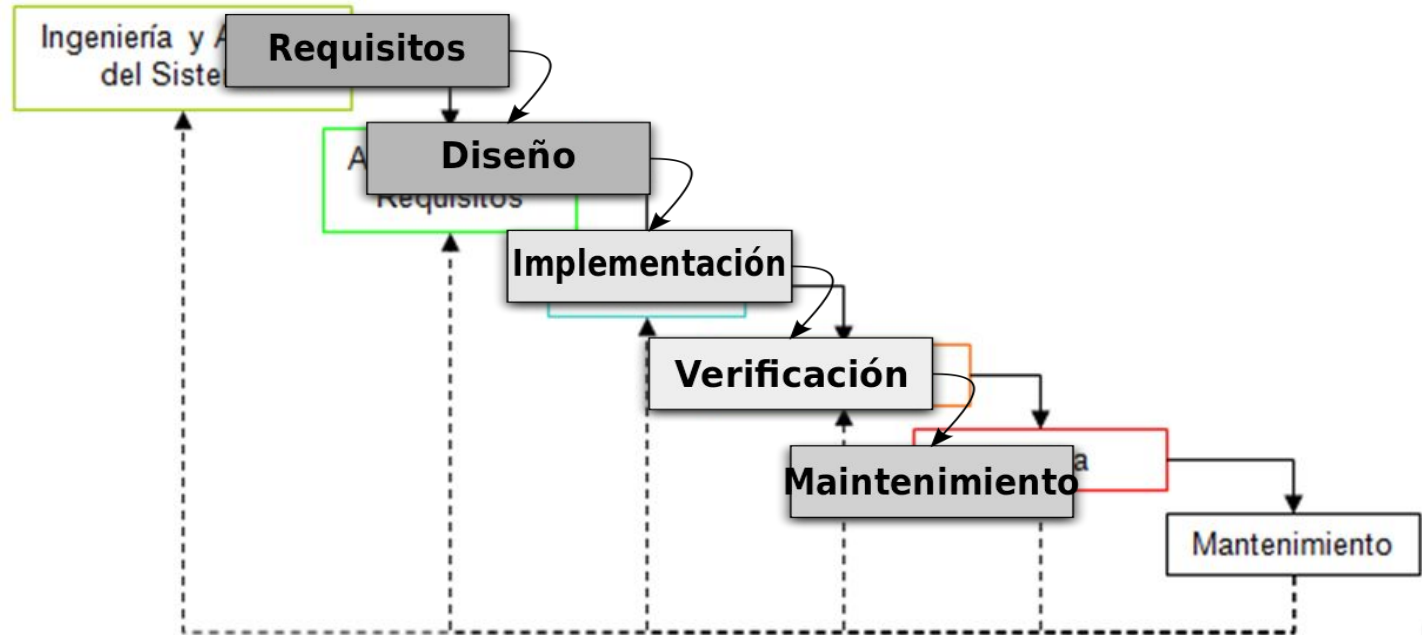
En V

Desarrollos que requieran gran robustez y confiabilidad

Desarrollo rápido (RAD)

