



UNIVERSIDADE KATYAVALA BWILA
INSTITUTO POLITÉCNICO

**MESTRADO EM
ENGENHARIA INFORMÁTICA**

TP2 – FIWARE IoT MIDDLEWARE – PROPOSTA

I). Objectivos

- o Instalar e iniciar o ambiente FIWARE via Docker Compose;
- o Criar entidades e dispositivos IoT simulados (ex: salas com sensores de temperatura e humidade);
- o Registrar dispositivos com o IoT Agent (IDAS);
- o Enviar dados simulados para o Context Broker (Orion);
- o Criar subscrições para aplicações externas;
- o Persistir dados históricos com o Cygnus em base de dados MySQL.

II). Pré-Requisitos

- o Conhecimentos introdutórios sobre:
 - o Conceitos básicos de redes e HTTP;
 - o Lógica de programação e APIs REST;
 - o Conceitos de sensores e IoT
- Ferramentas instaladas no computador:
 - o VirtualBox ou VMWare (opcional para quem preferir instalar Docker e Docker Compose).
 - o Docker e Docker Compose;
 - o Postman (ou equivalente);
 - o Cliente SSH (como PuTTY ou terminal nativo);
 - o Editor de código (VSCode, Sublime, etc.);

III). Materiais necessários

- o Computador com VM Ubuntu/Linux do Fiware Pré-configurado ou Docker e Docker Compose instalados;
- o Acesso à internet;
- o Repositório com o ambiente FIWARE (Docker Compose e *scripts json*: pode ser clonado de <https://github.com/mei-ip-ukb/fiware-tp.git>;

IV). **Duração e organização**

- o Até 120 minutos de trabalho individual prévio
- o Até 120 minutos em laboratório/sessão prática
- o Trabalho autónomo suplementar: pelo menos 240 minutos

V). **Introdução**

O FIWARE é uma plataforma de código aberto que oferece módulos (Generic Enablers) para construção de aplicações inteligentes, incluindo suporte a IoT, big data e serviços contextuais. Neste laboratório, é utilizado um cenário de IoT onde simulamos o envio de dados de sensores virtuais (por exemplo, temperatura e humidade numa sala) para o Context Broker (Orion), usando o protocolo UL2.0 via IoT Agent.

O ambiente é iniciado via docker-compose, onde todos os containers (MongoDB, Orion, IDAS, Cygnus, MySQL e Grafana) são levantados automaticamente.

Durante este guião, os estudantes irão:

- o Criar entidades como "Room1" com atributos *temperature* e *humidity*;
- o Registrar dispositivos como sensor-a87020747f via IoT Agent;
- o Simular envio de dados com o Postman (em vez de curl);
- o Criar subscrições em Orion para persistir os dados no MySQL com Cygnus;
- o **Aceder ao Grafana e configurar um painel de visualização para as medições recebidas.**

VI). **Actividades**

Instalação

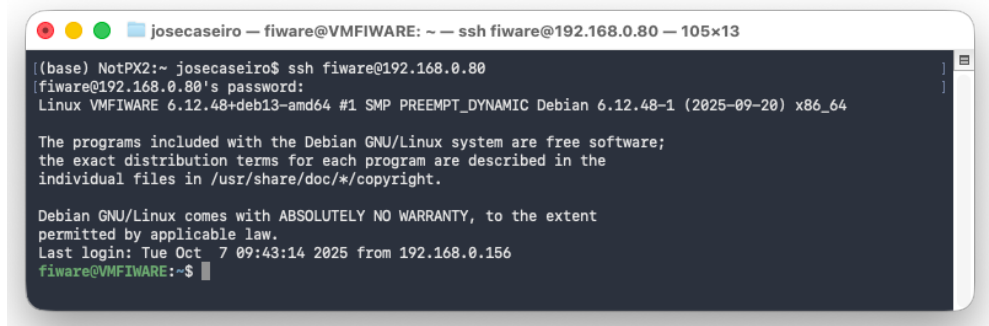
A instalação pode ser feita usando o *appliance* VMFIWARE.ova no Virtual Box, VMWare ou ainda pela instalação do Docker e Docker Compose no computador. Vamos explorar as duas formas.

a) Via VirtualBox ou VMWare

Após instalar o VirtualBox (ou outro software de virtualização) e descarregar o VMFIWARE.ova, precisamos importar o appliance. No VirtualBox siga as etapas:

1. Abra o VirtualBox, vá em "Player->File->Open", selecione o caminho do VMFIWARE.ova e clique em "Import";
2. Aguarde alguns segundos ou minutos e após o término do processo de importação "Ligue" a máquina virtual;

- Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). [+ infos](#)
3. Aguarde a inicialização do Debian e depois faça login com as seguintes credenciais:
 - o Username: fiware
 - o Password: fiware
 4. Obtenha com o comando "**ifconfig**" o endereço IP da máquina virtual para acesso via SSH (o endereço IP geralmente começa com 192.168.x.x). Tenha em atenção que as definições de rede da VM no Virtual Box precisa estar com a placa *Bridger* para receber o IP da rede;



```
josecaseiro — fiware@VMFIWARE: ~ — ssh fiware@192.168.0.80 — 105x13
((base) NotPX2:~ josecaseiro$ ssh fiware@192.168.0.80
fiware@192.168.0.80's password:
Linux VMFIWARE 6.12.48+deb13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.12.48-1 (2025-09-20) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 7 09:43:14 2025 from 192.168.0.156
fiware@VMFIWARE:~$
```

