Validació JSON

JSON Schema

1.- Introducció

Què és JSON Schema

- És una ferramenta que s'utilitza per **validar i descriure l'estructura de les dades i les seues restriccions** quan treballem amb formats JSON.
- La seua **funció** principal és la de establir un **conjunt de regles** i especificacions que definiran com han de ser les dades d'un document JSON.
- Estes regles inclouen especificacions sobre els **tipus de dades**, **les propietats** i **les restriccions** que s'han d'aplicar a les dades.
- JSON Schema ens ajuda a garantir la integritat i la consistència de les dades JSON.

Per què s'ha d'utilitzar JSON Schema?

- **Validació de Dades:** Assegura que les dades en format JSON compleixen amb les restriccions i els estàndards definits. Això ajuda **a evitar errors i dades incorrectes.**
- **Documentació**: Serveix com a **documentació** inherent de les dades**, ja que descriu l'estructura i les restriccions.** Facilita la comprensió i la comunicació de les dades.
- **Interoperabilitat:** Ajuda a assegurar que les dades siguen compatibles amb altres sistemes i aplicacions. Totes les parts poden seguir les mateixes regles.

Json Schema s'usa en entorns on la **consistència de dades** són fonamentals, com en la **validació de sol·licituds i respostes** en serveis web, **configuració d'aplicacions** i altres situacions on s'utilitzen dades estructurades

En definitiva, **JSON Schema** és una eina poderosa que **millora la qualitat de les dades en format JSON**, assegura la seva compatibilitat i facilita la seva documentació, i s'utilitza en una àmplia gamma d'aplicacions i sistemes.

Com funciona?

1. **Creació del JSON Schema**: Document en format JSON que descriu l'estructura i les restriccions de les dades que es volen validar. Aquest document defineix els tipus de dades, les propietats, les restriccions i altres aspectes relacionats amb les dades.

2. **Validació de Dades:** Una volta creat, l'utilitzem per validar les dades en el nostre document JSON. Aplicant el JSON Schema al document JSON a través de les diferents eines o biblioteques que ofereixen suport a JSON Schema.

3. **Resultats de la Validació:** Si les dades compleixen totes les regles definides en el JSON Schema, la validació és un èxit i les dades es consideren vàlides. En cas contrari, es produeix un error de validació i es notifiquen les raons específiques de la invalidesa de les dades.

4. **Utilització en Aplicacions:** Aplicarem JSON Schema l'entrada de dades per assegurar que les dades siguin correctes i compleixin amb els estàndards definits

Exemple Simple:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    },
    "name": {
      "type": "string"
    },
    "price": {
      "type": "number"
    }
  },
  "required": ["id", "name", "price"]
}
```

- **Exemple Complex:**

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",
  "properties": {
    "id": {
      "description": "The unique identifier for a product", "type": "integer"
    },
    "name": {
      "description": "Name of the product", "type": "string"
    },
    "price": {
      "type": "number"
    }
  },
  "required": ["id", "name", "price"]
}
```

En els dos exemples trobem:

```
"required": ["id", "name", "price"]
```

Vàlid	Invàlid
{ "id": 1, "name": "A green door", "price": 12.50, }	{ "id": 1, "name": "A green door", }

2.- Sintaxi de JSON Schema

JSON Schema és en realitat un document JSON amb unes propietats especials. Podem veure la referència completa de l'especificació a: http://json-schema.org/latest/json-schema-core.html

Elements Generals

- **$schema**: Indica la versió de JSON Schema que volem utilitzar. En este cas, la versió 0.7.

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

- **title**: Aquest camp proporciona un títol que ajuda a identificar l'objecte JSON.

```
"title": "Exemple de Persona"
```

- **description**: Descripció del que es va a representar l'esquema JSON.

```
"description":"Esquema que descriu una estructura per a representar informació de persones."
```

- **type**: Defineix el tipus de dades que s'espera per a un element. Pot ser un tipus primitiu com **"string**," "**number**," "**boolean**," o un **objecte o array**.

```
"type": "string"    S'utilitza per a cadenes de text

"type": "number"    Fa referència a numeros decimals

"type": "integer"   Número enters

"type": "boolean"   Indica valor nul o buit

"type": "object"    Defineix un objecte

"type": "array"     Indica un array d'elements
```

També admet **combinacions** de tipus:

```
"type": ["string", "number"]    Admet cadenes de text i nombres decimals
```

I també tipus polimòrfics (complexos):

Objectes: `{ "type": "object", "properties": { "name": { "type": "string" }, "age": { "type": "integer" } } }`

Arrays: `{ "type": "array", "items": { "type": "string" } }`

Elements relacionats amb OBJECTES

- **required**: Enumera les propietats d'un objecte que són obligatòries. Es a dir, un **Array** d'opcions.