Fusiona las fases de análisis y diseño. Desarrollos rápidos

MODELO CLÁSICO O EN CASCADA *WATERFALL*


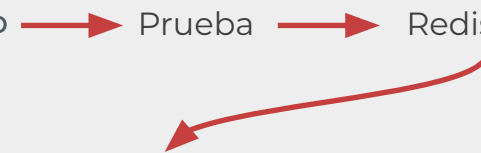



MODELO CLÁSICO O EN CASCADA

Ventajas

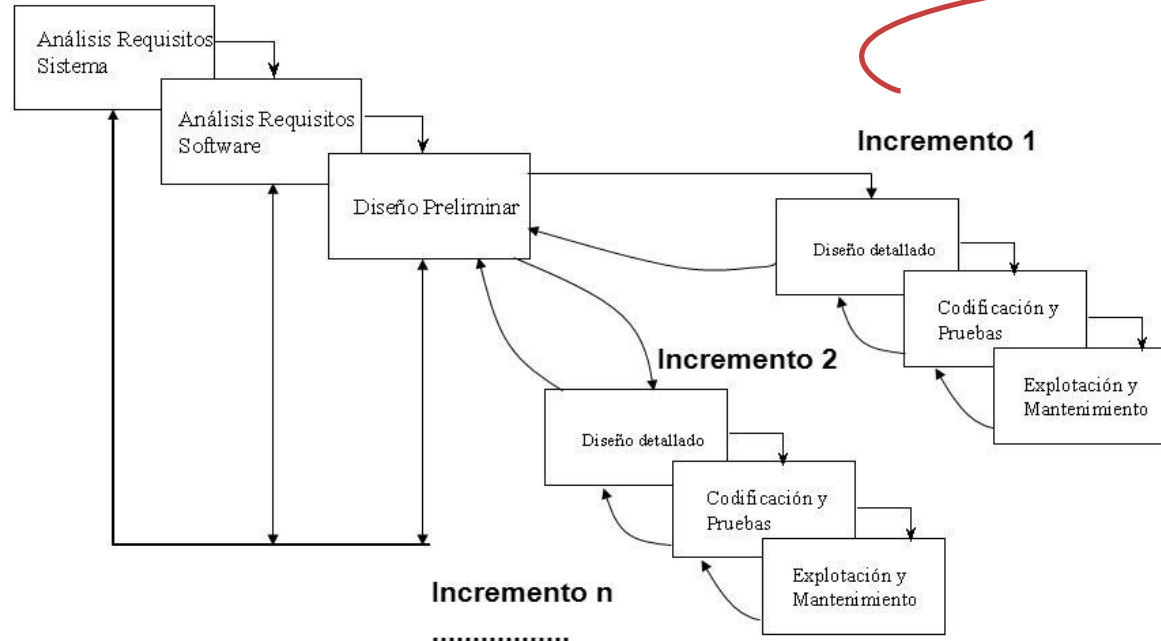
- Requiere de menos capital y herramientas para hacerlo funcionar de manera óptima
- Fácil de implementar y entender
- Está orientado a documentos
- Promueve una metodología de trabajo efectiva:
 - Definir antes que diseñar, diseñar antes que codificar

Desventajas

- Secuencia lineal 
 - Mala implementación del modelo
 - Etapa x etapa
 - Creación del software tarda mucho tiempo
 - Errores de diseño 
 - Prueba
 - Rediseño
-  Costos del desarrollo

ITERATIVO E INCREMENTAL

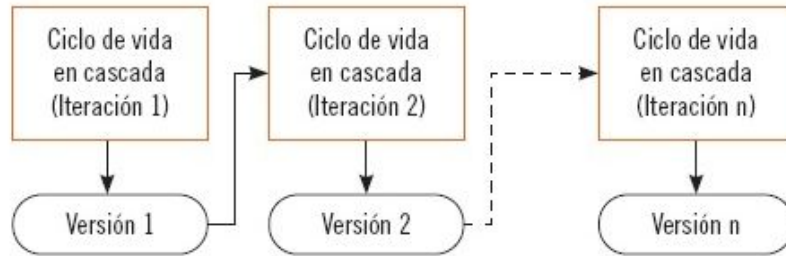
Proceso de desarrollo de software es creado en respuesta a las debilidades del modelo tradicional de cascada



Tareas agrupadas en pequeñas etapas repetitivas

ITERATIVO E INCREMENTAL

Modelo iterativo



Desventajas

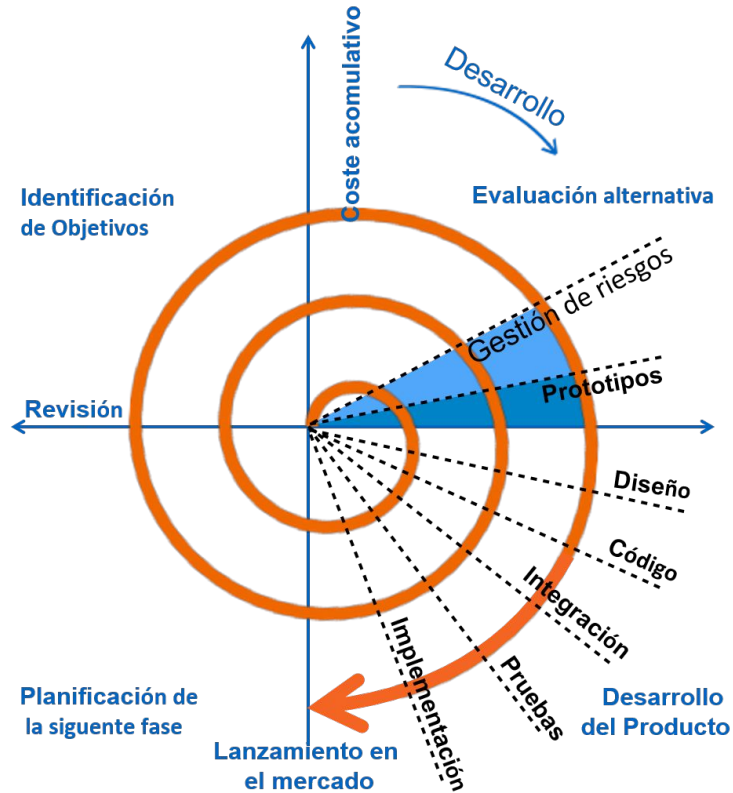
La entrega temprana produce sistemas demasiados simples

