1. Introducción

JSON Schema es un estándar que define un formato para describir la estructura y las restricciones que deben cumplir los documentos JSON. Proporciona una forma de validar y documentar la estructura de un objeto JSON, permitiendo establecer reglas sobre qué datos son permitidos, sus tipos, valores y otras características.

A continuación, se presenta una introducción teórica a algunos conceptos clave relacionados con JSON Schema:

- **Estructura Básica de JSON Schema**

JSON Schema utiliza un documento JSON para describir la estructura deseada del objeto JSON. El esquema puede contener varios elementos, pero generalmente incluye propiedades como "type" para definir el tipo de datos esperado y "properties" para especificar las propiedades del objeto.

json

```
{
  "type": "object",
  "properties": {
    "nombre": { "type": "string" },
    "edad": { "type": "integer" }
  },
  "required": ["nombre"]
}
```

- **Tipos de Datos**

object: Define un objeto JSON.
array: Define una lista de elementos.
string: Define una cadena de texto.
number: Define un número, puede ser entero o decimal.
integer: Define un número entero.
boolean: Define un valor booleano (true o false).
null: Define un valor nulo.

- **Propiedades y Objetos Anidados**

JSON Schema permite definir propiedades dentro de objetos y objetos anidados, lo que facilita describir estructuras complejas.

json

```
{
  "type": "object",
  "properties": {
    "persona": {
      "type": "object",
      "properties": {
        "nombre": { "type": "string" },
        "edad": { "type": "integer" }
      },
      "required": ["nombre"]
    },
    "hobbies": {
      "type": "array",
      "items": { "type": "string" }
    }
  }
}
```

- **Validación de Requerimientos**

required: Especifica las propiedades que deben estar presentes en el objeto.
minLength, maxLength: Define la longitud mínima y máxima de una cadena.
minimum, maximum: Establece los valores mínimo y máximo para números.

- **Validación de Formato**

format: Permite validar formatos específicos, como direcciones de correo electrónico, fechas, etc.

json

```
{
  "type": "object",
  "properties": {
    "correo": { "type": "string", "format": "email" },
    "fecha_nacimiento": { "type": "string", "format": "date" }
  }
}
```

- **Referencias y Combinación de Esquemas**

JSON Schema permite hacer referencia a otros esquemas y combinar múltiples esquemas para describir estructuras más complejas.

json

```
{
  "$ref": "definiciones.json#/definitions/persona"
}
```

Ejemplo Completo:

json

```
{
  "type": "object",
  "properties": {
    "nombre": { "type": "string" },
    "edad": { "type": "integer" },
    "correo": { "type": "string", "format": "email" },
    "hobbies": {
      "type": "array",
      "items": { "type": "string" }
    }
  },
  "required": ["nombre", "edad"]
}
```

En este ejemplo, el objeto JSON debe tener propiedades "nombre" y "edad", donde "nombre" es una cadena de texto, "edad" es un número entero y "correo" es una cadena de texto que debe seguir el formato de dirección de correo electrónico. La propiedad "hobbies" es opcional y, si está presente, debe ser un array de cadenas.

Estos son solo algunos conceptos básicos de JSON Schema. La capacidad de describir y validar estructuras de datos JSON de manera precisa es esencial para garantizar la consistencia y la integridad en aplicaciones que trabajan con datos en formato JSON.

Vamos a crear un ejemplo sencillo de JSON Schema y un JSON correspondiente que cumpla con ese esquema. En este ejemplo, definiremos un esquema para un objeto que representa información sobre una persona, incluyendo su nombre, edad y una lista opcional de hobbies. A continuación, se presenta el JSON Schema y un ejemplo de JSON que cumple con ese esquema:
JSON Schema:

json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "nombre": {
      "type": "string",
      "minLength": 1
```

```
  },
  "edad": {
   "type": "integer",
   "minimum": 0
  },
  "hobbies": {
   "type": "array",
   "items": {
    "type": "string"
   }
  }
 },
 "required": ["nombre", "edad"]
}
```

Este JSON Schema describe un objeto JSON que debe tener las propiedades "nombre" y "edad". La propiedad "nombre" debe ser una cadena de texto con una longitud mínima de 1, la propiedad "edad" debe ser un número entero con un valor mínimo de 0. La propiedad "hobbies" es opcional y, si está presente, debe ser un array que contiene elementos de tipo cadena.
JSON Correspondiente:

json

```
{
 "nombre": "Juan",
 "edad": 30,
 "hobbies": ["viajar", "leer"]
}
```

Este JSON cumple con el esquema especificado. Tiene las propiedades requeridas "nombre" y "edad", y la propiedad opcional "hobbies" es un array de cadenas.

Si proporcionas este JSON a un validador que utilice el esquema que definimos, debería pasar la validación. Ten en cuenta que este es solo un ejemplo básico, y en aplicaciones del mundo real, los esquemas pueden volverse más complejos para abordar casos específicos y garantizar la integridad de los datos.

Cuando deseas indicar en un archivo JSON que estás utilizando un esquema específico para validación, puedes hacer uso de la propiedad especial $schema. Esta propiedad se utiliza para declarar la URL del esquema JSON que se está utilizando para validar el documento. A continuación, te muestro un ejemplo donde se utiliza la propiedad $schema en un archivo JSON:

json

```
{
 "$schema": "http://json-schema.org/draft-07/schema#",
 "nombre": "Juan",
```

```
  "edad": 30,
  "hobbies": ["viajar", "leer"]
}
```

En este ejemplo, la propiedad $schema apunta a la URL del esquema JSON. En este caso, se utiliza el esquema de la especificación de JSON Schema versión 7 (draft-07). Puedes cambiar la URL según la versión del esquema JSON que estás utilizando.

El uso de la propiedad $schema es opcional, pero es una práctica recomendada, ya que proporciona información adicional sobre la versión y la ubicación del esquema JSON que debe usarse para validar el archivo. Esto facilita la comprensión del contexto y ayuda a los validadores a seleccionar el esquema correcto para la validación.