Per exemple, si volem que les propietats "**nom**" i "**edat**" siguin obligatòries:

```
"required": ["nom", "edat"]
```

- **properties**: Si l'objecte que s'està descrivint és un objecte JSON (**"type": "object"**), aquesta propietat conté una llista de les seves propietats i les seves definicions. Per exemple, si volem descriure un objecte amb dues propietats, **"nom"** i **"edat"**:

```
{
  "type": "object",
  "properties": {
    "nom": {
      "type": "string"
    },
    "edat": {
      "type": "integer"
    }
  }
}
```

- **additionalProperties**: especifica si es permeten propietats addicionals en un objecte JSON que no estan definides en l'esquema. Es a dir, que es poden afegir propietats **no previstes a** l'objecte, però han de seguir l'esquema.

```
"additionalProperties": {
  "type": "string"
}
```

En este cas, es permeten propietats adicionals que siguen cadenes de text.

També pot tindre valors **true** o **false**

```
"additionalProperties": true   : Es permeten propietats addicionals sense restriccions.
"additionalProperties": false  : No es permeten propietats adicionals.
```

Per exemple:

<table>
<tr>
<td>

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "nom": {
      "type": "string"
    },
    "edat": {
      "type": "integer"
    }
  },
  "additionalProperties": {
    "type": ["string", "integer"]
  }
}
```

</td>
<td>

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "nom": {
      "type": "string"
    },
    "edat": {
      "type": "integer"
    }
  },
  "additionalProperties": false
}
```

</td>
</tr>
<tr>
<td>

```
{
  "nom": "Pepe",
  "edat": 30,
  "poblacio": "La Llosa",
  "punts": 85
}
```

</td>
<td>

```
{
  "nom": "Pepe",
  "edat": 30,
}
```

</td>
</tr>
<tr>
<td>Les propietats "poblacio" i "punts" són addicionals i compleixen amb l'esquema, ja que una és de tipus "string" i l'altra de tipus "integer". Això permet flexibilitat en les propietats addicionals amb diversos tipus possibles.</td>
<td>Amb aquest esquema, només es permetran les propietats "nom" i "edat"i qualsevol altra propietat serà considerada invàlida.</td>
</tr>
</table>

Si modifiquem l'exemple anterior amb:

```
  "additionalProperties": {
    "type": ["string", "integer", “boolean”]
  }
```

Vàlid	Invàlid
`{ "nom": "Ana", "edat": 54, "poblacio": "Eslida", "teMascotes": false }`	`{ "nom": "Ana", "edat": 54, "habilitats": ["programació", 42, true] }`

- **$ref**: S'utilitza per referenciar un esquema extern mitjançant una URI. Això permet reutilitzar esquemes o referir-se a definicions externes:

```
"$ref": "http://example.com/schemas/person-schema"
```

Estes rutes o URL proporcionades amb la referència `"$ref"` poden ser tant locals com a remotes.

- **definitions**: ofereix un espai dedicat per organitzar i definir esquemes addicionals. Esta pràctica millora la llegibilitat i permet la reutilització d'esquemes en diferents parts del teu JSON Schema.

Estructura de "definitions"

```
{
  "definitions": {
    "NomEsquema1": {
      // Definició d'esquema 1
    },
    "NomEsquema2": {
      // Definició d'esquema 2
    },
    // Altres definicions d'esquemes...
  },
  // Altres parts del JSON Schema...
}
```

Dins del nostre JSON Schema principal, podem referenciar estes definicions utilitzant la paraula clau `"$ref"`. La ruta d'aquesta referència comença amb `"#/definitions/"` seguida del nom de l'esquema que volem utilitzar.

Utilització de "definitions" dins de l'esquema principal

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "prop1": {
      "$ref": "#/definitions/NomEsquema1"
    },
    "prop2": {
      "$ref": "#/definitions/NomEsquema2"
    },
    // Altres propietats...
  },
  "definitions": {
    "NomEsquema1": {
      // Definició d'esquema 1
    },
    "NomEsquema2": {
      // Definició d'esquema 2
    },
    // Altres definicions d'esquemes...
  },
  // Altres parts del JSON Schema...
}
```

Exemple:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "nom": {
      "type": "string"
    },
    "edat": {
      "type": "integer"
    },
    "poblacio": {
      "$ref": "#/definitions/Poblacio"
    },
    "punts": {
      "$ref": "#/definitions/Punts"
    }
  },
  "definitions": {
    "Poblacio": {
      "type": "string"
    },
    "Punts": {
      "type": "integer"
    }
  },
  "additionalProperties": {
    "type": ["string", "integer"]
  }
}
```

Elements relacionats amb Arrays

- **items**: Si l'objecte és un array (**"type": "array"**), **items** conté la definició dels elements de l'array.