Los incrementos se hacen en base a las necesidades del usuario

Requiere de un cliente involucrado

La entrega de un programa que es parcial pero funcional puede hacer vulnerable al programa

MODELO DE ESPIRAL



Se comienza fijando los objetivos y las limitaciones al empezar cada repetición. En la etapa siguiente se crean los modelos de prototipo del software, que incluye el análisis de riesgo.

Posteriormente se usa un modelo estándar para construir el software y finalmente se prepara el plan de la próxima repetición.

MODELO DE ESPIRAL

Ventajas

El análisis del riesgo se hace de forma explícita y clara

Reduce riesgos del proyecto

Incorpora objetivos de calidad

Integra el desarrollo con el mantenimiento

Nuevos requerimientos sin romper con la metodología

Desventajas

Genera mucho tiempo en el desarrollo del sistema

Modelo costoso

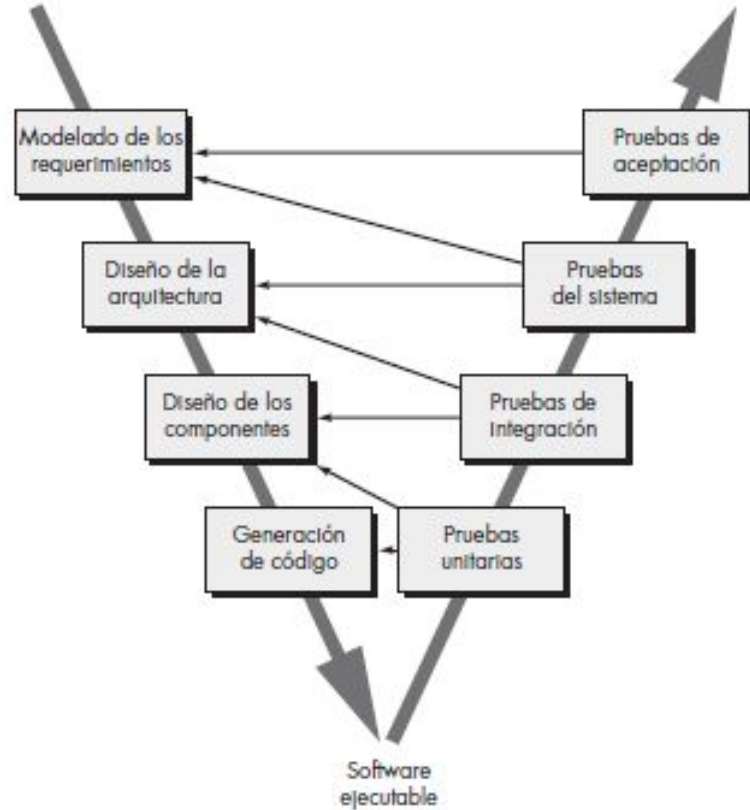
Requiere experiencia en la identificación de riesgos

MODELO EN V

El modelo en V ofrece más opciones de evaluación del software en cada etapa

En cada fase se crea la planificación de las pruebas y los casos de pruebas para verificar y validar el producto en función de los requisitos de la misma.

Las pruebas se realizan lo más pronto posible. De esta manera, verificación y validación van en paralelo



DESARROLLO RÁPIDO (RAD)



Intenta resolver problemas

- Largos tiempos de desarrollo
- Problemas de documentación en papel
- Elaborar de manera rápida parte del sistema
- Facilitar el entendimiento del sistema e identificación de requerimientos.