5. No terminal execute o comando **cd /opt/fiware-docker/** para aceder a pasta com o docker-composer.yml e siga os passos para inicialização do ambiente.

a) Via Docker e Docker Compose

Após instalar o Docker e Docker Compose em teu computador, abre a aplicação Docker e deixe a funcionar. Siga os seguintes passos para inicializar o ambiente Fiware:

1. Clone o repositório ou descarregue o ficheiro docker-compose.yml para uma pasta específica;
2. Abre o terminal e navegue até a pasta onde salvou o ficheiro docker-compose.yml usando o comando **cd**. Siga os passos para a inicialização do Fiware.

Inicialização do Fiware

- o Estando com terminal na pasta que contém o ficheiro docker-compose.yml (para VM em /opt/fiware-docker/ ou o local onde guardou o ficheiro docker-compose.yml) siga os passos:
 1. Execute o comando "**docker-compose up -d**" para iniciar todos os *containers* Fiware;
 2. Verifique os módulos em execução usando "**docker ps**", você verá todos os módulos em execução e algumas informações detalhadas (por

Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). [+ infos](#) exemplo, portas usadas);

3. Para parar todos os fiwaremodules use "docker-compose stop";

```
josecaseiro — fiware@VMFIWARE: /opt/fiware-docker — ssh fiware@192.168.0.80 — 181x10
fiware@VMFIWARE:/opt/fiware-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
37021105d705   fiware/orion   "/usr/bin/contextBro..." About a minute Up 54 seconds (health: starting) 0.0.0.0:1026->1026/tcp, :::1026->1026/tcp orion
c44ff1da9e3b   fiware/cygnus-ngsi "/cygnus-entrypoint.s..." About a minute Up About a minute 0.0.0.0:5050->5050/tcp, :::5050->5050/tcp, 5080/tcp cygnus
dc8af8657e11   mysql:5.7      "docker-entrypoint.s..." About a minute Up About a minute 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp mysql
c084997b04026   grafana/grafana "/run.sh"              About a minute Up About a minute 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp grafana
36aa0a52c075   mongo:4.4      "docker-entrypoint.s..." About a minute Up About a minute 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp mongo
fiware@VMFIWARE:/opt/fiware-docker$
```

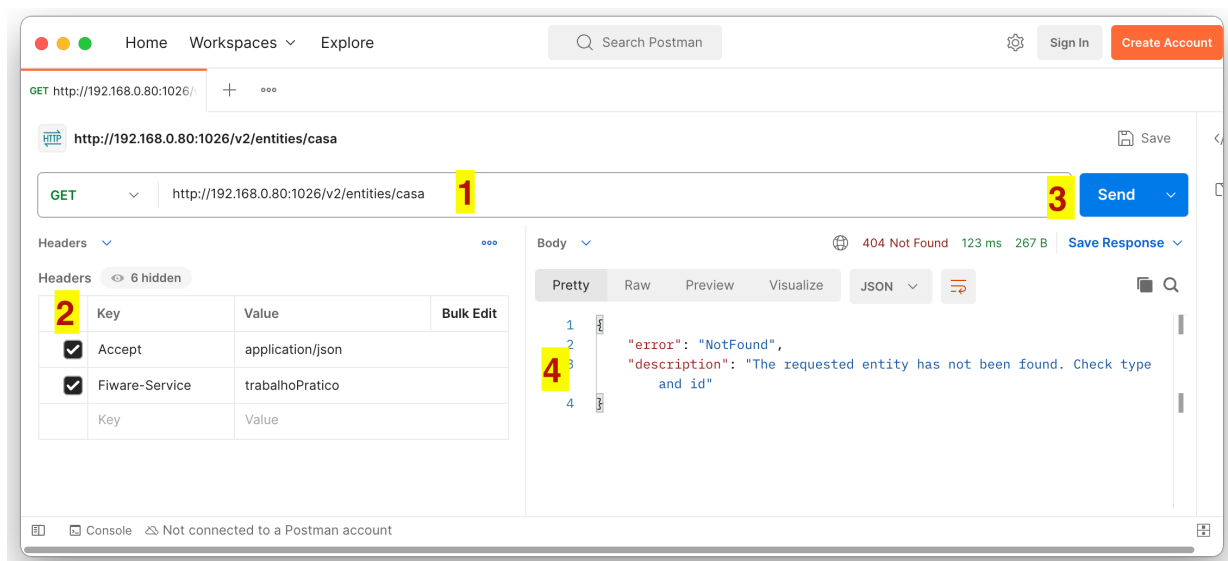
Interagindo com Fiware

Durante o TP, todas as informações recuperadas e solicitações feitas ao Fiware modules serão realizadas usando Postman;

Para tal abra o Postman e teste o módulo Orion:

