DEFINIÇÃO DE ONTOLOGIAS EM SOFIA2

JULHO 2014

VERSÃO 4

1 ÍNDICE

2 INTRODUÇÃO

2.1 Objetivos e âmbito do presente documento

O objetivo deste documento é explicar de forma clara e simples como se definem as ontologias na Plataforma.

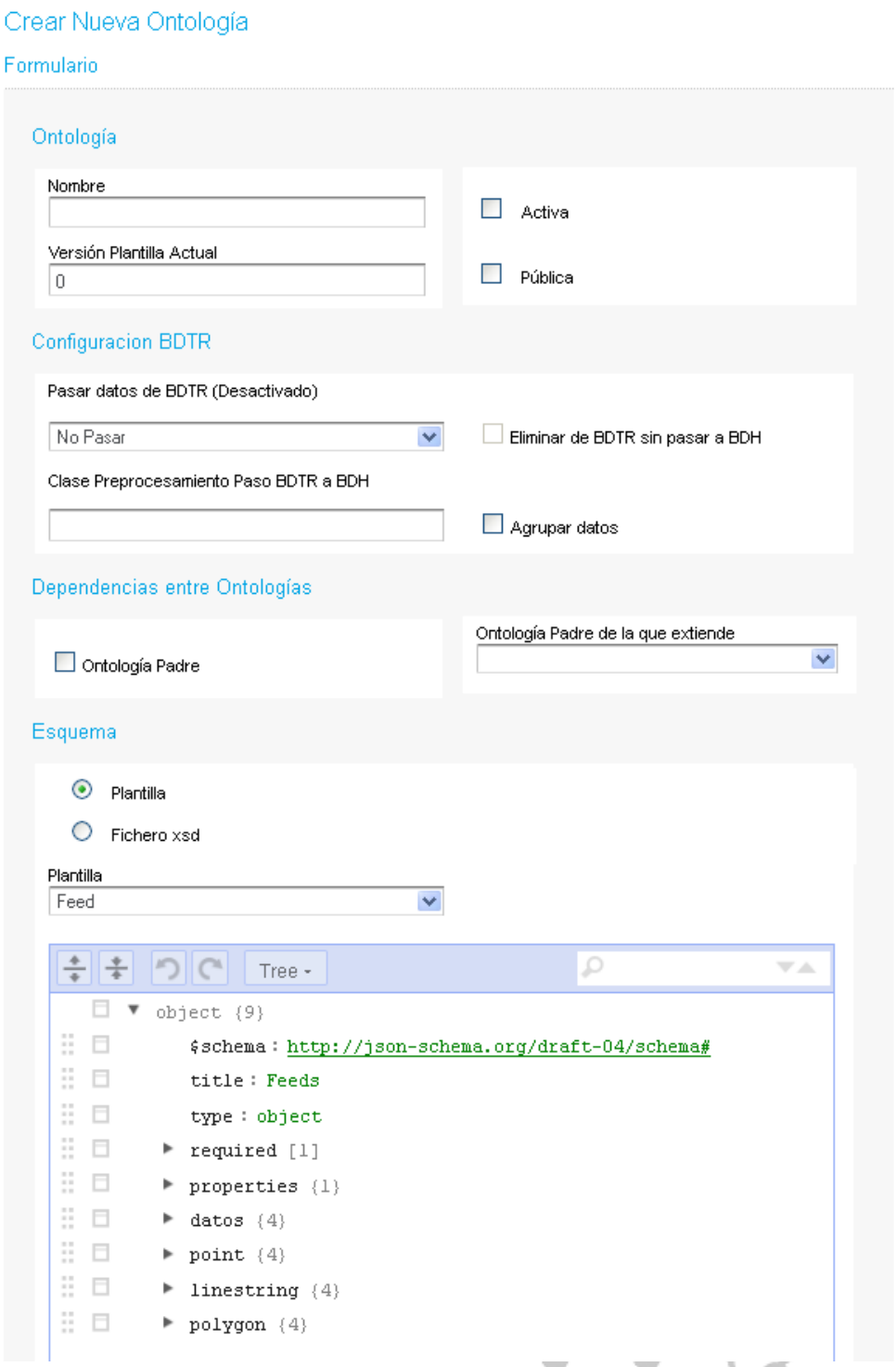
3 CONCEITOS BÁSICOS

- **JSON**: é um formato ligeiro para o intercâmbio de dados (como o XML mas menos verboso)
- **JSON-Schema:** um esquema JSON é um documento JSON, que permite especificar como é um documento JSON ao qual se refere (se existem atributos obrigatórios, se são do tipo *number*, se podem ser nulos). Numa equivalência, XML corresponderia a um esquema XML ou um DTD.
- **Ontologia** define formalmente um conjunto comum de termos que se utilizam para descrever e representar um domínio.
- **Ontologia SOFIA2:** Em SOFIA2, uma ontologia é a definição do conjunto de classes e atributos das mesmas que partilharão as distintas aplicações que interoperam dentro do SmartSpace. Em SOFIA2, as ontologias definem-se em JSON conforme um esquema JSON.
- **Instância de Ontologia:** é um elemento concreto de uma ontologia

4 VISÃO GERAL

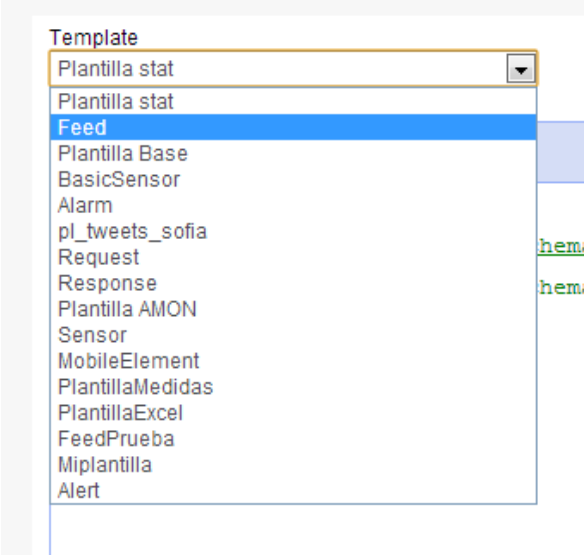
