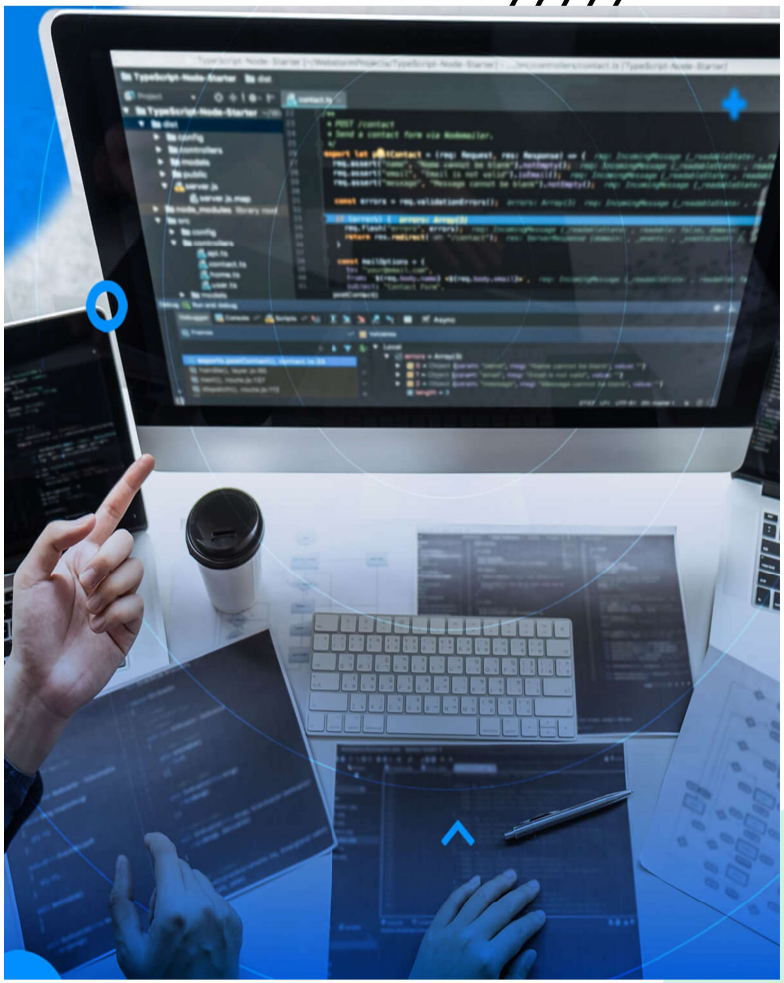
TEMA3

EXPLOTACIÓN DE CÓDIGO

Entornos de Desarrollo

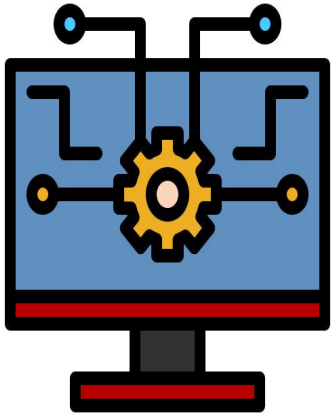
1 DAM/DAW



Índice

1. Introducción.
2. Proceso de obtención de código.
3. Características de un lenguaje de programación: interpretado vs compilado.
4. Depuradores.
5. Reutilización de código.

1. Introducción



El desarrollo del software requiere de un conjunto de herramientas imprescindibles para la generación de programas información.

Cuando nuestro código se ejecuta, pueden producirse **errores lógicos** que solo son detectables durante la **ejecución**. Como desarrolladores debemos saber **cómo solucionarlos**. Sin embargo, en programas grandes suele ser más complicado encontrar los errores, que realizar su corrección. Para facilitar la búsqueda de errores en el software, podemos utilizar **depuradores**.

2. Proceso de obtención de código

Un ordenador es una máquina con una electrónica que solo entiende impulsos eléctricos (0 y 1) dentro de determinado voltaje.

¿Cómo es posible convertir un programa escrito en lenguaje pseudonatural a impulsos eléctricos?