- **Desarrollo por fases**

Sistema desarrollado en múltiples versiones secuenciales

Se van agregando requerimientos

- **Prototipos**

Realiza el análisis, diseño e implementación de manera simultánea desarrollando prototipos

DESARROLLO RÁPIDO (RAD)

Hace uso de

Herramientas CASE (Computer Aided Software Engineering)

Lenguaje Visual que simplifica y acelera la programación

Generadores de código a partir de especificaciones de diseño

Ventajas

Usuarios interactúan pronto con el sistema

Permite refinar verdaderos requerimientos

Desventajas

Falta visión global

Nuevos requerimientos pueden conducir a cambios mayores en el prototipo

MODELO DEL CICLO DE VIDA ADECUADO

En la elección del modelo del ciclo de vida adecuado, hay que considerar cinco factores básicos

1. Tiempo hasta la entrega final
2. Complejidad del problema
3. Necesidad (o no) de entregas parciales
4. Definición y exactitud de los requerimientos
5. Recursos disponibles



PROCESO PARA EL DESARROLLO DE SOFTWARE

Procesos, actividades y tareas involucradas en el desarrollo, implementación y mantenimiento del software

Ciclo de vida del desarrollo de software



Estándar internacional ISO 12207.



Actividades

- Planificación
- Análisis y Diseño
- Implementación
- Pruebas
- Instalación o despliegue
- Uso y mantenimiento



ACTIVIDAD

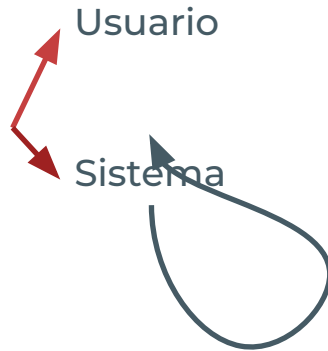
ANÁLISIS Y ESPECIFICACIÓN DE REQUISITOS



La especificación de requisitos de software (ERS)

Es una descripción completa del comportamiento del sistema que se va a desarrollar

- Funciones
- Datos
- Información
- Interacciones



Calidad
ISO
9126

A thick, dark grey curved arrow pointing from the text 'Calidad' down to 'ISO 9126'.

Análisis y especificación de requisitos

Una buena ERS debe ser:

- Completa
- Consistente
- Inequívoca
- Correcta
- Trazable

- Priorizable
- Modificable
- Verificable

Jerárquica

- Esenciales
- Condicionales
- Opcionales

- Historia
- Ubicación
- Aplicación

IEEE 830-1998.



Restricciones que afectan al sistema.

