

Search ...



DISEÑO DE LA CONSTRUCCIÓN



CONTENIDOS



INTRODUCCIÓN
TÉCNICAS DE DISEÑO
DISEÑO DE INTERACCIONES CON LA BD
DISEÑO DE ARCHIVOS
DISEÑO DE SALIDA
CONCLUSIONES

INTRODUCCIÓN



DISEÑO DE LA CONSTRUCCIÓN.

1 Introducción

En el diseño de software, se trata de usar técnicas y principios para definir claramente qué se quiere construir y cómo, proporcionando suficientes detalles para que pueda ser entendido y construido adecuadamente.



DISEÑO DE LA CONSTRUCCIÓN DE SOFTWARE.

Requerimientos

Comprender las necesidades y expectativas del cliente para asegurar que el software cumpla con sus requerimientos

1

Arquitectura del software

Definir una estructura sólida que organice los componentes y la comunicación entre ellos.

2

Modularidad

Crear módulos independientes y cohesivos para facilitar el desarrollo y mantenimiento del software.

3

Eficiencia y rendimiento

Optimizar el software para que funcione rápidamente y de manera eficiente.

04

Seguridad

Incorporar medidas de seguridad para proteger los datos y prevenir posibles ataques.

05

Usabilidad

Diseñar una interfaz intuitiva y amigable para que los usuarios puedan interactuar fácilmente con el software.

06

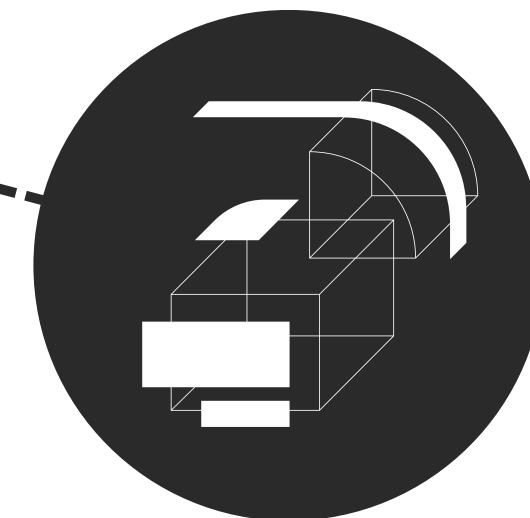


DISEÑO DE LA CONSTRUCCIÓN



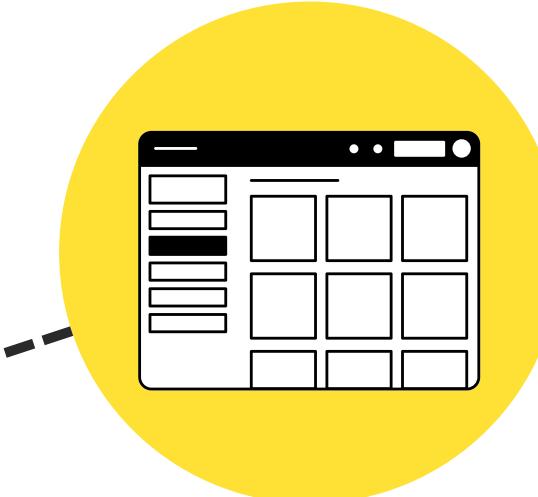
Base de Datos

Trasforma el modelo de dominio de la información, creado durante el análisis, en las estructuras de datos necesarios para implementar el Software.



Arquitectónico

Define la relación entre cada uno de los elementos estructurales del programa.



Interfaces

Describe como se comunica el Software consigo mismo, los sistemas que cooperan junto con él y usuarios que lo emplean a nivel de vista.



Procedimientos

Explica la conexión o interacción entre los distintos componentes que conforman la estructura del programa.

TÉCNICAS DE DISEÑO



TÉCNICAS DE DISEÑO

Para diseñar la construcción de un software, se debe regir a las siguientes técnicas que posibilitan al diseñador abordar todos los elementos del sistema que será construido.