Conforme referido, em SOFIA2 uma **ontologia** representa uma entidade no sistema (SmartSpace), e esta define-se em **JSON**.

A Plataforma oferece uma **Web (+API REST) de Configuração** na qual os utilizadores com permissões (colaboradores e administradores) podem criar as suas ontologias.



Uma ontologia define-se a partir de um **esquema JSON**.

Para simplificar a criação de Ontologias, a Plataforma oferece o conceito de modelos, que são esquemas JSON pré-carregados, que o utilizador pode utilizar e ampliar para criar as suas ontologias:

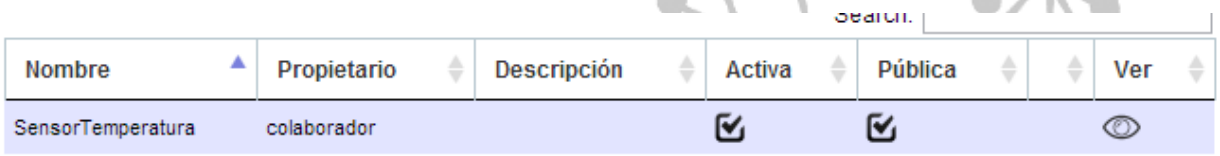


Podemos começar com a definição de uma Ontologia simples, como a que representa um Sensor de Temperatura que armazena: identificador, timestamp, medida, unidade e coordenadas GPS.

Uma instância desta ontologia teria o seguinte aspecto:

```
{
 "SensorTemperatura": {
  "identificador":"ST-TA3231-1",
  "timestamp":{"$date": "2014-01-27T11:14:00Z"},
  "medida":25.1,
  "unidad":"C",
   "geometry":{
     "type": "Point",
     "coordinates":[90,-10.1]
    }
 }
```

Pode verificar como se define o esquema JSON desta ontologia em SOFIA2:



Esta ontologia é pública, sendo assim qualquer pessoa pode consultar os dados da mesma.