Ej: debe cumplir con lo dispuesto en la Ley Orgánica de Protección de Datos

TIPOS DE REQUISITOS

Usuarios

Sistema

Funcionales

No funcionales

MODELOS PARA EL ANÁLISIS DE REQUISITOS

Definir una serie de requisitos que puedan validarse

Establecer una base para el diseño

Describir lo que quiere el cliente



Entrevistas

Análisis de documentos

Tormenta de ideas



Construir un modelo, identificar los problemas, crear el sistema propuesto



TÉCNICAS DE DESARROLLO DEL SOFTWARE



Recopilación de datos

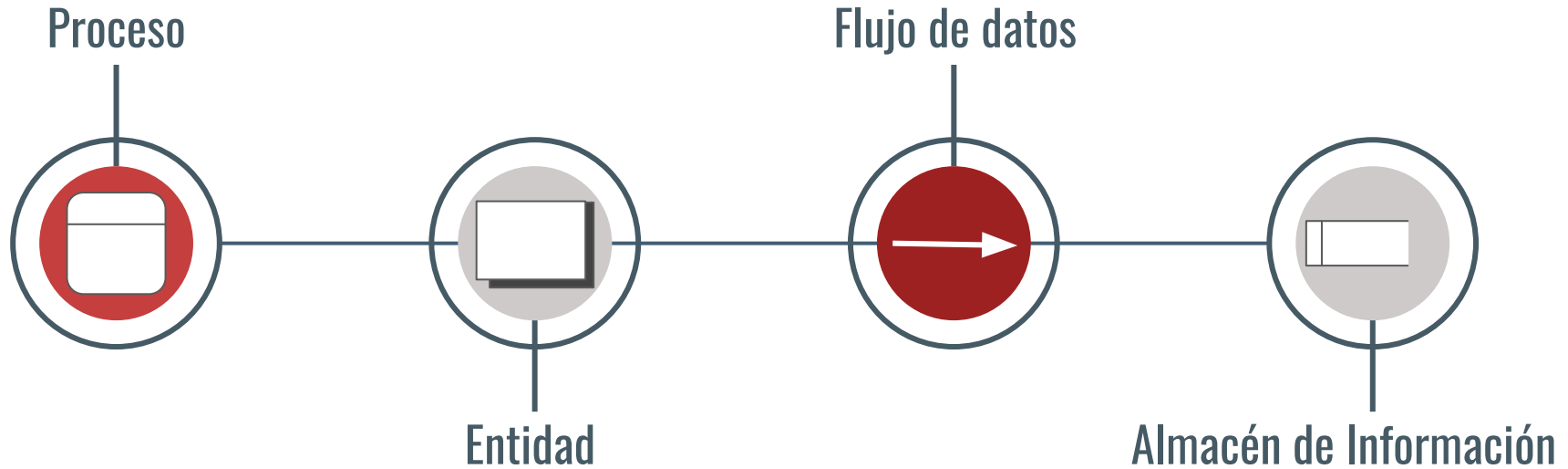
Análisis de costo
beneficio

Planificación y control de
proyectos



ACTIVIDAD

SÍMBOLOS PARA ELABORAR DFD'S



APLICACIÓN PRÁCTICA DE DFD

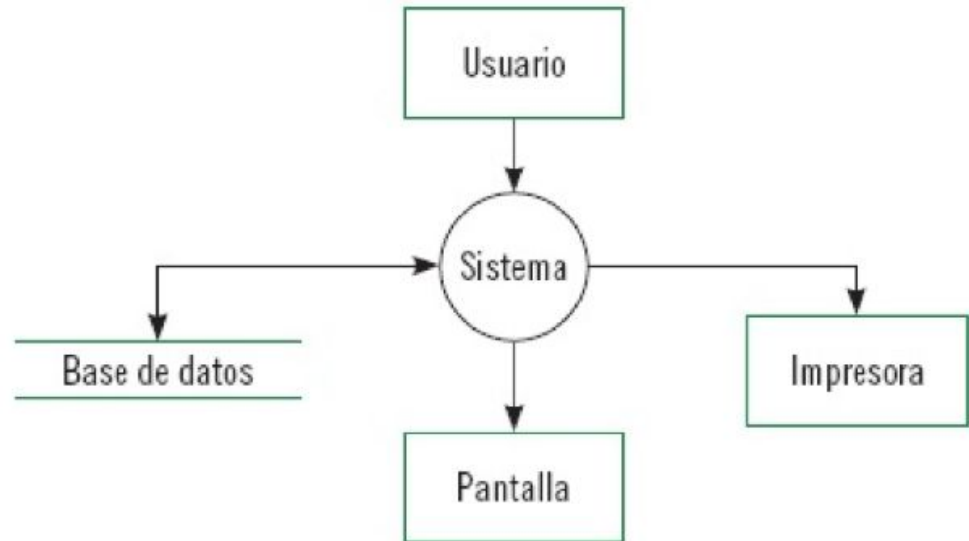


Supuesto

En un trabajo le proponen realizar un diagrama de flujo de nivel 0.

Se le informa que las entidades externas son Impresora, Usuario y Pantalla, mientras que existe un almacén externo que es Base de Datos.