- Incorporar todos los requisitos explícitos descritos en el modelo de análisis y recopilar, así también recopilar implícitos solicitados por el cliente.
- Ofrecer una visión completa del software, abarcando los dominios de datos y funcionalidades.
- Presentar estructura jerárquica entre los componentes del software.
- Interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno externo.

HERRAMIENTAS PARA EL DISEÑO DE SOFTWARE

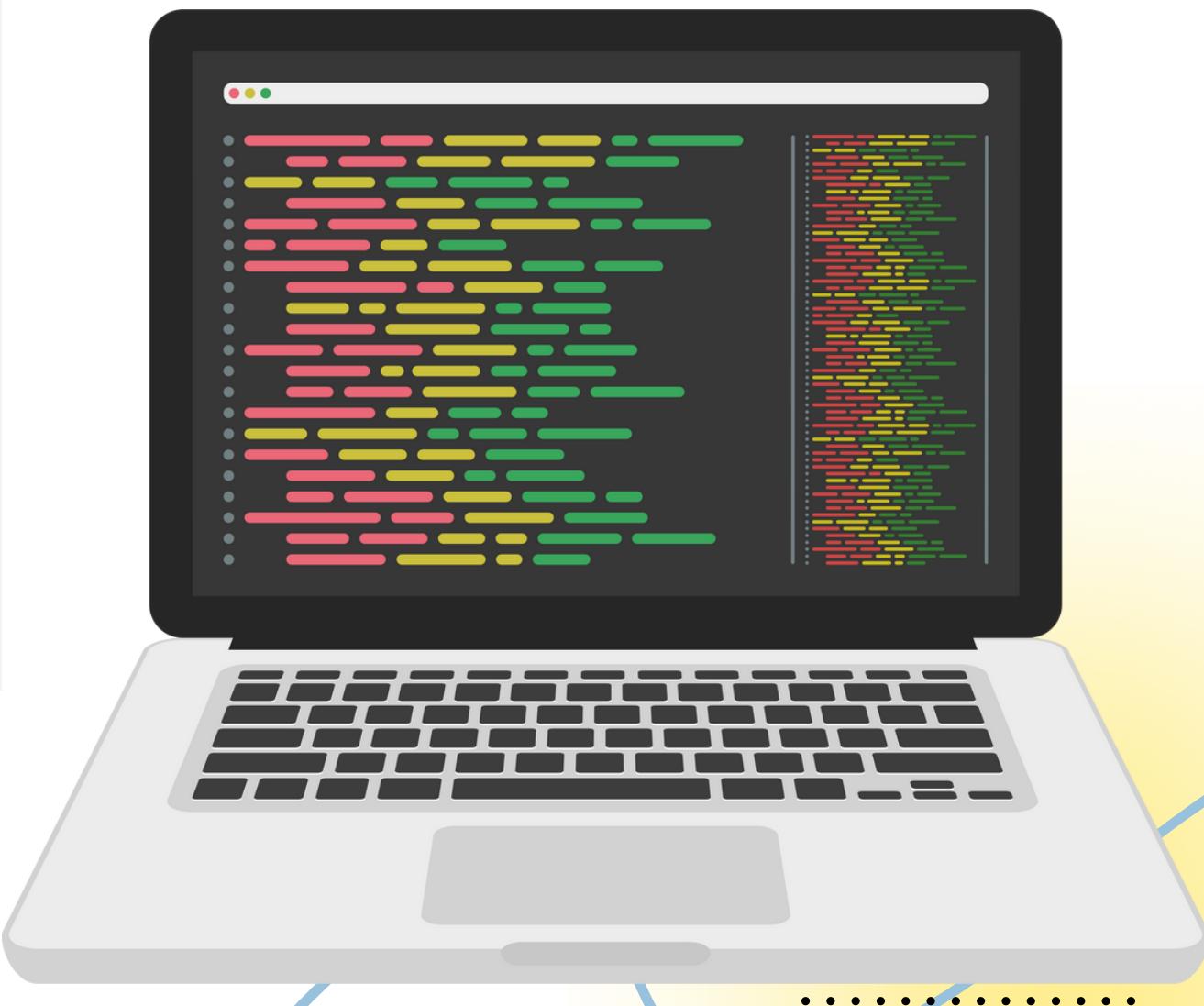
Ayudan en el proceso de definir las características que el sistema debe poseer para cumplir con los requisitos identificados durante las actividades de análisis.