Se carregarmos em Ver  será possível verificar o esquema JSON que descreve esta Ontologia (nos pontos a seguir entraremos em detalhe sobre a sintaxe deste esquema):


```
{"$schema":"http://json-schema.org/draft-04/schema#",
  "title":"SensorTemperatura Schema",
  "type":"object",
  "required":["SensorTemperatura"],
  "properties":{
    "_id":{
      "type":"object",
      "$ref":"#/identificador"
    },
    "SensorTemperatura":{
      "type":"string",
      "$ref":"#/datos"}
   },
   "additionalProperties":false,
   "identificador":{
      "title":"id",
      "description":"Id insertado del SensorTemperatura",
      "type":"object",
      "properties":{
        "$oid":{
          "type":"string"
        }
      },
      "additionalProperties":false
    },
  "datos":{
    "title":"datos",
    "description":"Info SensorTemperatura",
    "type":"object",
    "required":["identificador","timestamp","medida","unidad","coordenadaGps"],
    "properties":{
      "identificador":{
        "type":"string"
      },
      "timestamp":{
          "type":"object",
          "required":["$date"],
          "properties":{
            "$date":{
              "type":"string",
              "format":"date-time"
            }
          },
          "additionalProperties":false
```

```
        },
      "medida":{
        "type":"number"
      },
      "unidad":{
        "type":"string"},
      "geometry":{
        "$ref":"#/gps"
      }
    },
    "additionalProperties":false
    },
    "gps":{
      "title":"gps",
      "description":"Gps SensorTemperatura",
      "type":"object",
       "required":["coordinates","type"],
       "properties":{
            "coordinates":{
              "type":"array",
              "items":[
               {
                 "type":"number",
                “maximum”:180,
                 “mininum”:-180
               },
               {
                 "type":"number",
                “maximum”:180,
                 “mininum”:-180
               }
              ],
              "minItems":2,
              "maxItems":2
            },
            "type":{
              "type":"string",
              "enum":["Point"]
            }
        },
        "additionalProperties":false
      }
```

NOTA

No esquema podemos observar que as propriedades ***timestamp*** e ***geometry*** estão definidas de uma forma especial para permitir efetuar pesquisas por data e geoespaciais em MongoDB. Vejamos cada caso em particular:

- **timestamp**: MongoDB permite trabalhar com datas em formato ***ISO 8601*** data (**YYYY-MM-DDThh:mm:ss.fffZ).** Para que MongoDB interprete um campo do tipo data, este espera receber um JSON com uma estrutura similar a {"$date": "2014-01-27T11:14:00Z"}. O esquema que nos permite validar instâncias deste tipo é o seguinte:

```
{   "timestamp":{
            "type":"object",
            "required":["$date"],
            "properties":{
                 "$date":{
                      "type":"string",
                      "format":"date-time"
                    }
              },
              "additionalProperties": false
          }
   }
```

Isto permite-nos efetuar consultas como a seguinte:

```
db.SensorTemperatura.find({"Sensor.created":{"$lt": new ISODate()}});
```

- **geometry:** MongoDB permite realizar consultas geoespaciais. Para isso, requer que os campos a tratar para este fim sejam definidos, com a seguinte estrutura:

```
{  "geometry": {
        "type": "object",
        "required":["coordinates","type"],
        "properties":{
              "coordinates":{
                    "type":"array",
                    "items":[
                          {
                            "type":"number",
                            "maximum": 90,
                            "minimum": -90
                          },
                          {
```

```
                                "type":"number",
                                "maximum": 180,
                                "minimum": -180
                            }
                        ],
                        "minItems":2,
                        "maxItems":2
                    },
                    "type":{
                        "type":"string",
                        "enum":["Point"]
                    }
                },
                "additionalProperties":false
            }
```

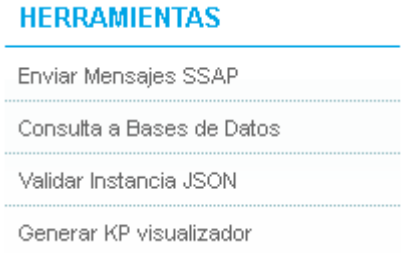
A propriedade *geometry* está composta pelo tipo "Point" e por umas coordenadas, que representam um ponto, dado pela latitude e longitude ("coordenates":[Latitud,Longitud]). O intervalo de valores que suporta MongoDB para este tipo de coordenadas situa-se entre [90, -90] para a latitude e [180,-180] para a longitude. Se tentar inserir um valor fora do intervalo, MongoDB retorna um erro.