Diagrama de Flujo de Nivel 0

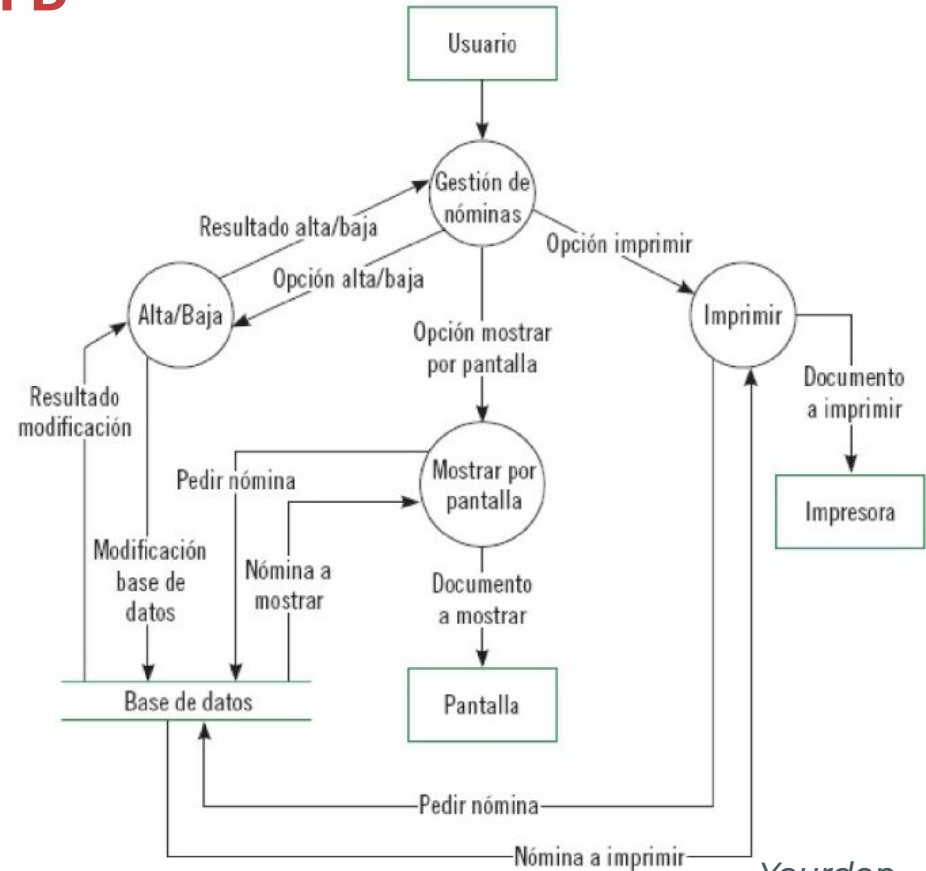


APLICACIÓN PRÁCTICA DE DFD



Realizar también el diagrama de flujo de **nivel 1** sabiendo que el sistema presenta las siguientes opciones:

- Imprimir (accede a la entidad base de datos)
- Mostrar por pantalla (accede a la base de datos y muestra información por pantalla)
- Realizar altas y bajas (accede/modifica a la base de datos)





ACTIVIDAD

DOCUMENTACIÓN

INTRODUCCIÓN



Propósito

Motivo del documento y la audiencia a la que va dirigido

Ámbito

Descripción y ámbito del software, destacando metas, objetivos y beneficios

Descripción general del documento

Contenido y estructura

DOCUMENTACIÓN

CUERPO

Descripción general del sistema

Funcionalidad, contexto y diseño del sistema

Arquitectura del sistema

- Diseño arquitectónico: estructura modular del programa
- Criterio de diseño: se justifica la elección de la arquitectura adoptada

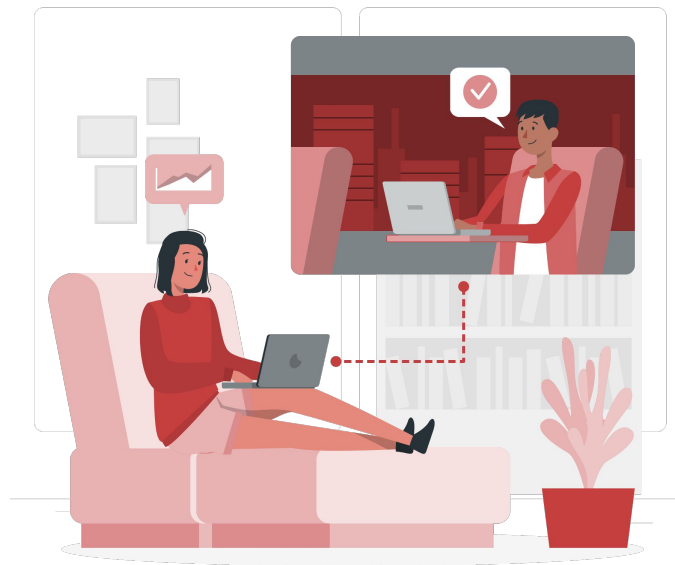
Diseño de datos

- Descripción de datos: cómo los datos van a ser representados, almacenados y procesados
- Diccionario de datos: lista de las entidades resultantes

