

RESTFul API

História

Os desenvolvedores de *front-end* da plataforma IoT proposta (Web, iOS e Android) precisam acessar uma *RESTful API* que suporte as operações que o usuário é capaz de desempenhar nos aplicativos.

É esperado que através dos apps o usuário possa **visualizar** sensores e *streams* ativos, **registrar** novos sensores e **vincular/registrar** novas streams aos mesmos. Para as streams, é possível **consultar** o fluxo de dados que vai sendo publicado pelos sensores reais ao longo do tempo.

É esperado que a *RESTful API* permita interagir com duas entidades principais **sensor** e **stream** através de mensagens em **JSON**. Foi definido um conjunto de requisitos mínimos que a API deve atender bem como determinados os formatos das mensagens de cada requisição, como listadas no fim do documento.

Objetivo

Implementar um serviço de **backend** (com um modelo e persistência de dados) que provê uma RESTFul API para os desenvolvedores de *front-end* atendendo as necessidades em anexo.

Implementação do Serviço

À implementação do serviço não impõe restrições tecnológicas, nem sobre banco de dados nem sobre linguagem/framework para desenvolvimento, algumas sugestões são:

- Express, Loopback, Restify (JS),
- Play Framework, Spark Framework, Restlet (Java)
- Django, Flask (Python)
- Ruby on Rails (Ruby)

Ferramentas

- Postman

Entregável

O candidato deve apresentar o serviço implementado e demonstrar o funcionamento dos *endpoints*

sugeridos em anexo, bem como apresentar a maneira como a base foi modelada.

Definição da API

[GET] Consultar unidades de grandeza (`unit`)

`<= response`

```
[
  {
    "oid": 1,
    "label": "°C"
  },
  {
    "oid": 2,
    "label": "mg/m³"
  },
  {
    "oid": 3,
    "label": "hPA"
  },
  {
    "oid": 4,
    "label": "lux"
  },
  {
    "oid": 5,
    "label": "%"
  }
]
```

[GET] Consultar Sensores (`sensor`) de um Usuário (`user`)

`<= response`

```
[
  {
```

```
"oid": 1,
"key": "10dd35008a0f4d838c3dc22856660928",
"label": "sensor 001",
"description": "Isaac's Room control",
"streams": [
  {
    "oid": 1,
    "key": "b4ea3ba494644200b679ac593f55cb87",
    "label": "temperature",
    "unit": 1,
    "sensor": 1,
    "totalSize": 84
  },
  {
    "oid": 3,
    "key": "ae194d2b61e0496fbf601f9edcf8b0c5",
    "name": "humidity",
    "unit": 5,
    "sensor": 1,
    "totalSize": 6
  },
  {
    "oid": 4,
    "key": "3170f851fd9045ed99e5d86ababdb80e",
    "label": "carbon dioxide",
    "unit": 2,
    "sensor": 1,
    "totalSize": 7
  }
],
{
  "oid": 2,
  "key": "27b26e48cd674cc38ec45808cf48fa07",    "label": "Kitchen's freezer sensor
(Arduino)",
  "description": "Kitchen's freezer sensor (Arduino)",
```

```
"streams": [  
  {  
    "oid": 2,  
    "key": "8961bd9a4d1e439ebf3b86af5b9d5c1f",  
    "label": "temperature",  
    "unit": 1,  
    "sensor": 2,  
    "totalSize": 19  
  }  
]  
}]
```

[GET] Consulta de um Sensor (`sensor`) específico. Ex: `key: 27b26e48cd674cc38ec45808cf48fa07`

`<= response`

```
{  
  "oid": 2,  
  "key": "27b26e48cd674cc38ec45808cf48fa07",    "label": "Kitchen's freezer sensor  
(Arduino)",  
  "description": "Kitchen's freezer sensor (Arduino)",  
  "streams": [  
    {  
      "oid": 2,  
      "key": "8961bd9a4d1e439ebf3b86af5b9d5c1f",  
      "label": "temperature",  
      "unit": 1,  
      "sensor": 2,  
      "totalSize": 19,  
      "data": [  
        {  
          "timestamp": 1506455591,  
          "data": -6.56  
        },  
        {  
          "timestamp": 1506455566,
```

```
        "data": -6.54
      },
      {
        "timestamp": 1506455551,
        "data": -6.56
      },
      {
        "timestamp": 1506455530,
        "data": -6.55
      },
      {
        "timestamp": 1506455510,
        "data": -6.56
      },
    ]
  }
}
```

obs: a consulta individual de um sensor retorna adicionalmente os 5 dados mais recentes de cada stream que o mesmo possui.

[GET] Consulta de dados de um Stream (`stream`) específico. Ex:

`key: 8961bd9a4d1e439ebf3b86af5b9d5c1f`

`<= response`

```
{
  "oid": 2,
  "key": "8961bd9a4d1e439ebf3b86af5b9d5c1f",
  "label": "temperature",
  "unit": 1,
  "sensor": 2,
  "totalSize": 19,
  "data": [
    {
      "timestamp": 1506455591,
```

```
    "data": -6.56
  },
  {
    "timestamp": 1506455566,
    "data": -6.54
  },
  {
    "timestamp": 1506455551,
    "data": -6.56
  },
  {
    "timestamp": 1506455530,
    "data": -6.55
  },
  {
    "timestamp": 1506455510,
    "data": -6.56
  },
  ...
]}
```

[POST] Registrar Sensor (`sensor`)

`request =>`

```
{
  "label": "Kitchen's freezer sensor (Arduino)", "description": "Kitchen's freezer sensor
(Arduino)",}
```

`<= response`

```
{
  "oid": 2,
  "key": "8961bd9a4d1e439ebf3b86af5b9d5c1f",
  "label": "Kitchen's freezer sensor (Arduino)", "description": "Kitchen's freezer sensor (Arduino)",}
```

obs: `oid` (identificador interno) e `key` (chave de identificação) são atribuídos no momento do registro do *sensor* pelo serviço.

[POST] Registrar Stream (`stream`) para Sensor (`sensor`). Ex:

key: 27b26e48cd674cc38ec45808cf48fa07

request =>

```
{
  "label": "temperature",
  "unit": 1}
```

<= response

```
{
  "oid": 2,
  "key": "8961bd9a4d1e439ebf3b86af5b9d5c1f",
  "label": "temperature",
  "unit": 1,
  "sensor": 2,
  "totalSize": 0}
```

obs: `oid` (identificador interno) e `key` (chave de identificação) são atribuídos no momento do registro da *stream* pelo serviço.

[POST] Publicar dado em um Stream (`stream`). Ex: **key: 8961bd9a4d1e439ebf3b86af5b9d5c1f**

request =>

```
{  
  "timestamp": 1506521102,  
  "value": 28.5}
```

*
<= response

```
{  
  "oid": 123,  
  "timestamp": 1506521102,  
  "value": 28.5}
```

obs: `oid` (identificador interno) e `key` (chave de identificação) são atribuídos no momento do registro da *stream* pelo serviço.

Revision #1

Created Wed, Oct 11, 2017 2:44 PM by Isaac Pereira

Updated Wed, Oct 11, 2017 2:44 PM by Isaac Pereira