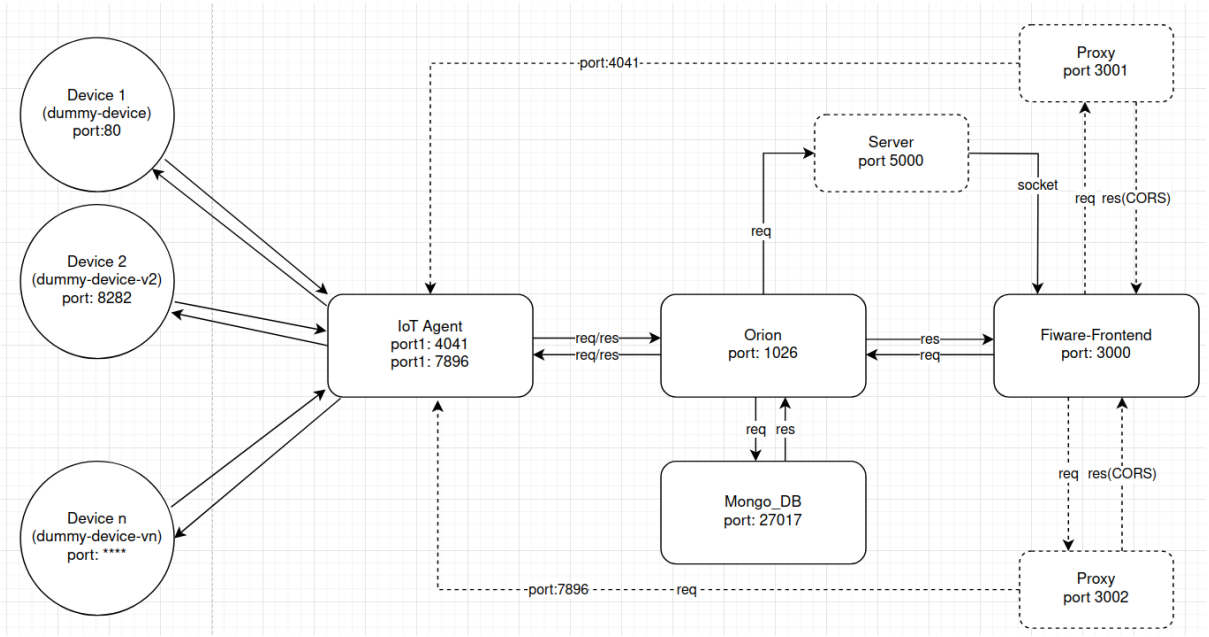
# Smart Classroom




Para baixar o projeto, você precisará clonar o repositório criado no github usando a seguinte linha de comando.

- git clone https://github.com/fventuraq/Smart-Classroom-IoT.git

Para criar as imagens dos containers, você deve entrar na pasta Smart-Classroom-IoT e executar a seguinte linha de comando.

- linux-ubuntu: sudo docker-compose build
- windows: docker-compose build

Para inicializar os containers você deve executar a seguinte linha de comando:

- linux-ubuntu: sudo docker-compose up
- windows: docker-compose up -d

# Fiware Frontend

**Fiware Frontend** é uma aplicação web criada com React, esta aplicação permite a criação das diferentes entidades utilizadas no nosso projeto, por sua vez é implementada uma lógica para que as entidades sejam criadas seguindo a seguinte hierarquia:

- Universidade
    - Campus
        - Andares
            - Salas
                - Devices

Para criar cada uma das entidades é utilizada a API **Orion**, que irá cadastrar cada entidade no banco de dados não relacionado ao **Mongodb**.

Os dados armazenados no **Mongodb** podem ser chamados usando a API Orion, chamando especificamente um elemento ou chamando um grupo de elementos que têm algo em comum.

Para inicializar a aplicação web, você deve ir até a pasta /fiware-frontend e executar o seguinte comando.

***Primeira execução.***

1. npm install --force
2. npm start

***Se você já instalou as dependências:***

1. npm start

# IoT Agent

No caso dos **serviços**, estes são criados no **IoT Agent**. Mas para isso utilizaremos também a **aplicação web**, que permitirá cadastrar cada serviço através de um formulário, que deverá então ser enviado ao Agente IoT.