Diseño de componentes

Diseño de interfaz

IMPLENMENTACIÓN



Principios básicos del desarrollo del software

1. Plan de desarrollo basado en el ciclo de vida
2. Validación continua
3. Control de versiones
4. Técnicas de programación modernas
5. Control de resultados
6. Calendario
7. Mejorar el proceso de desarrollo

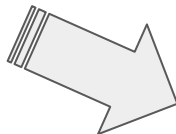
VALIDACIÓN Y VERIFICACIÓN DE SISTEMAS



Validación

Se está haciendo el producto correcto.

¿Cumple con los requisitos del cliente?



Verificación

Se está haciendo el producto correctamente.

¿Es funcionalmente correcto y robusto?



MÉTODOS DE VERIFICACIÓN Y VALIDACIÓN

Hay herramientas de software
que escanean el código de un
programa en busca de posibles
errores

DEPURACIÓN

Verificación y Validación

Establece que hay errores

Control de cambios

Corregir errores y volver a probar



Depuración

Detecta y elimina los errores

Integradas

Ejecución paso a paso

PRUEBAS DE SOFTWARE

Actividades destinadas a verificar de manera **objetiva** que se ha creado un software libre de errores y que cumple con lo exigido



60%

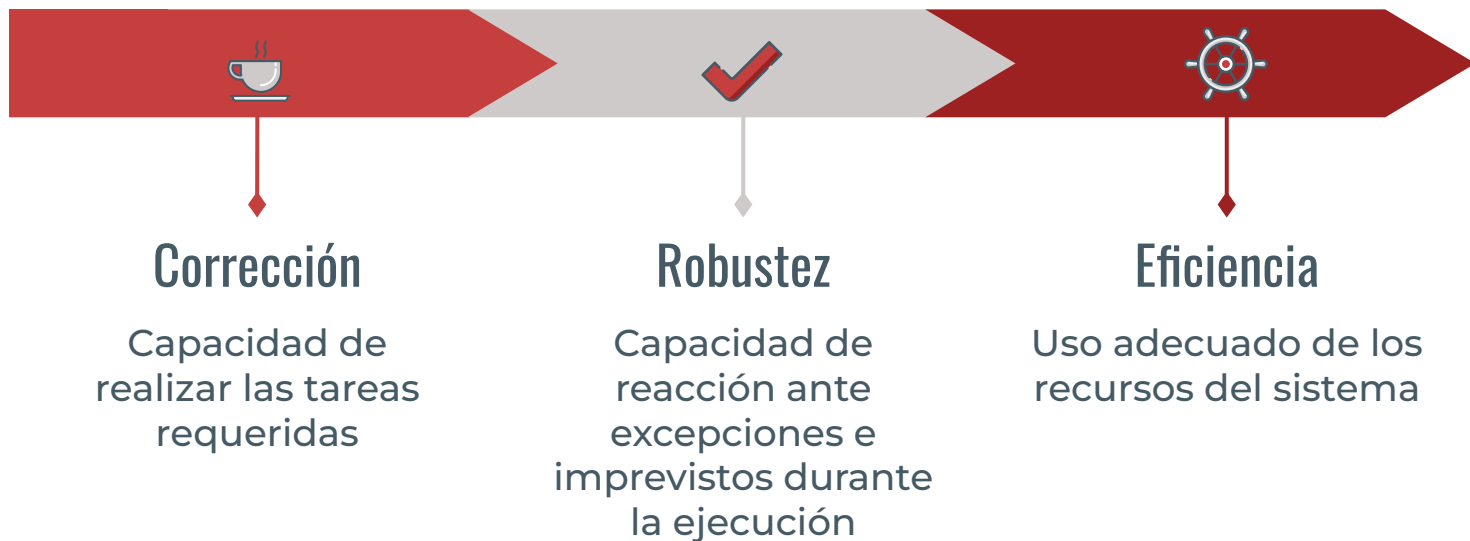


Tipos de pruebas

Según el ámbito de las pruebas:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de sistema

CALIDAD DEL SOFTWARE



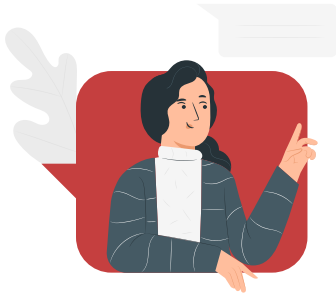
CALIDAD DEL SOFTWARE



CALIDAD DEL SOFTWARE



MÉTRICAS



¿Qué es una métrica?