Teste do módulo Orion usando o Postman

Vamos testar uma consulta de informações sobre uma entidade específica no serviço trabalhoPratico, siga os seguintes passos no Postman:



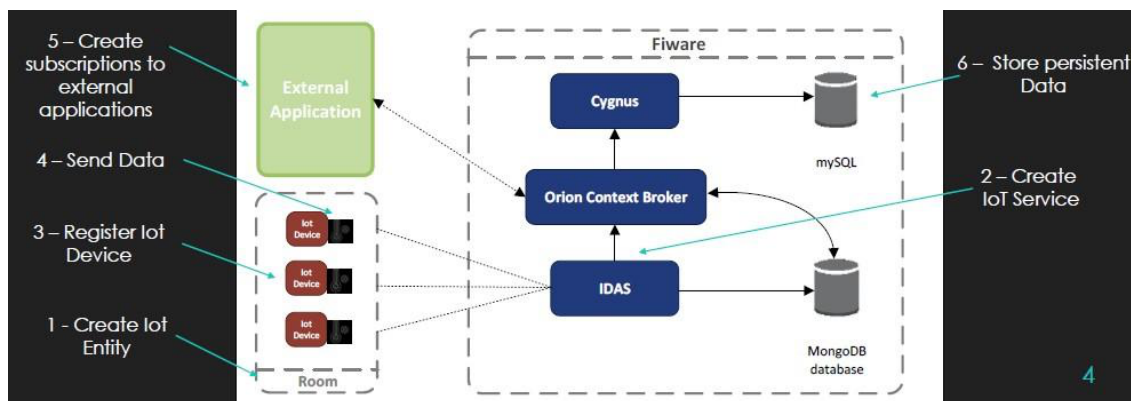
1. Selecione o método GET e insira o endereço **`http://<IP>:1026/v2/entities/casa`**
Obs: se estás a usar Docker na tua máquina o IP é localhost.
se estás a usar Virtual Box ou VM Ware insira o IP da máquina virtual;
2. Insira os cabeçalhos Accept e Fiware-Service conforme a image;
3. Execute a Requisição;
4. Observe a resposta. Como ainda não temos entidades criadas vamos receber o erro NotFound.

Agora que temos o Orion a funcionar vamos criar os nossos objectos.

Sendo que o objetivo deste TP é fornecer as operações básicas necessárias para interagir com a plataforma Fiware, usando um cenário IoT simple, iremos:

1. Criar entidades IoT usando o Orion Context Broker;
2. Criar e registrar dispositivos IoT no Idas;
3. Enviar dados de sensores usando o UltraLightIotAgent;
4. Criar assinaturas para aplicações externas;
5. Armazenar dados persistentes em MySQL usando o Cygnus;

Cenário IoT



© D. Raposo, 2017

Criar entidades IoT

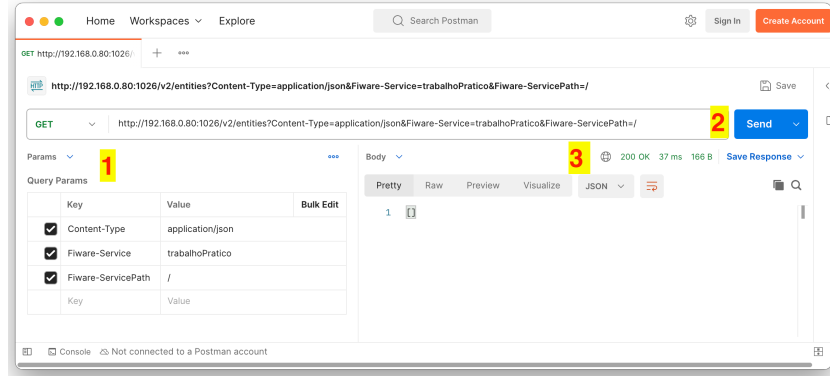
No FIWARE, a informação de contexto é representada por valores atribuídos a atributos que caracterizam as entidades na aplicação IoT. No nosso cenário de IoT, iremos representar a entidade **Sala (Room)** e alguns atributos a ela relacionados, como **temperatura** e **humidade**.

Para criar essa entidade, precisamos interagir com o **gestor de contexto do FIWARE**, chamado **Orion**, utilizando a porta **TCP 1026**.