    - **items** com a valor **unic** de tots els elements de l'array:

```
{
  "type": "array",
  "items": {
    "type": "string"
  }
}
```

En este cas, tots els elements de l'array seran String

    - **items** com a **Llista** de tipus:

```
{
  "type": "array",
  "items": [
    { "type": "string" },
    { "type": "number" },
    { "type": "boolean" }
  ]
}
```

El primer element és obligatòriament un String, el segon un numero decimal i el tercer un valor booleà

- **minItems** i **maxItems**: Aquestes propietats indiquen el nombre mínim i màxim d'elements que s'esperen en una llista. Per exemple, si volem que una llista contingui com a mínim 2 elements i com a màxim 5 elements:

```
{
  "type": "array",
  "items": { "type": "string" },
  "minItems": 2,
  "maxItems": 5
}
```

- **additionalItems**: Aquesta propietat s'utilitza per indicar si s'accepten elements addicionals que no estan definits en l'esquema.

`"additionalItems": true` : Es permeten elements addicionals.
`"additionalItems": false` : No es permeten elements adicionals.

Exemple

```
{
  "type": "array",
  "items": [
    { "type": "string" },
    { "type": "number" }
  ],
  "additionalItems": { "type": "boolean" }
}
```

Els dos primers elements de l'array han de ser, respectivament, cadenes de text i números.
L'element `additionalItems` especifica que quan posem elements addicionals (a continuació dels dos primers), hauran de ser de tipus booleà `"type": "boolean"`).

ENUMS

- **enum**: Ens permet restringir els valors que pot tindre una propietat al d'una llista, es a dir, que una propietat pot prendre **un** dels valors enumerats i cap altre, els valors en enums són "`literals`" i no expressions regulars.

```
{
  "color": {
    "type": "string",
    "enum": ["roig", "blau", "verd"]
  }
}
```

En este cas "color" només pot prendre **un** dels valors de l'array, és a dir, "roig", "blau", "verd"

- **enum** es pot utilitzar tant amb **objectes** com **arrays**

Arrays	

`{`
`  "noms": {`
`  "type": "array",`
`  "enum": [`
`  ["Juan", "Fina"],`
`  [1, 2, 3]`
`  ]`
`  }`
`}`
	La propietat `"noms"` sols pot ser igual a dos arrays concrets `["Juan", "Fina"]`,o `[1, 2, 3]`

Objectes

```
{
  "persones": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "nom": {
          "type": "string"
        },
        "edat": {
          "type": "integer"
        }
      },
      "required": ["nom", "edat"],
      "additionalProperties": false
    },
    "enum": [
      [
        { "nom": "Jaume", "edat": 55 },
        { "nom": "Bel", "edat": 25 }
      ],
      [
        { "nom": "Sergi", "edat": 35 },
        { "nom": "Saioa", "edat": 29 }
      ]
    ]
  }
}
```

Definim una llista d'objectes `"persones"`. Sols permetem dos conjunts d'objectes.

El primer conjunt conté dues persones (**`Jaume`** i **`Bel`**) i el segon conjunt conté dues persones diferents (**`Sergi`** i **`Saioa`**).

Si l'objecte en la llista no coincideix amb un d'estos valors, no cumpleix amb l'esquema

Exemple

```
{
  "geometria": {
    "type": "object",
    "required": ["coordinades", "tipus"],
    "properties": {
      "coordinades": {
        "type": "array",
        "items": {
          "type": "number"
        },
        "minItems": 2,
        "maxItems": 2
      },
      "tipus": {
        "type": "string",
        "enum": ["Punt"]
      }
    },
    "additionalProperties": false
  }
}
```

En este exemple:

- `"type"` especifica el tipus de l'objecte que es vol validar, en aquest cas, `"object"`
- `"required"` és un array que enumera les propietats dins de `"geometria"` que són obligatòries. En aquest cas, les propietats `"coordinades"` i `"tipus"` són obligatòries.
- "`properties`" és un objecte que defineix les propietats vàlides de `"geometria"`. A l'exemple: `"coordinades"` i `"tipus"`, cadascuna amb les seves regles de validació.