“Una métrica es una **medida** que permiten evaluar la calidad del software manera objetiva.”

Características

Simples

Facilidad de cálculo

De naturaleza empírica

Objetivas

Independientes del lenguaje de programación usado

Facilidad de uso para realimentación

Principales métricas

Métricas en el modelo de análisis

Intentan predecir el tamaño del sistema



Métricas de diseño

Son utilizadas durante la fase de diseño



Métricas de codificación

Ofrecen datos sobre la complejidad del código



Métricas para pruebas

Miden el éxito de las pruebas, la cobertura de requisitos y la tasa de eliminación de errores

Estándar ISO/IEC 9126

Es un estándar internacional para la evaluación de la calidad del software. Está dividido en cuatro partes:

Modelo de calidad	Métricas externas	Métricas internas	Calidad de uso
Características de un software de calidad	Cuantifican la calidad externa del software	Miden la calidad interna del software	Miden el cumplimiento de las necesidades del usuario
Funcionalidad Fiabilidad Usabilidad Eficiencia Portabilidad	Funcionalidad Fiabilidad Usabilidad Eficiencia Portabilidad	Eficacia Seguridad Tolerancia Operabilidad ...	Efectividad Productividad Seguridad Satisfacción



HERRAMIENTAS DE USO COMÚN EN EL DESARROLLO DEL SOFTWARE

Editores de texto plano:

SublimeText 3
NotePad++
Gedit

OTRAS HERRAMIENTAS



Generadores de Programas

PHPRuner | ScriptCase

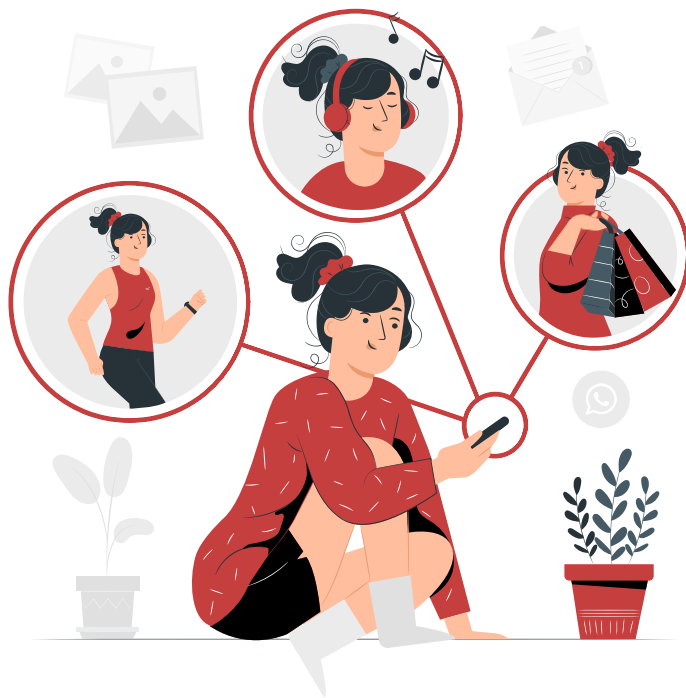
Optimizadores de código

Forma parte del compilador

Empaquetadores

Herramientas que crean un paquete instalable

OTRAS HERRAMIENTAS



Depuradores

PHPRuner | ScriptCase

De control de versiones

Git | GitHub

Entornos integrados de desarrollo

Eclipse | NetBeans | VisualStudio

OTRAS HERRAMIENTAS



Generadores de documentación

[JavaDoc](#) | [Phpref](#) | [Phpdoc](#)

Gestores y repositorios de paquetes

[GitHub](#)

Gestores de actualización de software

Automatizar la actualización del software

“Los ordenadores son buenos
siguiendo instrucciones, no leyendo
tu mente.”

—Donald Knuth

GRACIAS



¿Tienes preguntas?
bitonobit@gmail.com

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**

