Persistencia con lenguajes estructurados

0. Introducción

El objetivo de este documento es conocer qué es JSON y para que se suele utilizar. Este formato es ampliamente utilizado en el desarrollo de aplicaciones específicamente para la transferencia de información, ya que al final es un texto plano con todas las consideraciones y ventajas que eso conlleva.

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos que es fácil de leer y escribir tanto para los humanos como para las máquinas. Se usa comúnmente para transmitir datos entre un servidor y un cliente en aplicaciones web. A continuación, te explico lo básico de JSON con algunos ejemplos.

1. Estructura Básica de JSON

El formato de JSON se basa en dos estructuras principales:

- **Objetos**: Se representan con llaves {}. Un objeto es una colección de pares clave-valor.
- **Arreglos (Arrays)**: Se representan con corchetes []. Un arreglo es una lista ordenada de valores.

Pairs clave-valor:

- **Clave**: Es una cadena de texto que describe el nombre del dato.
- **Valor**: Puede ser un número, cadena de texto, booleano (true o false), objeto, arreglo o null.

2. Ejemplos básicos

Ejemplo 1: Objeto JSON simple

```
{
  "nombre": "Juan",
  "edad": 30,
  "activo": true
}
```

- "nombre", "edad", y "activo" son **claves**.
- "Juan", 30, y true son **valores** asociados a esas claves.

Ejemplo 2: JSON con una matriz

```
{
  "nombre": "Ana",
  "hobbies": ["leer", "correr", "viajar"]
}
```

- La clave "hobbies" tiene como valor un **arreglo** de tres elementos: "leer", "correr" y "viajar".

Ejemplo 3: JSON con objetos anidados

```
{
  "nombre": "Carlos",
  "direccion": {
    "calle": "Avenida Siempre Viva",
    "numero": 742
  }
}
```

La clave "direccion" tiene como valor un **objeto** anidado que contiene dos claves: "calle" y "numero".

Ejemplo 4: JSON con varios objetos dentro de una matriz

```
{
 "personas": [
  {
   "nombre": "Laura",
   "edad": 25
  },
  {
   "nombre": "Pedro",
   "edad": 28
  }
 ]
}
```

La clave "personas" contiene una matriz de dos objetos, donde cada objeto tiene claves "nombre" y "edad".

Ejemplo 5: JSON con valores de tipos diferentes

```
{
 "nombre": "Luis",
 "edad": 22,
 "esEstudiante": true,
 "promedio": 8.5,
 "direccion": null
}
```

Los valores pueden ser cadenas ("Luis"), números (22, 8.5), booleanos (true), y null.

3. Reglas de Sintaxis en JSON

- **Comillas dobles** (") deben usarse para las claves y los valores de texto (cadenas).
- Los valores deben estar separados por comas.
- No se permite una coma extra al final de un arreglo u objeto.
- Los objetos deben estar rodeados por {} y los arreglos por [].

4. Uso común de JSON

JSON es ampliamente utilizado en aplicaciones web para intercambiar datos entre el cliente y el servidor. Un ejemplo típico es cuando se hace una solicitud HTTP a un servidor, que responde con datos en formato JSON.

2. Usos de JSON

1. Intercambio de Datos entre Cliente y Servidor (APIs RESTful)

Una de las aplicaciones más comunes de JSON es como formato de intercambio de datos entre un cliente (generalmente un navegador web o una aplicación móvil) y un servidor.

Ejemplo:

- Cuando un usuario interactúa con una aplicación web, como una tienda en línea, la interfaz de usuario puede hacer una solicitud HTTP al servidor para obtener los detalles de los productos. El servidor responderá con un objeto JSON que contiene información sobre los productos, como el nombre, precio, disponibilidad, etc.

```
{
 "productos": [
  {
   "id": 1,
   "nombre": "Camiseta",
   "precio": 19.99,
   "disponibilidad": "En stock"
  },
  {
   "id": 2,
   "nombre": "Pantalón",
   "precio": 39.99,
   "disponibilidad": "Agotado"
  }
 ]
}
```

2. Configuración de Aplicaciones

JSON se usa comúnmente para almacenar configuraciones de aplicaciones. Dado que es fácil de leer y escribir, muchos programas, tanto en el servidor como en el cliente, lo emplean para almacenar sus parámetros de configuración.

Ejemplo:

- **Configuración de una base de datos** en una aplicación backend, donde se definen los parámetros como el host, puerto, usuario y contraseña.

{

```
 "db": {
   "host": "localhost",
   "port": 5432,
   "usuario": "admin",
   "password": "secreta"
 }
}
```

3. Almacenamiento Local en el Navegador (LocalStorage o SessionStorage)

JSON se utiliza ampliamente para almacenar información en el navegador del usuario. Esto puede incluir preferencias del usuario, datos de sesión, y cualquier otra información que se deba conservar entre las recargas de la página.