Dins de `"properties"` tenim:

- `"coordinades"`: de tipus `"array"`
  - Utilitza l'element `"items"` per especificar el tipus de les dades que pot contenir l'array: `"number"`.
  - Utilitza `"minItems"` per especificar que com a mínim ha de contenir 2 elements i `"maxItems"` per indicar que com a màxim ha de contenir 2 elements. Això assegura que `"coordinades"` siga un array amb exactament dues coordenades.

- `"tipus"` és una propietat de tipus `"string"` Utilitza `"enum"` per especificar una llista de valors vàlids per aquesta propietat. Només s'accepta el valor `"Punt"`

- `"additionalProperties"` s'estableix com a `"false".` Això significa que no s'acceptaran altres propietats que no siguin `"coordinades"` o `"tipus"` a l'objecte `"geometria"`. Això restringeix l'objecte perquè només puga contenir les propietats definides i no altres propietats addicionals.

Us d'Expressions Regulars

Una expressió regular, **(regex o regexp)**, és una seqüència de caràcters que defineix un **patró de cerca**. Este patró s'utilitza principalment per a la **recerca i manipulació de cadenes de text**, ja que permet especificar regles complexes per a la identificació de subcadenes.

Les expressions regulars consten d'una combinació de caràcters normals i caràcters especials anomenats **metacaràcters**.

Els **metacaràcters** tenen significats especials i s'utilitzen per a definir patrons més complexos. Algunes operacions que es poden realitzar amb expressions regulars inclouen la cerca, substitució, validació i extreure dades d'una cadena.

Exemples de metacaràcters comuns:

- **.** : Coincideix amb qualsevol caràcter excepte un salt de línia.
- ***** : Coincideix amb zero o més repeticions del caràcter o grup anterior.
- **+** : Coincideix amb una o més repeticions del caràcter o grup anterior.
- **?** : Fa que el caràcter o grup anterior sigui opcional (zero o una repetició).
- **[ ]:** Coincideix amb qualsevol caràcter dins dels brackets.
- **^** : Coincideix amb l'inici de la línia.
- **$** : Coincideix amb el final de la línia.
- **|**: Opera com un operador "o" per a múltiples opcions.
- **:** Escapa un metacaràcter per a que es tracti com a caràcter normal.

Per exemple, l'expressió regular **\d{4}[A-Z]{3}** podria utilitzar-se per validar un número de matricula española.

Pattern

El camp "**pattern**" en JSON Schema s'utilitza per aplicar una expressió regular a una cadena de text per validar el seu format. Aquesta expressió regular defineix un patró que la cadena ha de seguir.

- ### Sintaxi Bàsica de pattern:

```
{
"type": "string",
"pattern": "expressió_regular"
}
```

- **Exemples:**

Validació de Correu Electrònic

```
{
"type": "string",
"pattern": "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"
}
```

Valida adreces de correu electrònic amb un patró comú.

Validació de Números de Telèfon

```
{
"type": "string",
"pattern": "^\\+?[1-9]\\d{1,14}$"
}
```

Valida números de telèfon amb o sense el prefix internacional.

Validació de Nom d'Usuari

```
{
"type": "string",
"pattern": "^[a-zA-Z0-9_-]{3,16}$"
}
```

Valida noms d'usuari que contenen lletres, números, guions baixos i guions.

Exemple 5: Validació de URL

```
{
"type": "string",
"pattern": "^(https?|ftp):\\/\\/[\\w\\d\\-]+(\\.[\\w\\d\\-]+)+([\\w\\d\\-
.,@?^=%&:/~+#]*[\\w\\d\\-@?^=%&/~+#])?$"
}
```

Valida URLs amb els protocols http, https o ftp.

Validació de Data (format AAAA-MM-DD)

```
{
"type": "string",
"pattern": "^\\d{4}-\\d{2}-\\d{2}$"
}
```

Valida dates amb el format "AAAA-MM-DD".

Exercicis Resolts

JSON de Base per als Exemples:

```
{
  "persona": {
    "nom": "Maria",
    "edat": 28
  },
  "colors": ["verd", "blau", "groc"],
  "numeros": [1.5, 3.8, 7.2],
  "dadesAddicionals": {
    "cert": true,
    "secreta": "12345"
  }
}
```

- **Exercici 1: Creació d'un esquema senzill**

Crea un esquema JSON que valida l'objecte amb les propietats "**persona**" i assegura't que "**nom**" és de tipus "**string**" i "**edat**" de tipus "**integer**". Totes dues són obligatòries.