1. **De especificación:** Ayudan en el proceso de definir las características necesarias de una aplicación, como entradas, salidas, procesamiento y especificaciones de control.
2. **De presentación:** Se utilizan para describir la disposición de datos, mensajes y encabezados en las pantalla.
3. **De desarrollo de sistemas:** Apoyan el proceso de formular diseños de software, incluyendo procedimientos, controles y la documentación correspondiente.
4. **De generación de código:** Producen el código fuente y las aplicaciones a partir de especificaciones funcionales bien articuladas.
5. **De pruebas:** Apoyan la etapa de evaluación de un sistema o partes de mismo en comparación con las especificaciones

DISEÑO DE INTERACCIONES BASE DE DATOS



Es una etapa importante en el desarrollo de aplicaciones que utilizan una base de datos. Implica la planificación y definición de cómo la aplicación se comunicará con la base de datos para realizar las operaciones necesarias, manteniendo una separación adecuada entre la lógica de la aplicación y los detalles específicos de la base de datos.



Acceso y privilegios

Se definen las reglas de seguridad y control de acceso a la base de datos.

Manejo de errores

Lidiar con situaciones inesperadas al interactuar con la base de datos.

Optimización de consultas

Las consultas realizadas a la base de datos deben ser eficientes y estar optimizadas.

Planificación de Operaciones

Se planifican y definen las operaciones que la aplicación realizará sobre la base de datos

MÉTODOS



Define cómo la aplicación realizará las operaciones básicas de Crear, Leer, Actualizar y Eliminar sobre los datos de la base de datos.

**Operaciones
CRUD**



Permiten recorrer los resultados de una consulta fila por fila en lugar de obtener todas las filas a la vez.

Cursos



Ayudan a mejorar la seguridad y el rendimiento, se invocan desde la aplicación para realizar tareas específicas en la base de datos.

**Procedimientos
Almacenados**



Las transacciones son fundamentales cuando se realizan operaciones que deben ser tratadas como una unidad indivisible de trabajo.

Transacciones

MÉTODOS



El ORM mapea objetos de la aplicación a tablas de la base de datos, evitando la necesidad de escribir consultas SQL directamente.

ORM (Object-Relational Mapping)



Facilita la escritura de consultas sobre colecciones de objetos de la aplicación de manera similar a las consultas SQL.

LINQ (Language Integrated Query)



Proporcionan una interfaz de programación que permite a las aplicaciones comunicarse y realizar operaciones en bases de datos

