Consulta #1: JSon

¿Qué es JSON?

JSON (JavaScript Object Notation) es un formato de texto ligero para almacenar y transmitir datos entre aplicaciones.

Nació en JavaScript, pero hoy se usa en casi todos los lenguajes de programación.

Permite que distintos sistemas (aplicaciones, servidores, bases de datos) se comuniquen de forma sencilla y estandarizada.

Características principales

Ligero y simple: ocupa poco espacio y es fácil de leer.

Legible por humanos: la información está organizada en clave–valor.

Estandarizado: funciona igual en cualquier lenguaje de programación.

Flexible: soporta datos simples y estructuras complejas (listas, objetos anidados).

Universal: ampliamente usado en APIs, bases de datos NoSQL y archivos de configuración.

Sintaxis básica

🔹 Objetos

Se definen con llaves { }

Contienen pares clave–valor

No tienen un orden obligatorio

Ejemplo:

{

"nombre": "Matías",

"edad": 20

}

🔹 Listas / Arreglos

Se definen con corchetes [ ]

Contienen valores en orden

Los elementos no tienen clave, solo posición (índice)

Ejemplo:

["Quito", "Guayaquil", "Cuenca"]

🔹 Tipos de valores en JSON

Tipo Ejemplo

String "Matías"

Number 20, 3.14

Boolean true, false

Array [1, 2, 3]

Object { "clave": "valor" }

Null null

Clave–Valor

Cada elemento dentro de un objeto JSON se llama par clave–valor

La clave siempre es una cadena (string) entre comillas

El valor puede ser cualquier tipo de dato válido en JSON

Ejemplo:

{

"nombre": "Matías", // clave: "nombre", valor: "Matías"

"edad": 20 // clave: "edad", valor: 20

}

Uso de JSON en programación

Comunicación entre cliente y servidor: APIs REST devuelven datos en JSON

Bases de datos NoSQL: MongoDB usa JSON/BSON para almacenar documentos

Archivos de configuración: muchos programas usan .json para guardar parámetros

Lenguajes compatibles: Java, C++, Python, PHP, C#, JavaScript, entre otros

Ejemplo en Java:

JSONObject persona = new JSONObject();

persona.put("nombre", "Matías");

persona.put("edad", 20);

System.out.println(persona.toString());

Salida:

{"nombre":"Matías","edad":20}

Importancia de JSON

Facilita la interoperabilidad entre sistemas y lenguajes

Es más simple y ligero que XML, por eso es más rápido de procesar

Es universal, la mayoría de herramientas modernas lo soportan

Permite crear estructuras de datos complejas con objetos y listas anidadas

Diferencia entre objetos y listas

Concepto Símbolos Contiene Orden

Objeto { } Clave–valor No necesariamente

Lista / Array [ ] Valores Ordenados (por índice)

Un objeto puede contener listas, y una lista puede contener objetos. Son estructuras complementarias.

Representación gráfica simple

{

├─ "nombre": "Matías"

├─ "edad": 20

├─ "activo": true

├─ "materias": [ "Programación", "POO", "Cálculo" ]

└─ "direccion": {

"ciudad": "Quito",

"pais": "Ecuador"

}

}

Resumen final

Qué es: Formato de texto para intercambiar datos.

Elementos básicos: Objetos {}, listas [ ], clave–valor.

Tipos de datos: string, number, boolean, array, object, null.

Ventajas: Ligero, legible, universal, rápido, ampliamente usado.

Usos: APIs, bases de datos, archivos de configuración.

Consulta #2: Espacio en bytes  
Tengo una clase llamada Datos que posee dos atributos: uno de tipo int y otro de tipo float.

Cada tipo int y float ocupa 32 bits, es decir 4 bytes cada uno, por lo tanto, el objeto en total ocupa 8 bytes en memoria (4 bytes del int + 4 bytes del float).

En cuanto a los métodos (get, set o constructores), estos no ocupan espacio adicional en memoria dentro del objeto, ya que su código se almacena en la memoria de código (segmento de texto) del programa.

Todos los objetos de la clase comparten el mismo código de los métodos, por lo que no se duplica en memoria cada vez que se crea un nuevo objeto.

En conclusión:

El espacio total ocupado en memoria por un objeto de la clase Datos es de 8 bytes, correspondientes únicamente a los atributos.

Representación gráfica:  


  
  