Solució:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "persona": {
      "$ref": "#/definitions/Persona"
    }
  },
  "definitions": {
    "Persona": {
      "type": "object",
      "properties": {
        "nom": {
          "type": "string"
        },
        "edat": {
          "type": "integer"
        }
      },
      "required": ["nom", "edat"]
    }
  }
}
```

Vàlid	Invàlid
{ "persona": { "nom": "Carlos", "edat": 35 } }	{ "persona": { "nom": "Laura" } }

- **Exercici 2: Restricció de propietats addicionals**

Amplia l'esquema anterior per permetre propietats addicionals que siguin de tipus "**boolean**" a l'objecte "**dadesAddicionals**".

Solució:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "persona": {
      "$ref": "#/definitions/Persona"
    },
    "dadesAddicionals": {
      "type": "object",
      "additionalProperties": {
        "type": "boolean"
      }
    }
  },
  "definitions": {
    "Persona": {
      "type": "object",
      "properties": {
        "nom": {
          "type": "string"
        },
        "edat": {
          "type": "integer"
        }
      },
      "required": ["nom", "edat"]
    }
  }
}
```

Vàlid	Invàlid
`{ "persona": { "nom": "Ana", "edat": 28 }, "dadesAddicionals": { "cert": true, "secret": false } }`	`{ "persona": { "nom": "Pau", "edat": 42 }, "dadesAddicionals": { "secret": "xyz" } }`

- **Exercici 3: Restricció de valors per una propietat**

Crea un esquema que valida l'objecte amb la propietat "colors," la qual només pot tenir els valors "verd," "blau" o "groc".

Solució:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "colors": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["verd", "blau", "groc"]
      }
    }
  },
  "required": ["colors"]
}
```

Vàlid	Invàlid
{ "colors": ["verd", "blau"] }	{ "colors": ["roig", "groc"] }

- **Exercici 4: Arrays amb restriccions**

Defineix un esquema que valida arrays amb números decimals a la propietat "numeros." Assegura't que hi hagi com a mínim 2 elements a l'array.

Solució:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "properties": {
    "numeros": {
      "type": "array",
      "items": {
        "type": "number"
      },
      "minItems": 2
    }
  },
  "required": ["numeros"]
}
```

Vàlid	Invàlid
{ "numeros": [3.14, 2.5, 7.0] }	{ "numeros": [1, "dos"] }

- **Exercici 5: Referència a un esquema extern**

Utilitza la propietat "$ref" per referenciar l'esquema creat a l'exercici 1. Assegura't que l'objecte compleixi amb aquest esquema.

Solució:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$ref": "#/definitions/Persona",
  "definitions": {
    "Persona": {
      "type": "object",
      "properties": {
        "nom": {
          "type": "string"
        },
        "edat": {
          "type": "integer"
        }
      },
      "required": ["nom", "edat"]
    }
  }
}
```

Vàlid	Invàlid
{ "nom": "Eva", "edat": 30 }	{ "nom": "Pablo" }

Exercicis:

- **Exercici 1: Validació de Propietats de Color**

Crea un esquema que valida un objecte amb propietats de color. Les propietats han de ser "colorPrincipal" (cadena de text), "tons" (array de cadenes de text) i opcionalment "hex" (cadena de text amb format hexadecimal).

Regex per a la cadena exadecimal: `"^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})?$"`

- **Exercici 2: Validació de Configuració d'Aplicació**

Defineix un esquema que valida la configuració d'una aplicació. L'objecte ha de contenir les propietats "idioma" (cadena de text), "mode" (cadena de text amb opcions "normal" o "dark"), i opcionalment "notificacions" (boolean).

- **Exercici 3: Restricció d'Elements a una Llista**

Crea un esquema que valida una llista d'elements, on cada element ha de ser un objecte amb les propietats "nom" (cadena de text) i "puntuacio" (número enter). Assegura't que la llista tinga com a mínim 3 elements.

- **Exercici 4: Validació d'Informació de Producte**

Defineix un esquema que valida la informació d'un producte. L'objecte ha de contenir les propietats "nom" (cadena de text), "preu" (número decimal) i "disponible" (boolean). La propietat "nom" és obligatòria.

- **Exercici 5: Referència a un Esquema Extern**

Utilitza un esquema extern per validar una llista d'usuaris. L'esquema de l'usuari ha de contenir les propietats "nom" (cadena de text) i "email" (cadena de text amb format d'email). La llista ha de tenir com a mínim 2 usuaris.