Uma instância que cumpre esta estrutura: {"geometry ": {"type":"Point", "coordinates":[1.9, -3.9]}}

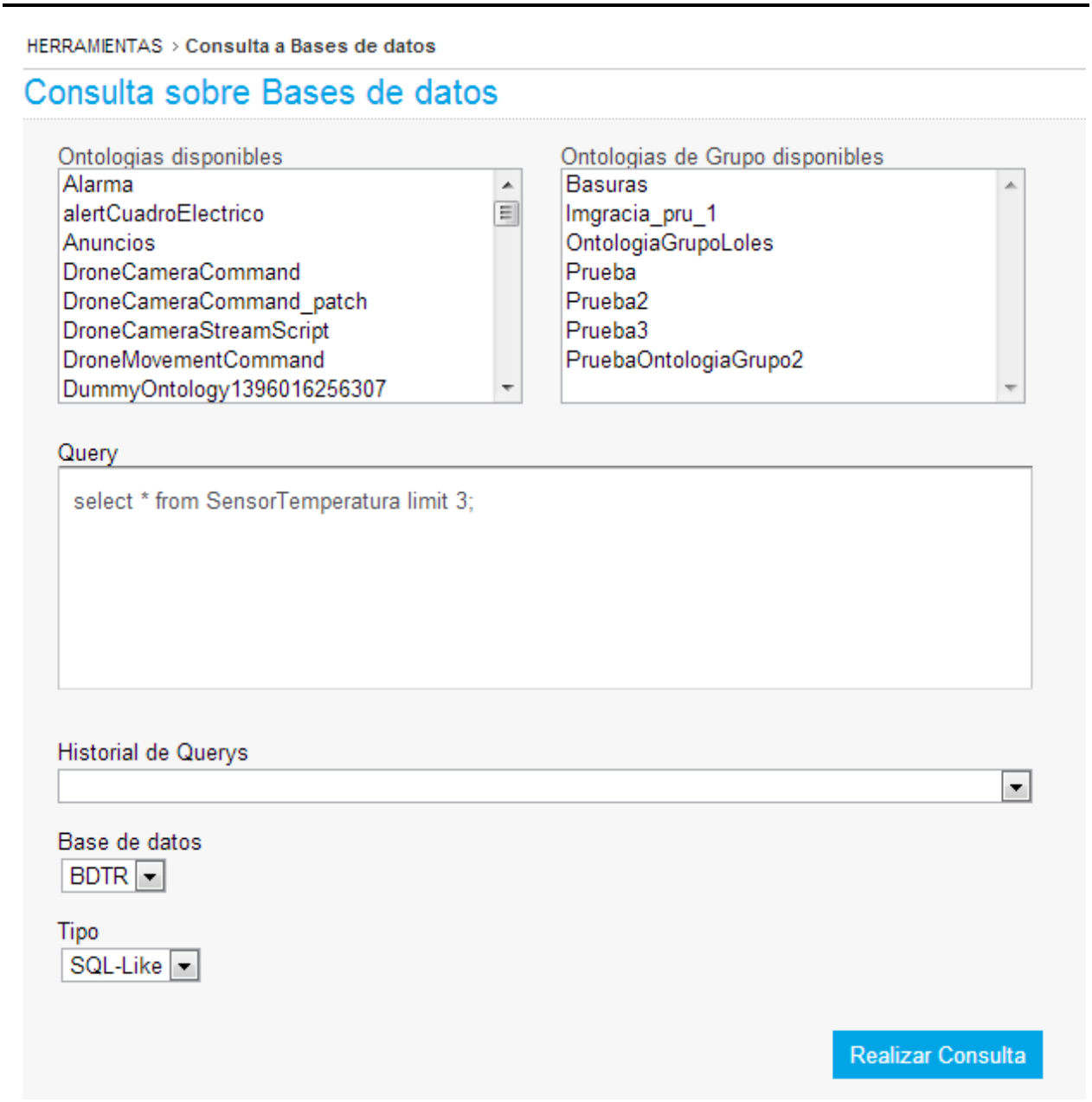
Poderemos efetuar pesquisas geoespaciais em MongoDB como a seguinte:

```
db.SensorTemperatura.find({"Sensor.geometry.coordinates":{$near:[12,12],$maxDistance:1}})
```

Pode visualizar as **instâncias das minhas ontologias,** a partir da Web de Configuração, através da opção Consulta da Base de Dados:



Se nesta consulta executou uma consulta deste estilo:



Verá a información da última instância inserida na BDTR de SOFIA2.


```
{
  "_id": {
    "$oid": "51e3dbd465701fd8e0f69828"
  },
  "contextData": {
    "session_key": "08bf50c8-6ea6-41dc-99ac-5d12a6f517a3",
    "user_id": 1,
    "kp_id": 9,
    "kp_identificador": "gatewaysensores",
    "timestamp": {"$date": "2014-01-27T11:14:00Z"}
  },
  {
```

```
"SensorTemperatura": {
  "identificador":"ST-TA3231-1",
  "timestamp":{"$date": "2014-01-27T11:14:00Z"},
  "medida":25.1,
  "unidad":"C",
   "geometry":{
     "type": "Point",
     "coordinates":[90,-10.1]
    }
 }
}
```

Observamos que a información devolvida inclui:

- O **identificador** dessa instância:

```
"_id": {
  "$oid": "52794144abc7d0b77e686c8b"
}
```

- **Informação de contexto:** como o KP, instância, utilizador, sessão e data em que se inseriu.

```
"contextData": {
 "session_key": "08bf50c8-6ea6-41dc-99ac-5d12a6f517a3",
 "user_id": 1,
 "kp_id": 9,
 "kp_identificador":  "gatewaysensores",
 "timestamp": {"$date": "2014-01-27T11:14:00Z"}
},
```

- **Instância da Ontologia**

```
"SensorTemperatura": {
 "identificador":"ST-TA3231-1",
 "timestamp":{"$date": "2014-01-27T11:14:00Z"},
 "medida":25.1,
 "unidad":"C",
  "geometry":{
    "type": "Point",
    "coordinates":[110.2,1233.1]
   }
}
```

5 TECNOLOGIAS IMPLICADAS

5.1 JSON

JSON é o acrónimo de JavaScript Object Notation.

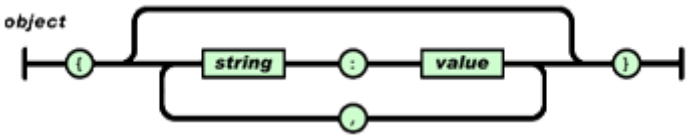
JSON é um formato ligeiro originalmente concebido para o intercâmbio de dados na Internet.

5.1.1 Tipos de dados

- **string** : Cadeia de texto
- **number**: Numérico
- **object**: Objeto
- **char**: Caracteres Unicode válidos
- **array**: Coleção de valores
- **null**: Nulo
- **boolean**: Valores *true* ou *false*

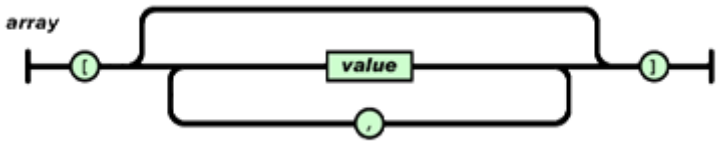
Em JSON, apresentam-se destas formas:

Um **objeto** é um conjunto, sem ordenação de pares chave-valor. Começa por "{" e termina com "}". Cada nome será seguido por ":", os pares chave-valor estarão separados por ",".



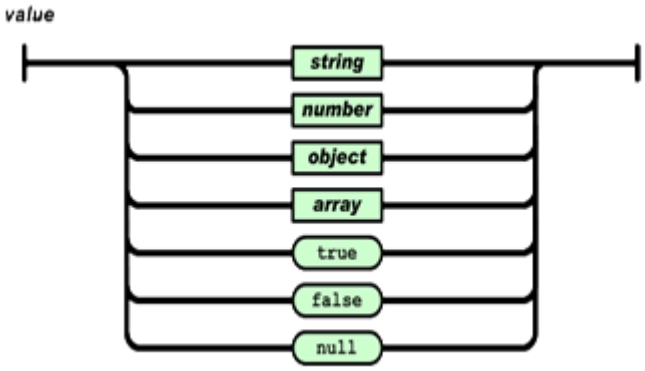

```
{
  "address" : {
    "line1" : "555 Main Street",
    "city" : "Denver",
    "stateOrProvince" : "CO",
    "zipOrPostalCode" : "80202",
    "country" : "USA"
  }
}
```

Um **array** é uma coleção de valores. Começa por "[" e acaba com "]". Os valores separam-se por ",".