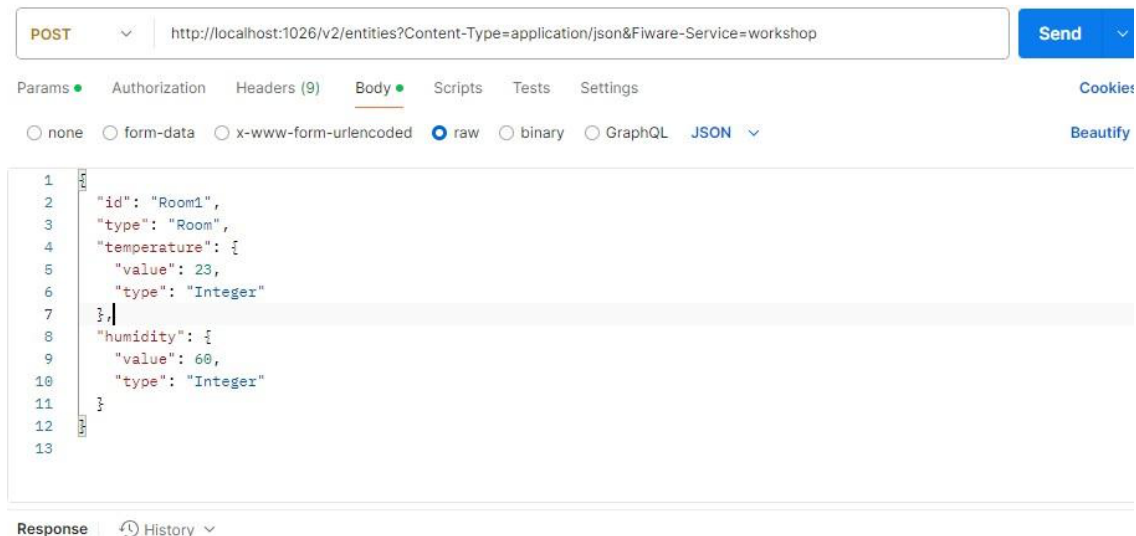
Usando postman criaremos a entidade **Room1** no **FiwareService** chamado **"trabalhoPratico"**. A partir deste momento, todas as requisições deverão referir este serviço. Inicie uma nova requisição, selecione o método GET, coloque na URL `http://<Seu IP>:1026/v2/entities`, e de seguida:

1. Os Query Params devem estar conforme a imagem;
2. Clica em send;
3. Receberás resposta 200 OK, indicando que o servisse foi criado com sucesso, conforme a imagem que se segue:

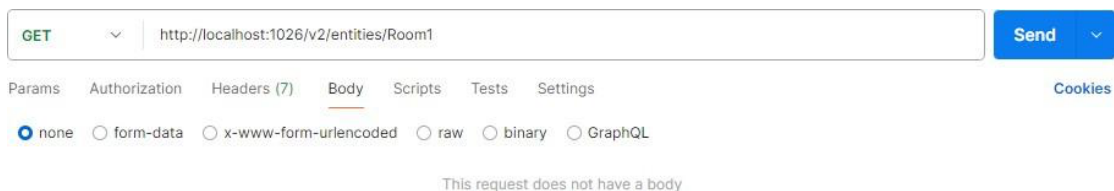
Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). + [infos](#)



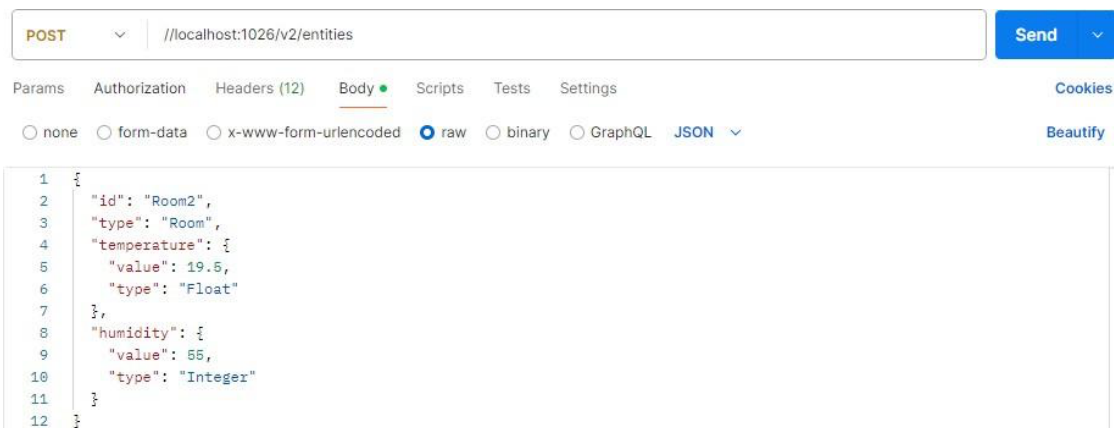
Agora vamos criar a Room no Service, preenchendo a request no postman conforme a imagem que segue e pressionar em send para fazer a request:



Para verificar se a entidade **Room1** foi criada no **FiwareService** chamado **"trabalhoPratico"** podemos fazer uma requisição **GET** específica:



Repita os mesmos passos para criar a entidade Room2 no Postman com os dados



Podemos obter a lista todas as entidades disponíveis no FiwareService workshop usando o Postman a requisição GET na url

Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). [+ infos](#)
http://<SeuIP>:1026/v2/entities :

GET http://localhost:1026/v2/entities/ Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Atualizar o atributo temperature da entidade Room2 no Postman

Para atualizar o valor do atributo **temperature** da entidade **Room2** no Postman, selecione o método **PUT**, insira a URL <http://localhost:1026/v2/entities/Room2/attrs/temperature/value>, depois vá até à aba **Headers** e adicione três cabeçalhos: o primeiro com a chave Fiware-Service e valor trabalhoPrático, o segundo com a chave Fiware-ServicePath e valor /, e o terceiro com a chave Content-Type e valor text/plain. Em seguida, vá até à aba **Body**, selecione a opção **raw**, escolha o formato **Text** (em vez de JSON), e insira apenas o valor que deseja definir, por exemplo 28.5. Por fim, clique em **Send** para enviar a requisição. Se a operação for bem-sucedida, o Postman retornará um status 204 No Content, indicando que o atributo foi atualizado com sucesso.