Ejemplo:

- Puedes almacenar un objeto JSON con la configuración personalizada del usuario, como temas de colores o preferencias de idioma.

```
{
 "idioma": "es",
 "tema": "oscuro"
}
```

Este JSON se puede guardar en localStorage del navegador para que persista incluso cuando el usuario cierre y vuelva a abrir la página.

4. Serialización y Deserialización de Datos

JSON es frecuentemente utilizado para **serializar** (convertir a texto) y **deserializar** (convertir de nuevo a objetos) datos en muchas aplicaciones. Esto es útil cuando se necesita almacenar o transmitir datos complejos.

Ejemplo:

- En una aplicación móvil o de servidor, podrías tener una clase de "usuario", y quieres enviar los datos del usuario a través de una API. El objeto de la clase se convierte a JSON para enviarlo por la red.

```
{
 "id": 1,
 "nombre": "María",
```

```
 "email": "maria@ejemplo.com",

 "activo": true

}

{

 "id": 1,

 "nombre": "María",

 "email": "maria@ejemplo.com",

 "activo": true

}
```


5. Intercambio de Datos entre Microservicios

En arquitecturas basadas en microservicios, los diferentes servicios a menudo se comunican entre sí utilizando JSON. Esto es porque JSON es independiente del lenguaje, lo que permite que servicios escritos en diferentes lenguajes (Java, Python, Node.js, etc.) intercambien datos sin problemas.

Ejemplo:

- Un servicio de autenticación puede devolver un token de acceso como un objeto JSON:


```
{

 "token": "abc123xyz456",

 "expiracion": "2025-12-31T23:59:59Z"

}
```


6. Bases de Datos NoSQL (Como MongoDB)

Muchas bases de datos NoSQL, como **MongoDB**, almacenan datos en formato BSON (una extensión binaria de JSON). Esto permite representar datos complejos y jerárquicos de manera eficiente.

Ejemplo:

- En MongoDB, un documento podría ser almacenado como JSON para representar un usuario con su información.


```
{

 "_id": "507f1f77bcf86cd799439011",

 "nombre": "Elena",
```

```
 "edad": 30,
 "correo": "elena@ejemplo.com",
 "direccion": {
  "calle": "Avenida Central",
  "numero": 100
 }
}
```

7. Intercambio de Datos entre Frontend y Backend

En aplicaciones web modernas, JSON se usa como formato estándar para intercambiar datos entre el **frontend** (interfaz de usuario) y el **backend** (servidor). Esto es especialmente común en aplicaciones web basadas en JavaScript (usando frameworks como React, Angular o Vue.js), ya que JSON es muy compatible con JavaScript.

Ejemplo:

- El **frontend** hace una solicitud para obtener los detalles de un producto y luego lo muestra en la interfaz de usuario:

Frontend (JavaScript):

```
fetch('https://api.ejemplo.com/productos')
 .then(response => response.json())
 .then(data => {
  console.log(data);
  // Mostrar los productos en la UI
 });
```

Backend (API REST en JSON):

```
{
 "id": 1,
 "nombre": "Laptop",
 "precio": 999.99,
 "esNovedad": true
}
```

8. Mensajes y Notificaciones

En muchas aplicaciones, especialmente en sistemas de mensajería o notificaciones en tiempo real, JSON es utilizado para estructurar los mensajes que se envían.

Ejemplo:

- Un servidor podría enviar una notificación en formato JSON a un cliente para informarle sobre una actualización o un mensaje nuevo.

```
{
 "titulo": "Nueva oferta",
 "mensaje": "¡Obtén un 20% de descuento en tu próxima compra!",
 "fecha": "2025-01-08T12:00:00Z"
}
```

9. Interfaz de Configuración de Aplicaciones (Panel de Administración)

Muchas aplicaciones web incluyen paneles de administración donde los administradores configuran la aplicación o gestionan datos. En estos paneles, el backend puede usar JSON para enviar configuraciones o estadísticas al frontend.

Ejemplo:

- Un sistema de gestión de usuarios podría usar un JSON para mostrar estadísticas sobre los usuarios registrados.

```
{
 "usuarios": 500,
 "activos": 450,
 "inactivos": 50
}
```

JSON es un formato fundamental en el desarrollo de aplicaciones modernas. Sus usos van desde la **comunicación entre servidores y clientes** hasta el **almacenamiento de configuraciones** y la **integración de bases de datos**. Dado que es sencillo, flexible y ampliamente compatible, es una herramienta indispensable para los desarrolladores de hoy en día.