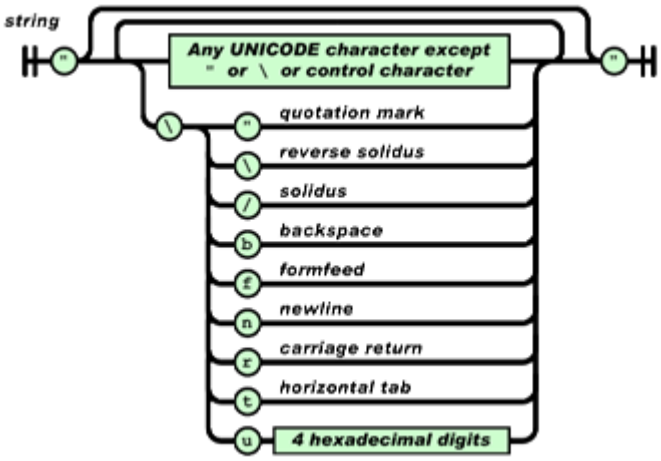



```
{
  "people" : [
    { "firstName": "John", "lastName": "Smith", "age": 35 },
    { "firstName": "Jane", "lastName": "Smith", "age": 32 }
  ]
}
```

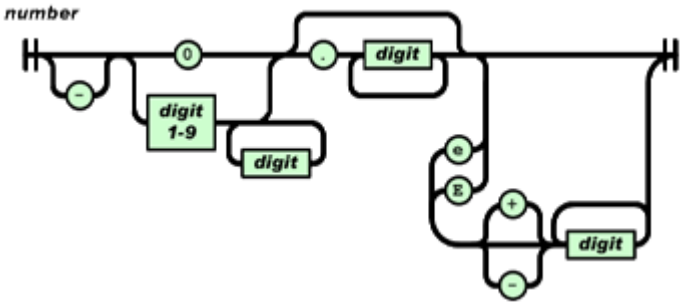
Um **valor** pode ser uma cadeia de caracteres com aspas, um número, *true*, *false*, *null*, um objeto ou um array. Estas estruturas podem ser acumuladas:



Uma **string** é uma sequência de zero ou mais caracteres Unicode, fechados entre aspas (“)



Um **número** é como um número decimal em Java.



5.1.2 Referências

http://cdn.dzone.com/sites/all/files/refcardz/rc173-010d-JSON_2.pdf

5.2 Esquemas JSON (JSON-SChema)

JSON-Schema (**http://json-schema.org**) é um formato JSON para descrever dados em JSON. JSON é idêntico a XSD para o XML. Oferece um contrato para definir os dados exigidos para uma determinada aplicação e a forma de interatuar com a mesma.

5.2.1 Exemplo

Para termos uma ideia, vejamos um exemplo de um esquema JSON simples:

```
{
    "$schema": "http://json-schema.org/draft-04/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",
            "minimum": 0,
            "exclusiveMinimum": true
        }
    },
    "required": ["id", "name", "price"]
}
```

Que efetua a validação de JSONs como a seguinte:

```
{
```

```
    "id": 1,
    "name": "A green door",
    "price": 12.50,
    "tags": ["home", "green"]
}
```

E este como inválido por não ter o atributo price:

```
{
    "id": 1,
    "name": "A green door",
    "tags": ["home", "green"]
}
```

5.2.2 Atributos de um esquema JSON

Podemos ver a referência completa da especificação JSON aqui: **http://json-schema.org/latest/json-schema-core.html**

```
    {
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Product",
    "description": "A product from Acme's catalog",
    "type": "object",
    "properties": {
        …
        …
        …
        …
    },
    "required": ["id", "name", "price"]
}
```

Os atributos mais utilizados num esquema JSON são:

- ***"$schema"***: Permite-nos indicar a versão do Schema JSON que pretendemos utilizar: 0.4 ou 0.3, SOFIA2 que se apoia na versão 0.4 (*http://json-schema.org/draft-04/schema#*).
- **"title"**: indicar um título com o qual pretende identificar o esquema.
- **"description"**: Pode utilizar-se este atributo para incluir uma descrição daquilo que vai representar o esquema JSON.
- **"type"**: Para indicar o tipo que representará o esquema.
- **"properties"**: Este atributo é um objeto com as definições de propriedades que definem os valores estáticos de uma instância de objeto. É uma lista não ordenada de

propriedades. Os nomes das propriedades devem cumprir-se e o valor das propriedades define-se a partir de um esquema, que deve igualmente cumprir-se.