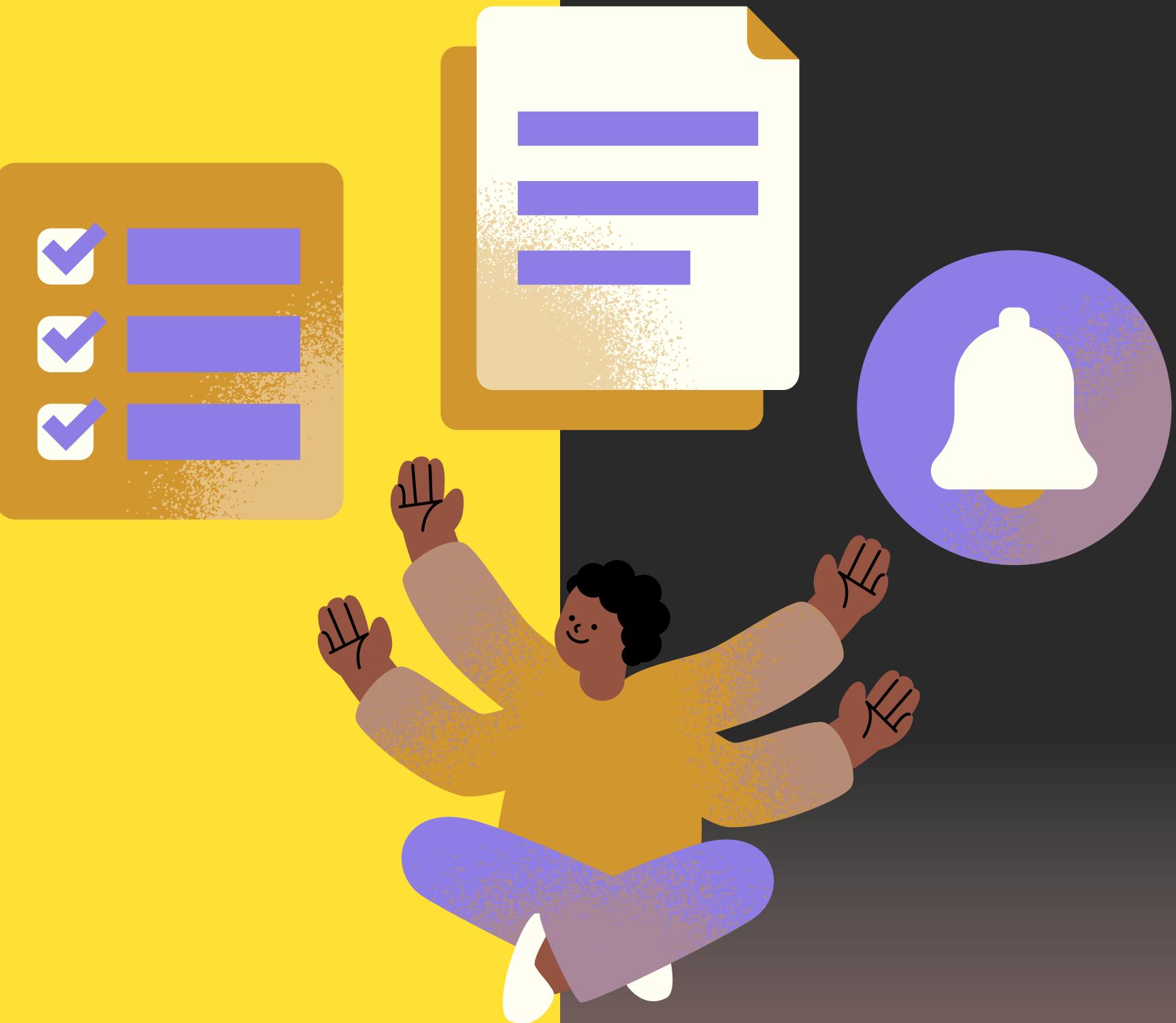
APIs



Este tipo de lenguaje permite realizar operaciones específicas para el tipo de base de datos NoSQL

NoSQL Query Language

DISEÑO DE ARCHIVOS



DISEÑO DE ARCHIVOS

Incluye decisiones con respecto a la naturaleza y contenido del propio archivo, como si fuera a emplear para guardar detalles de las transacciones, datos históricos o información de referencia .



Transacciones

DISEÑO DE ARCHIVOS

FORMATO DE REGISTROS

Los datos que se deben de incluirse en el formato de registros contenidos en el archivo, haciendo referencia a cómo se presenta cada unidad de información dentro del archivo.

LONGITUD DE REGISTRO

Cuando creamos un archivo para almacenar datos, cada registro debe tener una longitud específica. La longitud del registro se determina según las características de los datos que contiene.

DISPOSICIÓN DE REGISTROS

Incluye cómo decidimos organizar la información dentro del archivo, la "secuencia a disposición de los registros", que es cómo ordenamos los registros dentro del archivo.

DEF.



Ejemplo



DEF.