PUT http://192.168.0.80:1026/v2/entities/Room2/attrs/temperature/value Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> Fiware-Service	trabalhoPratico	
<input checked="" type="checkbox"/> Fiware-ServicePath	/	
<input checked="" type="checkbox"/> Content-Type	text/plain	
Key	Value	

Console Not connected to a Postman account

PUT http://localhost:1026/v2/entities/Room2/attrs/temperature/value Send

Params Authorization Headers (12) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL Text

1 28.5

Eliminar a entidade Room2

DELETE http://localhost:1026/v2/entities/Room2 Send

Params Authorization Headers (10) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

1

Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). [+ infos](#)
Criar serviço IoT

Depois de criarmos a nossa entidade no Orion, estamos prontos para o segundo passo. Nos passos 2 e 3 utilizaremos o módulo **IDAS**, que será responsável por gerir todos os dispositivos IoT, tecnologias e protocolos, e comunicar as atualizações das entidades ao Orion.

Neste caso, usaremos o **agente IDAS** para o protocolo **UltraLight 2.0**. O agente IoT funciona nas portas **4041** e **7896**.

Obter o IP do container Docker do Orion:

`docker inspect orion | grep "IPAddress"`

Registrar o Orion no IoT Agent

- **Método HTTP:** POST
- **URL:** <http://localhost:4041/iot/services>
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/

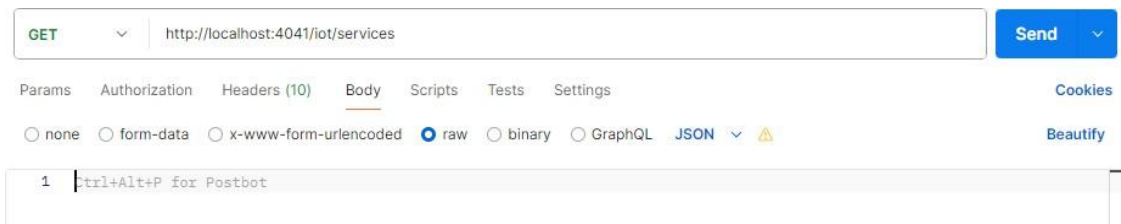
- **Body (Corpo):**
 - Selecione **raw**
 - Formato: **JSON**
 - Insira o seguinte conteúdo, substituindo `your_orion_ip_address` pelo IP real do container Orion (que você obteve com `docker inspect`):

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:4041/iot/services
- Body Type:** raw
- Format:** JSON
- Body Content:**

```
1 {
2   "services": [
3     {
4       "apikey": "workshop-devices",
5       "cbroker": "http://your_orion_ip_address:1026",
6       "entity_type": "thing",
7       "resource": "/iot/workshop-rooms"
8     }
9   ]
10 }
11 }
```

Para verificar o serviço utilize:

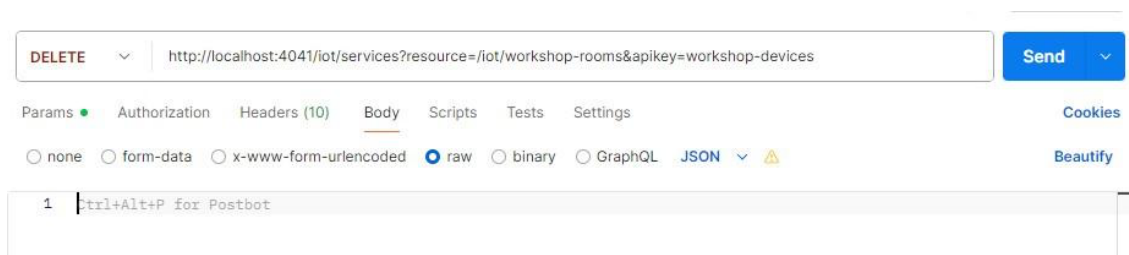


Nota: Se por algum motivo precisar eliminar o serviço, use:

Método: DELETE

- **URL:** <http://localhost:4041/iot/services?resource=/iot/workshop-rooms&apikey=workshop-devices>
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Accept	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/



Registre o dispositivo IoT

Após criar o serviço IoT, estamos prontos para o próximo passo: o registo do dispositivo. O registo do dispositivo pode ser feito pelo próprio dispositivo ou por qualquer outra aplicação externa.

- **Método HTTP:** POST
- **URL:** <http://localhost:4041/iot/devices>
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Accept	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/

POST ▼

Send ▼

Params Authorization Headers (12) **Body** ● Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼ ⚠ [Beautify](#)

```
1 {
2   "devices": [
3     {
4       "device_id": "sensor-a87020747f",
5       "protocol": "UL20",
6       "entity_name": "Room1",
7       "entity_type": "Room",
8       "timezone": "Angola/Benguela",
9       "attributes": [
10        {
11          "object_id": "t",
12          "name": "temperature",
13          "type": "Float"
14        }
15      ]
16    }
17  ]
18 }
19
```

Após clicar em **Send**, o IoT Agent registrará o dispositivo com ID sensor-a87020747f, associado à entidade Room1. A partir daí, será possível simular medições usando o protocolo **UltraLight 2.0** com esse device_id.