- **"patternProperties"**: Este atributo é um objeto com as definições de propriedades que definem os valores de uma instância de objeto. É uma lista desordenada de propriedades. Os nomes das propriedades são padrões de expressões regulares, as instâncias das propriedades devem cumprir o padrão definido e o valor da propriedade com o esquema, que define essa propriedade.
- **"additionalProperties"**: Permite indicar se a instância JSON pode conter propriedades que não tenham sido definidas no esquema. Tem dois valores possíveis (*true* ou *false*), para indicar se se admite qualquer propriedade ou não. Se não adicionar a propriedade, poder-se-á incluir qualquer outra propriedade.
- **"required"**: Permite indicar todas as propriedades que são obrigatórias para uma instância JSON e que deve incluir como mínimo. As propriedades incluir-se-ão entre parênteses rectos e separadas pelo carácter ",".

  (É obrigatório incluir esta propriedade no esquema).
- **"$ref"**: Define um URI de um esquema, que contém a representação completa para essa propriedade.

Vejamos neste extrato do esquema um exemplo para os atributos definidos:

```
{
  "$schema":"http://json-schema.org/draft-04/schema#",
  "title":"SensorTemperatura Schema",
  "type":"object",

  "required":["SensorTemperatura"],

  "properties":{
    "_id":{
      "type":"object",
      "$ref":"#/identificador"
    },
    "SensorTemperatura":{
      "type":"string",
      "$ref":"#/datos"}
  },
  "additionalProperties":false,
```

Neste exemplo, podemos verificar que há uma propriedade que é obrigatória "**SensorTemperatura**" e que existem duas propriedades **"_id"** e "**SensorTemperatura**", que

incluem uma referência a um elemento, que contém a representação completa dessa propriedade.

```
"identificador":{
      "title":"id",
      "description":"Id insertado del SensorTemperatura",
      "type":"object",
      "properties":{
        "$oid":{
          "type":"string"
        }
      },
      "additionalProperties":false
    },

"datos":{

    "title":"datos",
    "description":"Info SensorTemperatura",
    "type":"object",
    "required":["identificador","timestamp","medida","unidad","coordenadaGps"],
    "properties":{
      "identificador":{
        "type":"string"
      },
      "timestamp":{
            "type":"object",
            "required":["$date"],
            "properties":{
              "$date":{
                "type":"string",
                "format":"date-time"
              }
            },
            "additionalProperties":false
            },
      "medida":{
        "type":"number"
      },
      "unidad":{
        "type":"string"},
      "geometry":{
        "$ref":"#/gps"
      }
    },
  "additionalProperties":false
```

Como podemos verificar, quer o "identificador" assim como os "dados" são esquemas que definem a sua representação. Podemos verificar ainda que não se admite nenhum tipo de propriedade que não seja definida (incluiu-se "additionalProperties").

- **Listas**: As listas serão representadas entre parênteses rectos e separadas entre o carácter ",". As listas são sempre de tipo "string". Por exemplo, se queremos definir uma propriedade chamada "tipo" que apenas pode ter um dos dois valores "latitude" ou "longitude", esta teria o seguinte aspecto:

```
"tipo":{
        "type":"string",
        "enum":["latitud","longitud"]
}
```

  Para instanciá-lo, *"tipo": "latitude"*

- **"items":** Define os elementos permitidos num array, deve ser um esquema ou um conjunto de esquemas.
- **"additonalItems":** Para indicar se se admitem elementos no array, para além dos definidos no esquema.
- **"minItems":** Número mínimo de elementos que pode ter o array.
- **"maxItems":** Número máximo de elementos que pode ter o array.

No seguinte exemplo, podemos ver como é o esquema para um array, "coordinates", que deve ser de tipo numérico e que pode apenas ter dois elementos. Também verificamos que a propriedade "type" é uma lista com um único valor possível "Point".

```
"geometry":{
    "type": "object",
    "required":["coordinates","type"],
    "properties":{
        "coordinates":{
            "type":"array",
            "items":{
                "type":"number"
            },
            "minItems":2,
            "maxItems":2
        },
        "type":{
            "type":"string",
            "enum":["Point"]
```

```
            }
    },
      "additionalProperties":false
}
```

Uma instância para este objeto teria o seguinte aspecto:

```
"geometry":{
     "type": "Point",
     "coordinates":[110.2,1233.1]
  }
```

Podemos encontrar mais informação e exemplos no link seguinte: **http://json-schema.org/**