Ejemplo



DEF.



Ejemplo



Archivo como una tabla con filas y columnas, y cada fila representa un "registro" que contiene información relacionada. Cada registro está compuesto por diferentes "campos", que son las columnas de esa tabla, y cada campo guarda un tipo específico de dato, como texto, números o fechas.

Almacenar información de empleados, y cada registro debe contener el nombre, la edad y el salario del empleado, necesitamos definir cuántos caracteres se usarán para el nombre, la edad y el salario. Por ejemplo 30 caracteres.

Supongamos que tenemos un archivo para almacenar información de empleados. Cada registro en el archivo contiene:

1. Nombre completo del empleado.
2. Edad del empleado.
3. Número de identificación único del empleado.
4. Salario mensual del empleado

DISEÑO DE ARCHIVOS

ESTRUCTURAS DE ALMACENAMIENTO

Secuencial



Los registros se almacenan uno detrás del otro en orden secuencial. Es como una lista en la que cada registro sigue al anterior. Es sencillo y fácil de implementar, pero puede ser lento si necesitamos buscar o acceder a registros específicos.

Producto: Camiseta, Precio: \$20, Cantidad: 50
Producto: Pantalón, Precio: \$35, Cantidad: 30
Producto: Zapatos, Precio: \$50, Cantidad: 25

Indexada



Además de los registros, hay una tabla de índices que contiene información para acceder rápidamente a los registros. Cada índice apunta a un registro específico dentro del archivo. Esto permite búsquedas más rápidas

Tabla de índices:

Índice **101** apunta al Registro 1

Índice **102** apunta al Registro 2

Índice **103** apunta al Registro 3

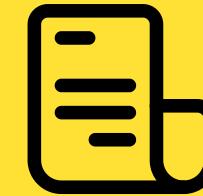
Registros:

Registro 1: Juan, 20, 101, 85

Registro 2: Ana, 19, 102, 78

Registro 3: Pedro, 21, 103, 92

Relativa



los registros se relacionan entre sí mediante referencias relativas en lugar de direcciones absolutas. Cada registro contiene información para encontrar el siguiente o el anterior en la secuencia. Es útil cuando los registros tienen tamaños variables.