A comunicação entre a **aplicação web** e o **Agente IoT** é realizada através de dois servidores proxy, para cada uma de suas portas de **IoT Agent** (4041, 7896).

Os **servidores proxy** foram criados, pois devido às políticas CORS (Cross-Origin Resource Sharing) em **aplicações web**, os serviços externos neste caso IoT Agent devem incluir cabeçalhos CORS em suas respostas. E como o IoT Agent não inclui esses cabeçalhos, por meio de servidores proxy, adicionamos cabeçalhos CORS nas respostas do IoT Agent.

A aplicação web também contém uma interface onde são exibidos os diferentes serviços criados no IoT Agent.

A criação de ‘Devices’ também é feita através de um formulário na aplicação web. Os dispositivos são registrados usando a API IoT Agent. Este por sua vez enviará uma solicitação ao Orion, registrando uma entidade do tipo Device.

Para inicializar os servidores proxy você deve ir até a pasta fiware-frontend/src e executar os seguintes comandos.

1. node proxy.js
2. node proxy2.js

Quando os 'Devices' são iniciados, eles enviam dados periodicamente ao IoT Agent, que é responsável por validar se o 'Device' está autorizado para enviar dados. Após a autorização do Device pelo IoT Agent, seus dados são enviados para o Orion, que se encarrega de registrar esses dados no MongoDB.

Uma das funcionalidades oferecidas pelo Orion é o processo de assinatura, que consiste em monitorar um tipo específico de dados de um 'Device' e, caso haja alguma alteração, comunicar a um servidor externo sobre essa mudança por meio de uma solicitação POST.

Na nossa proposta, o Orion não pode estabelecer uma comunicação direta com a nossa aplicação web, pois esta última não permite a criação de rotas de serviço nem o tratamento de solicitações POST. Visto que a nossa aplicação web opera no lado do cliente por meio de uma interface e não foi projetada para lidar com lógica de servidor ou receber solicitações diretamente, implementamos um microservidor (server.js). Neste microservidor, criamos "rotas" específicas para permitir que o Orion comunique as alterações nas variáveis do 'Device'. Além disso, como os dados provenientes do Orion são periódicos (a cada 5 segundos), incorporamos um socket que facilita uma comunicação constante entre server.js e a aplicação web.

Para inicializar o microservidor você deve ir até a pasta fiware-frontend/src e executar os seguintes comandos.

3. node server.js

Ao criar a assinatura, lembre-se que o endereço onde as notificações chegaram é no seu IP na porta 5000/notify.

Exemplo: "http://192.168.0.106:5000/notify"

# Device 1 AND 2

'Device' 1 e 2 são aplicações que simulam o funcionamento de um dispositivo que contém um conjunto de sensores e atuadores.

A funcionalidade atual destes permite receber solicitações, o que ativará funcionalidades como o envio de dados como se fosse um sensor ou uma solicitação para desativar ou desligar os sensores, o que na realidade seria parar de enviar dados.

Para testes parciais foi criada uma interface onde são acessados os dados de um dispositivo e a partir desta interface é enviada uma solicitação para ativar um dos dispositivos.

O processo de comunicação ocorre da seguinte:

- A interface web faz uma solicitação ao Orion, esta solicitação contém o ID do dispositivo e os comandos a serem executados.
- Orion recebe a solicitação da aplicação web, valida a solicitação e envia uma solicitação ao IoT Agent e, tendo comunicação direta com os dispositivos ‘Device’,

faz uma solicitação enviando um JSON que contém o comando a ser executado e as variáveis de configuração.

- O ‘Device’ analisa o comando a ser executado, processa as informações e retorna a resposta ao IoT Agent, mas neste caso as respostas enviadas do ‘Device’ são enviadas para a porta 7826 do IoT Agent.
- O IoT Agent processará os dados e enviará atualizações para o Orion, e o Orion registrará os dados no mongodb.
- A aplicação web pode consumir os dados de atualização usando a API Orion.

O objetivo final é criar uma funcionalidade na aplicações web, para que os dados sejam atualizados periodicamente caso o Dispositivo esteja em estado ativo.