Verificar o registo do dispositivo.

- **Método:** GET
- **URL:** http://localhost:4041/iot/devices
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Accept	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/

GET ▼

Send ▼

Params Authorization Headers (12) **Body** ● Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼ ⚠ [Beautify](#)

```
1
2
```

Nota: Se por algum motivo precisar eliminar o dispositivo, use:

DELETE ▼

Send ▼

Params Authorization Headers (12) **Body** ● Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼ ⚠ [Beautify](#)

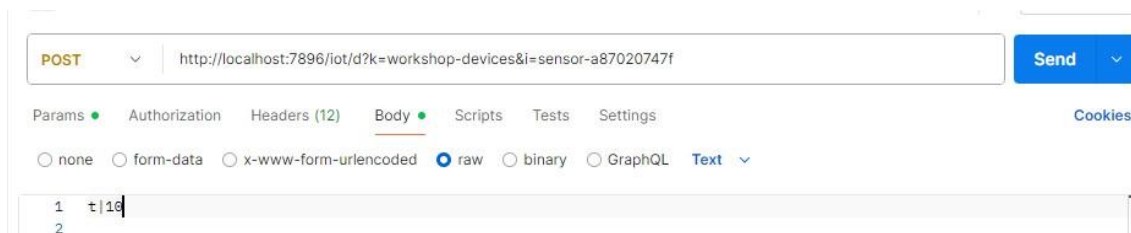
```
1
2
```

Enviar dados

Neste cenário, vamos emular um sensor enviando dados para o IoT Agent UL2.0. Note que o envio dos dados será feito utilizando o protocolo UL2.0.

- **Método HTTP:** POST
- **URL:**
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	text/plain
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/



POST ▼ http://localhost:7896/iot/d?k=workshop-devices&i=sensor-a87020747f Send ▼

Params ● Authorization Headers (12) **Body** ● Scripts Tests Settings Cookies

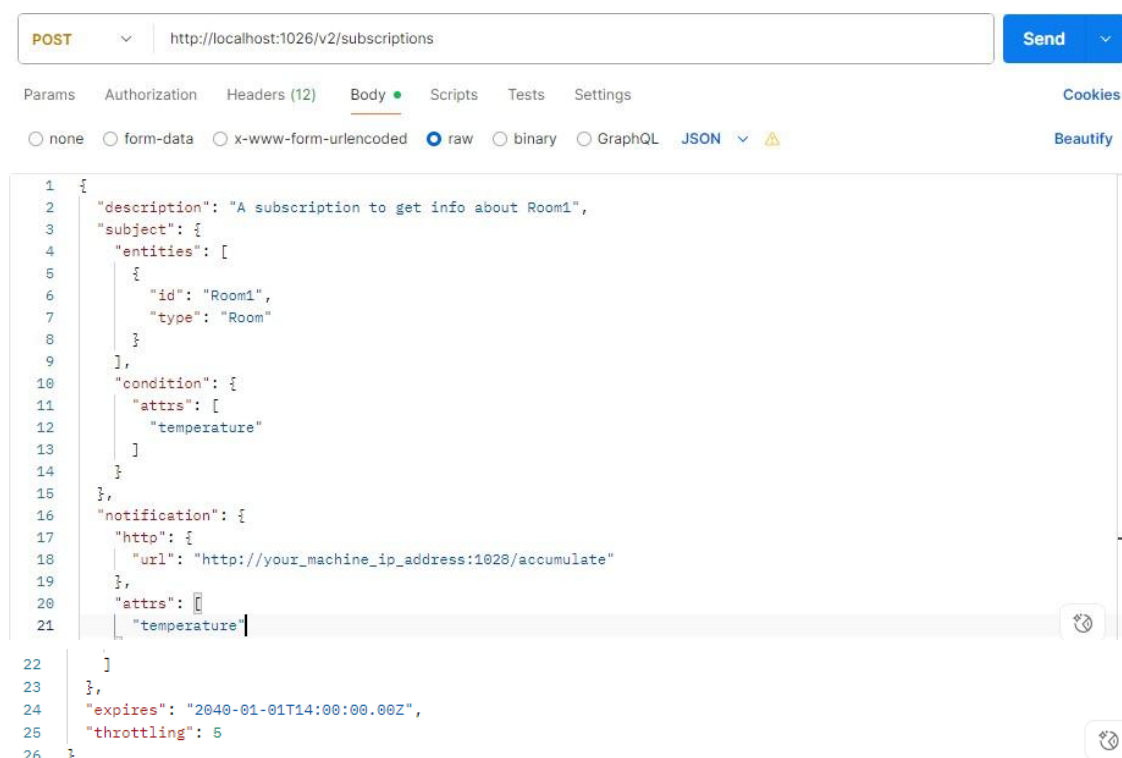
☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL Text ▼

```
1 t|10
2
```

Criar subscrições para aplicações externas:

- **Método:** POST
- **URL:** <http://localhost:1026/v2/subscriptions>
- **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/



POST ▼ http://localhost:1026/v2/subscriptions Send ▼

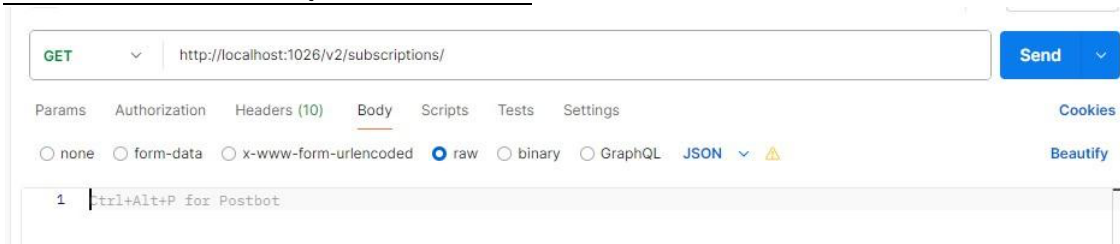
Params Authorization Headers (12) **Body** ● Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ ⚠ Beautify

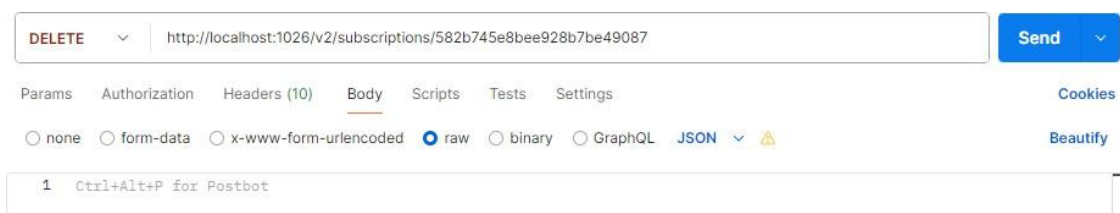
```
1 {
2   "description": "A subscription to get info about Room1",
3   "subject": {
4     "entities": [
5       {
6         "id": "Room1",
7         "type": "Room"
8       }
9     ],
10    "condition": {
11      "attrs": [
12        "temperature"
13      ]
14    }
15  },
16  "notification": {
17    "http": {
18      "url": "http://your_machine_ip_address:1028/accumulate"
19    },
20    "attrs": [
21      "temperature"
22    ]
23  },
24  "expires": "2040-01-01T14:00:00.00Z",
25  "throttling": 5
26 }
```

Nota: No espaço "**Your virtual machine IP**" , substitua pelo ip da máquina virtual.

Verificar Subscrições em Orion



Nota: Se por algum motivo precisar eliminar a subscrição, use:



Armazenar Dados Persistentes

Por fim, como provavelmente já notou, o Orion armazena apenas o último estado da entidade, e no contexto do IoT também é necessário ter acesso ao histórico dos dados. Neste tópico, iremos utilizar o Cygnus para subscrever a nossa entidade e guardar todas as alterações numa base de dados MySQL.

O Cygnus utiliza a tecnologia Apache Flume e pode ser usado em outros cenários e casos de uso.

Obtenha o endereço IP do container Cygnus usando o comando ***docker inspect***.

Registrar o Cygnus no Orion e enviar novamente mais dados dos sensores.

- o **Método:** POST
- o **URL:** <http://localhost:8081/v1/subscriptions?ngsi version=1>
- o **Headers (Cabeçalhos):**

Chave	Valor
Content-Type	application/json
Fiware-Service	trabalhoPratico
Fiware-ServicePath	/