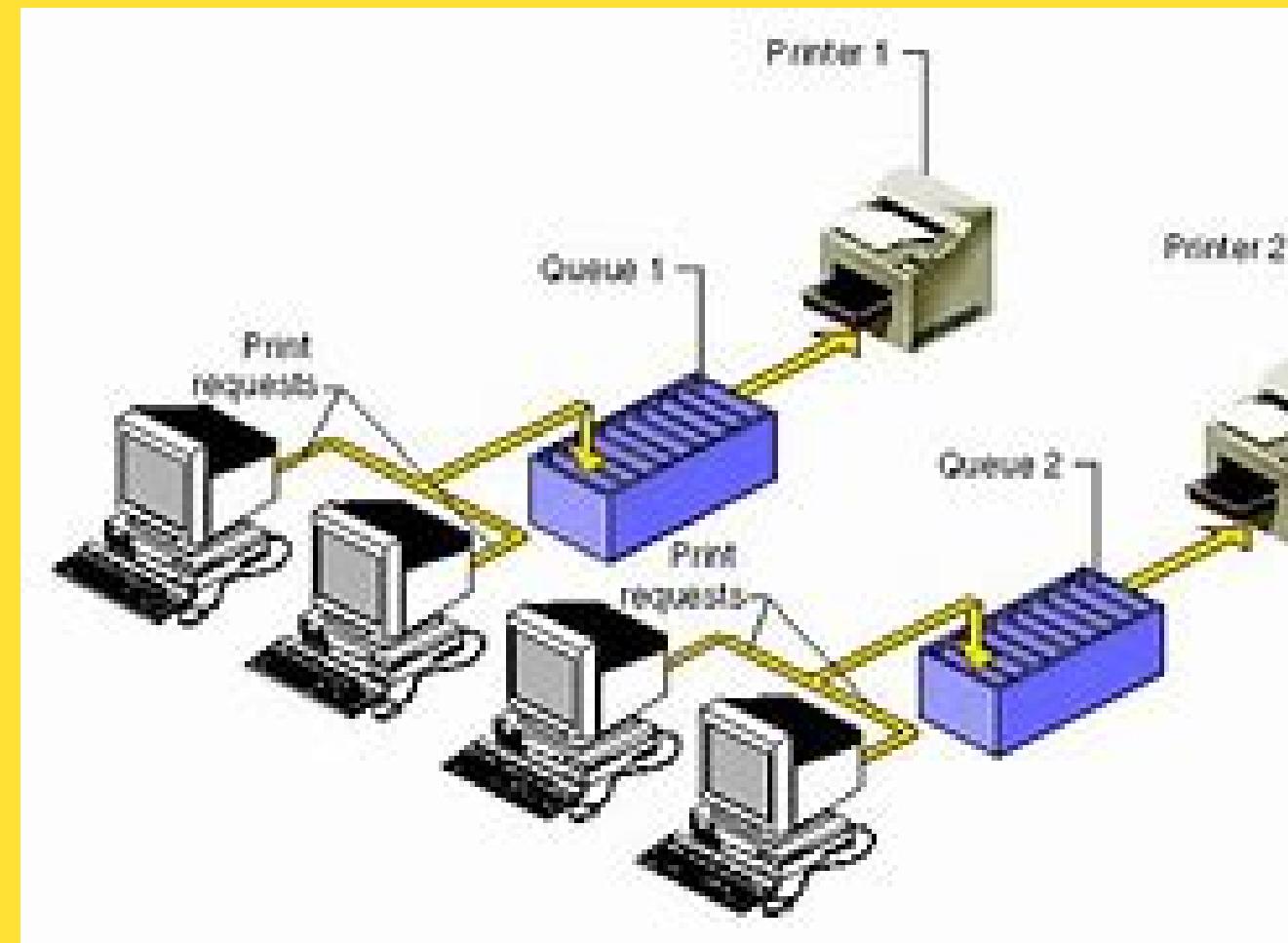
Registro 1: Juan, 20, Siguiente: 2, 85

Registro 2: Ana, 19, Siguiente: 3, 78

Registro 3: Pedro, 21, Siguiente: N/A, 92

Cada registro tiene un campo que indica el número del siguiente registro en la secuencia. Esto es útil cuando los registros tienen tamaños variables y queremos mantener su relación sin depender de direcciones absolutas.