- ✔ Cada procesador es capaz de entender un conjunto determinado de instrucciones en un determinado lenguaje, que recibe el nombre de lenguaje ensamblador o lenguaje máquina.
- ✔ De esta forma el lenguaje ensamblador es el más cercano a la máquina, siendo el nivel más bajo de abstracción en el que se puede crear un software.

Como desarrolladores de software en muy pocas ocasiones tendremos que lidiar con este tipo de lenguaje, pues el código que se genera va a depender del procesador en el que se va a ejecutar.

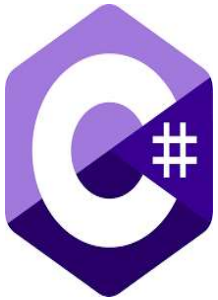
Mediante la creación de lenguajes de alto nivel de abstracción, somos capaces de crear programas informáticos independientes del procesador con mayor facilidad.

3. Características de un lenguaje de programación: interpretado vs compilado

LENGUAJES COMPILADOS

- Requieren de un compilador para pasar de código fuente – código objeto – código máquina.

- Un **compilador** = herramienta que permite traducir de un lenguaje de alto nivel a un lenguaje que pueda comprender la máquina.



- Este tipo de lenguajes diferencian la etapa de compilación de la etapa de ejecución. Si no tenemos esta etapa el rendimiento es mejor.

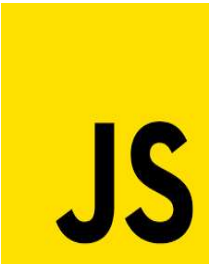
- Para poder realizar la compilación y obtener un programa ejecutable, es necesario disponer también de un enlazador o linker.
  - **Enlazador** = herramienta que permitirá resolver las dependencias que tengamos en nuestro desarrollo, por ejemplo, con componentes externos.

- Aunque el código es más seguro , no son tan flexibles para modificarlos como los lenguajes interpretados.

3. Características de un lenguaje de programación: interpretado vs compilado

LENGUAJES INTERPRETADOS

- No generan código objeto.
- No hay una fase de compilación separada. El código fuente se ejecuta línea por línea directamente por un intérprete.
- El **intérprete** es un programa que tiene que estar cargado en memoria. El intérprete "interpreta" cada sentencia del programa y lo ejecuta.
- Cada vez que se ejecuta el programa, el intérprete analiza y ejecuta el código fuente en tiempo real. No se genera un archivo ejecutable previamente.
- Las instrucciones se traducen " on the fly " (al vuelo), a medida que van ejecutándose.



ENLACE DE IN TERÉS

"Facilitan la vida al programador"

3. Características de un lenguaje de programación: interpretado vs compilado

LENGUAJES COMPILADOS

LENGUAJES INTERPRETADOS

https://www.youtube.com/watch?v=pQCAk6-IH-E&t=507s

4. Depuradores

Un **depurador** (*en inglés, debugger*), es un programa que permite depurar o eliminar los errores de otro programa informático.

Al iniciarse la depuración, el depurador lanza el programa a depurar → este se ejecuta normalmente hasta que el depurador detiene su ejecución, permitiendo al usuario examinar la situación.

El depurador permite detener el programa en:

- Un punto determinado mediante un punto de ruptura.
- Un punto determinado bajo ciertas condiciones mediante un punto de ruptura condicional.
- Un momento determinado cuando se cumplan ciertas condiciones.
- Un momento determinado a petición del **usuario**.

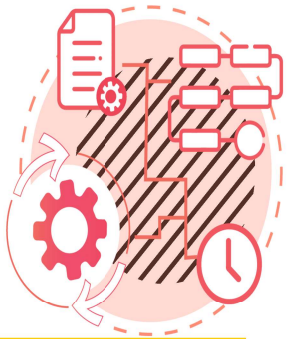
4. Depuradores

Durante esa interrupción, el usuario puede:

1. Examinar y modificar la memoria y las variables del programa.
2. Examinar el contenido de los registros del procesador.
3. Examinar la pila de llamadas que han desembocado en la situación actual.
4. Cambiar el punto de ejecución, de manera que el programa continúe su ejecución en un punto diferente al punto en el que fue detenido.
5. Ejecutar instrucción a instrucción.
6. Ejecutar partes determinadas del código.