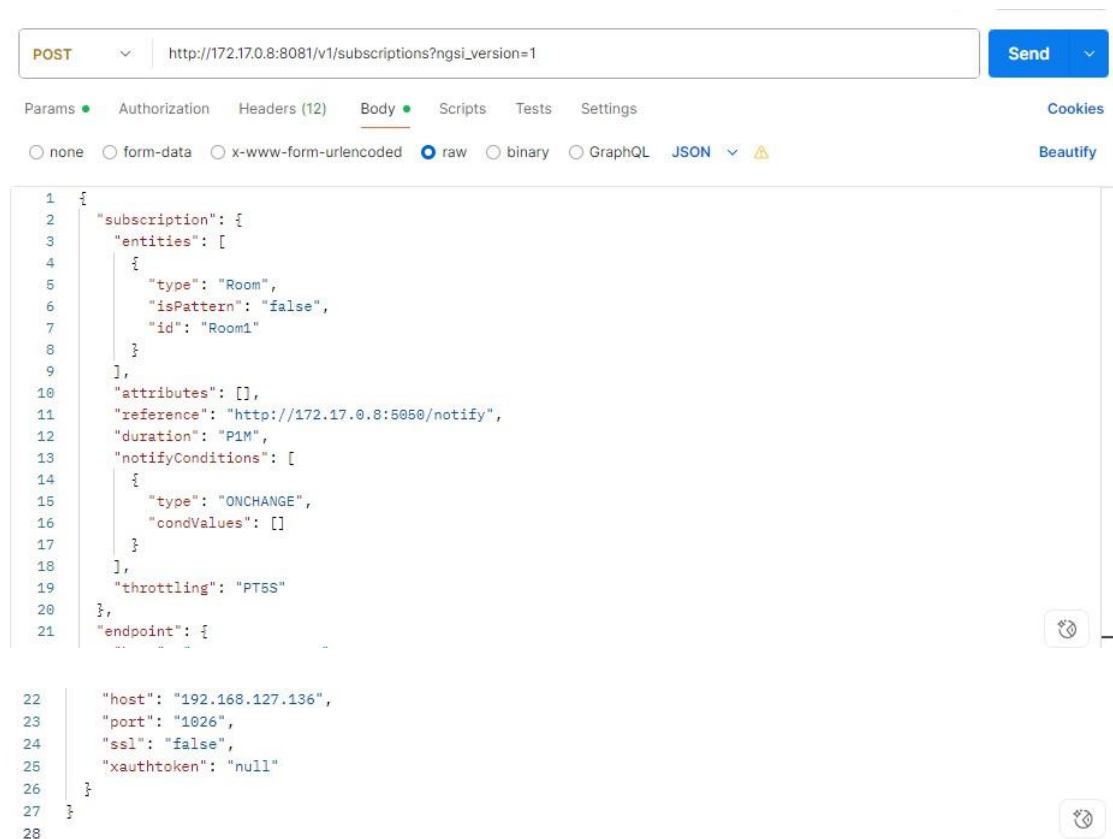
- o *Reference:* IP e porta onde o Cygnus está a escutar notificações

Versão inicial: [C.kiluando](#), Revisão e formatação: [N. Armando](#). [+ infos](#) (5050 por padrão).

- o `Endpoint.host`: IP do Orion acessível a partir do Cygnus. Ajusta se necessário.

- o `Ngisi_version=1`: especifica que a subscrição usa NGSI v1 (Cygnus requer esta versão, exceto se configurado para NGSIv2).

Nota: colocar o IP do seu container Cygnus.



Aceda ao terminal (bash) do container MySQL e digite o comando: **docker exec**

-it mysql bash

Inicie a sessão na ferramenta de linha de comandos do MySQL, introduza a palavra-passe "**mysql**" e execute os seguintes comandos:

- o `mysql-u root -p`
- o `show databases;`
- o `use workshop;`
- o `show tables;`
- o `Select * from Room1_Room;`

recvTimeTs	recvTime	fiwareServicePath	entityId	entityType	attrName	attrType	attrValue	attrMd
1479256444549	2016-11-16T00:34:04.549	/	Room1	Room	humidity	Integer	711	{}
1479256444549	2016-11-16T00:34:04.549	/	Room1	Room	temperature	Float	22	{}

2 rows in set (0.00 sec)

Lista de comandos do Docker

Executar o Docker Compose em segundo plano: **docker-compose up -d**

Parar o Docker Compose: **docker-compose stop**

Aceder à linha de comandos de um container Docker: **docker exec -it**

<containerIdOrName> bash

Ligar-se (anexar-se) a um container Docker: **docker attach**

<containerIdOrName>

Obter o IP do container Docker: **docker inspect <containerIdOrName> |
grep "IPAddress"**

Desafio (Trabalho autónomo)

- o Visualizar os dados na interface Grafana.

VII). Bibliografia

- o Cirillo, F., Solmaz, G., Berz, E. L., Bauer, M., Cheng, B., & Kovacs, E. (2020). A Standard-Based Open Source IoT Platform: FIWARE. *IEEE Internet of Things Magazine*, 2(3), 12-18. <https://doi.org/10.1109/iotm.0001.1800022>
- o Fernandes J.M., Raposo D., Armando N., Sinche S., Sá Silva J., Rodrigues, A., Pereira V., Gonçalo Oliveira H., Macedo L., Boavida F., "ISABELA A Socially-Aware Human-in-the-Loop Advisor System", *Online Social Networks and Media*, vol. 16, pp. 100060-100060, 2020. <https://doi.org/10.1016/j.osnem.2020.100060>
- o Armando N.; Kiluando, P. (2024). Cutting-edge Approaches to Mitigate the National Agro-Digital Deficit Towards Angola 2050. Technical Papers Presentation, 10th UNESCO Africa Engineering Week and 8th Africa Engineering Conference. Luanda-Angola, September 9th-13th, 2024. <https://asric.africa/engineering-sciences/asric-journal-engineering-sciences-2024-v4-i2/cutting-edge-approaches-mitigate>
- o <https://github.com/Fiware/tutorials.TourGuide-App>
- o <https://www.fiware.org/devguides/fiware-tour-guide-application-a-tutorial-on-how-to-integrate-the-main-fiware-ges/>
- o <http://fiwaretourguide.readthedocs.io/en/latest/development-context-aware-applications/deep-dive/>
- o <https://github.com/telefonicaid/iotagent-ul>
- o <http://fiwaretourguide.readthedocs.io/en/latest/>
- o <https://www.youtube.com/playlist?list=PLR9elAI9JscSOuSnwIkGzSVWlQKgfDk6d>

Fim do documento