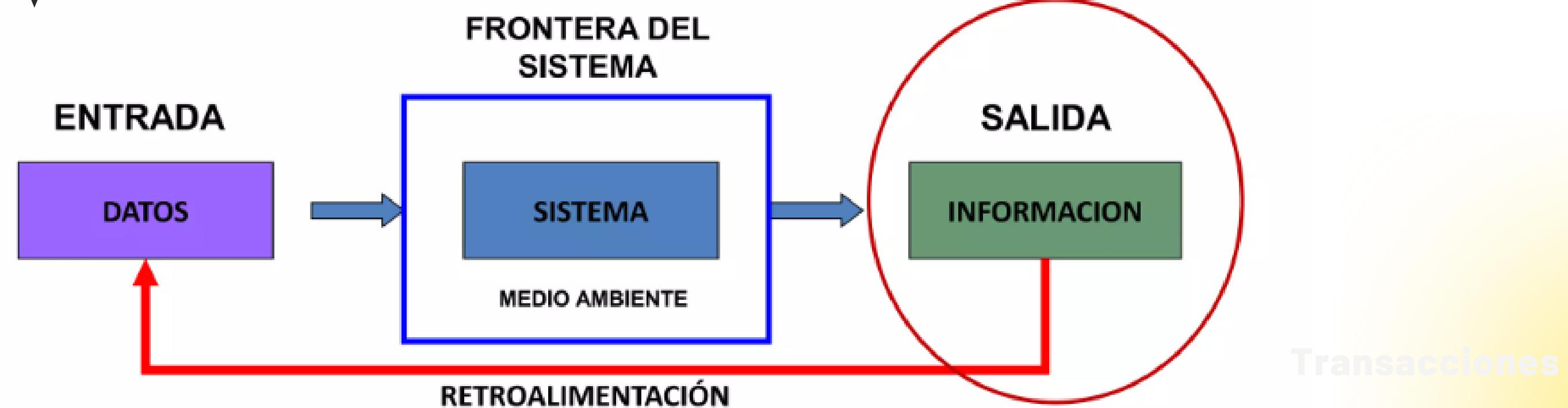
DISEÑO DE SALIDAS



DISEÑO DE SALIDAS

Una salida, es aquella que se entrega al usuario por medio de un sistema de información, donde los datos de entrada realizan varios procesos para que se conviertan en datos de salidas.

El diseño de salidas tiene una importancia notable, ya que los usuarios confían en ellas para tomar decisiones que afectan a toda la empresa, y muchas veces juzgan la utilidad de todo el sistema, sólo por sus salidas.

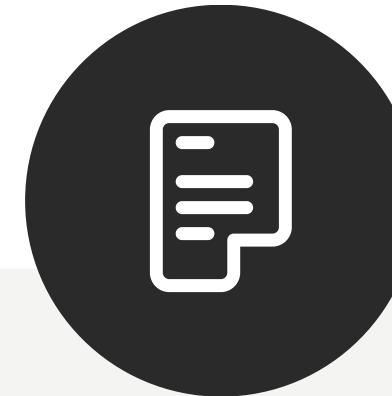


OBJETIVOS



Toda salida debe de tener un propósito. En la fase de análisis se determinan cuáles serán los propósitos que se aplicarán a la hora de ejecutar una salida.

Diseño de salida para que sirva al propósito deseado



Diseñar salidas que los usuarios puedan personalizar

- Encabezado descriptivo
- Instrucciones claras
- Agrupar opciones relacionadas
- Uso de iconos o símbolos
- Confirmaciones y advertencias

Diseño de salida para satisfacer al usuario



Proporciona la información necesaria y relevante sin abrumar al usuario con una sobrecarga de información.

- Relevancia
- Concisión
- Evita la sobrecarga de datos
- Interactividad opcional

Entregar la cantidad adecuada de salida

NoSQL Query
Language

OBJETIVOS



Esto implica colocar la información o resultados de manera estratégica y accesible para que los usuarios puedan encontrarlos fácilmente y en el momento adecuado.

Asegurarse de que la salida se encuentra donde se necesita



Una queja frecuente de los usuarios es que no reciben la información a tiempo para tomar decisiones importantes. Una salida a tiempo garantiza una toma de decisiones oportuna y efectiva

Entrega de la salida a tiempo



Se refiere a la elección de la forma o formato en que se presentará la información o resultados al usuario o receptor.