El depurador depende de la arquitectura y sistema en el que se ejecute, por lo que sus funcionalidades cambian de un sistema a otro.

5. Reutilización de código

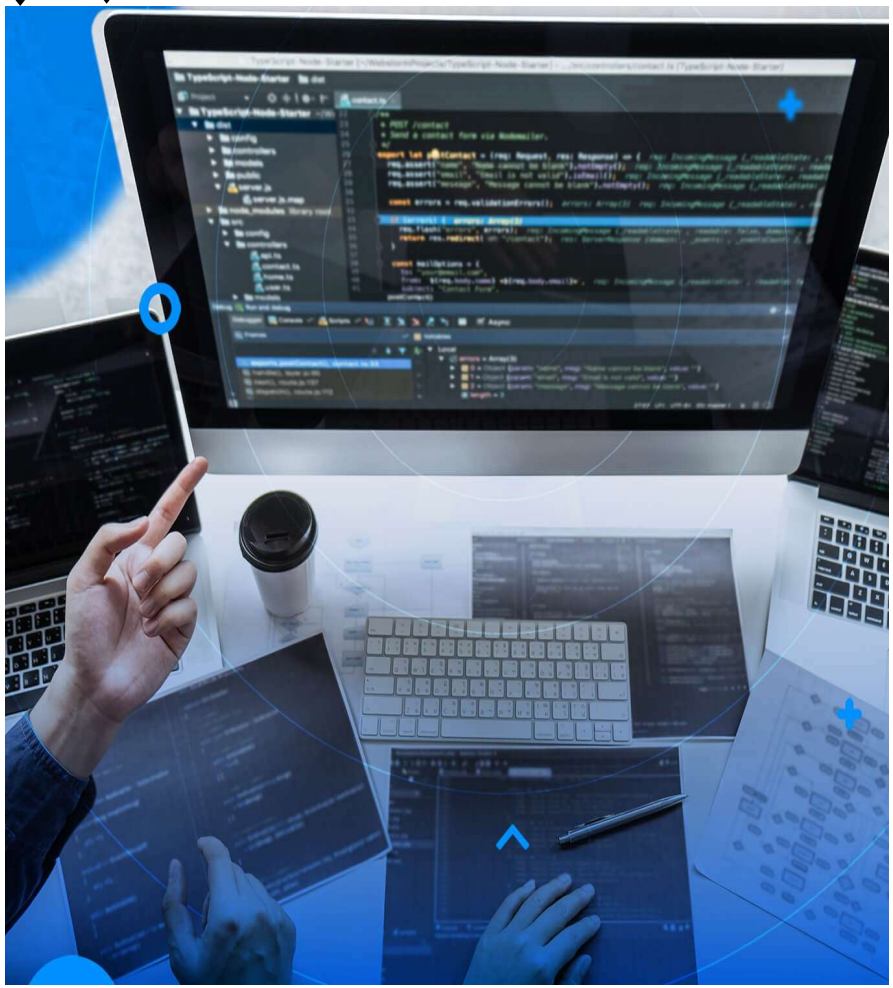


- Una buena práctica de desarrollo es crear un **código estructurado e independiente de un contexto**.

- Si somos capaces de crear componentes aislados que no dependan unos de otros podremos reutilizar código en el futuro. Las funciones, procedimientos, creación de componentes son formas de creación de contenido que nos permiten reutilizar el software.

- Como se puede intuir, la **reutilización de código** puede requerir la creación de componentes reutilizables e independientes que pueden ser de cualquier tipo, por ejemplo, pruebas genéricas, entornos de depuración, etc.

- Una buena aproximación a la reutilización de código es la **creación de librerías que se pueden publicar y ser utilizadas por terceras partes.** Al tener librerías que realizan funcionalidades muy acotadas, es más sencillo crear pruebas simples que sirvan para



¿ALGUNA DUDA?

GRACIAS