- Un reporte
- Un documento
- Un mensaje



Selección del método de salida adecuado

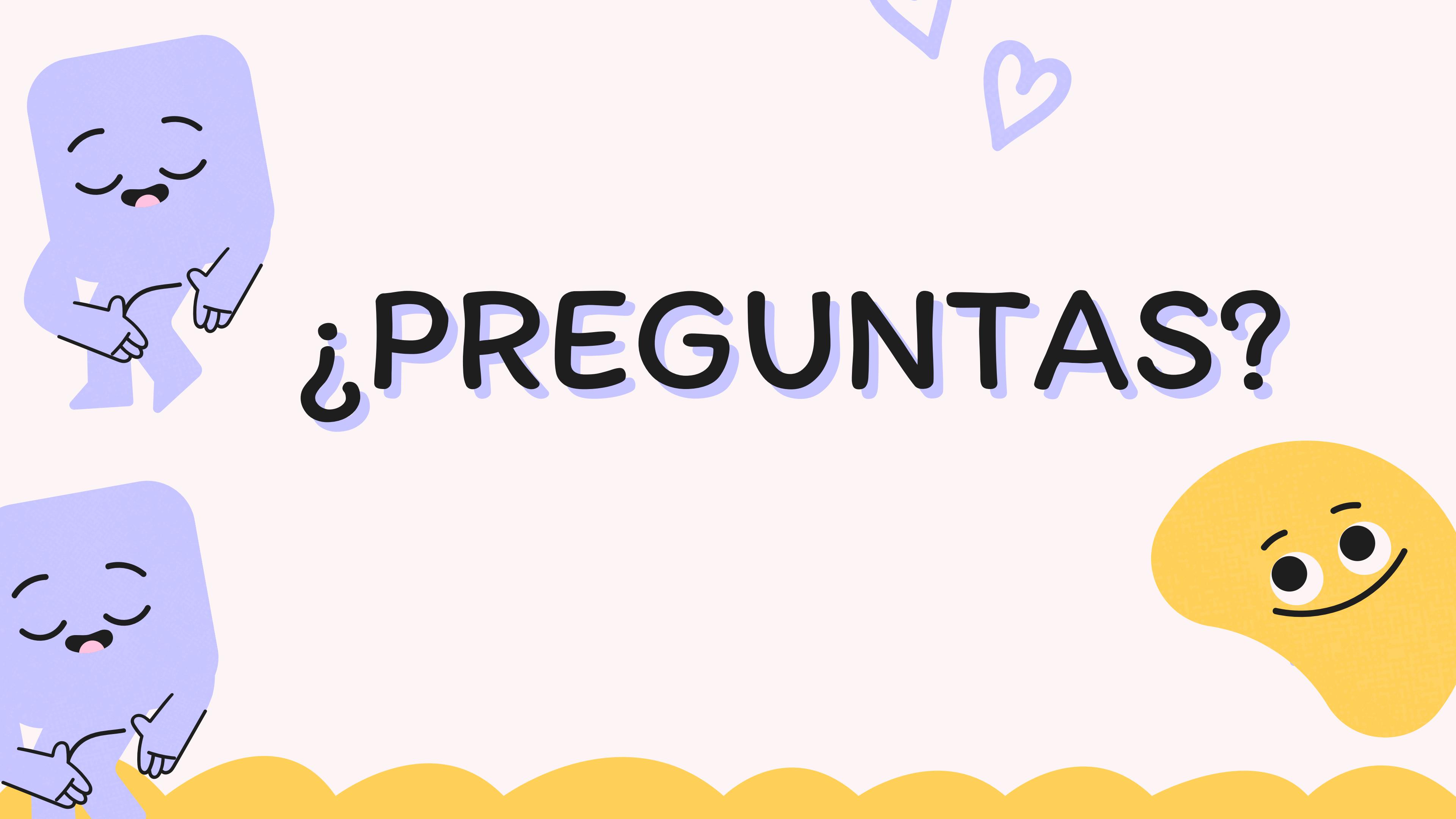
CONCLUSIÓN



Conclusión

El diseño y construcción de software es un proceso complejo e integral que exige un enfoque multidisciplinario. Para garantizar el éxito, es primordial priorizar al usuario y sus necesidades, creando una experiencia intuitiva y atractiva. La modularidad y mantenibilidad aseguran la flexibilidad y facilidad de actualización. La escalabilidad y el rendimiento optimizado son fundamentales para una aplicación eficiente en un entorno cambiante. La seguridad del software se convierte en una prioridad ineludible en un mundo digital amenazante. Las pruebas rigurosas garantizan la calidad y fiabilidad del producto final. La documentación completa facilita el entendimiento, mantenimiento y mejora continua del software. Al integrar estos elementos, los desarrolladores pueden crear aplicaciones sólidas y confiables que satisfagan las expectativas del usuario y alcancen el éxito en el mercado tecnológico.





¡PREGUNTAS?