Universidade Estácio de Sá

- DESENVOLVIMENTO FULL STACK-
- Disciplina: RPG0024 - Vamos Interligar as Coisas Com a Nuvem!

- Júlio Cesar Santos Ramos - MATRICULA: 202302798721

Missão Prática | Nível 5 | Mundo 4

Azure IoT Hub Integration with Raspberry Pi Simulator

Objetivos

- Criar um Hub IoT no Azure.
- Registrar um novo dispositivo no Hub IoT.
- Adicionar a extensão Hub IoT do Azure para Visual Studio Code.
- Gerenciar e interagir com o Hub IoT.
- Conectar o simulador online Raspberry Pi ao Hub IoT do Azure.

Procedimentos

```
1. Criar um Hub IoT no Azure

1. Acesse o portal do Azure através do seu navegador web.
2. No painel de controle do Azure, clique em **"Criar um recurso"**.
3. No menu Categorias, escolha **Internet das Coisas** e, em
seguida, selecione **Hub IoT**.
4. Selecione **Avançar: Rede** para continuar criando o hub.
5. Na guia Rede, preencha os campos conforme necessário e selecione
**Avançar: Gerenciamento**.
6. Na guia Gerenciamento, aceite as configurações padrão ou
modifique conforme desejado.
7. Selecione **Avançar: Complementos** e aceite as configurações
padrão.
8. Selecione **Avançar: Marcas** e ignore a adição de marcas para
esta prática.
9. Selecione Avançar: Analisar + criar** para revisar suas escolhas.
10. Selecione Criar para iniciar a implantação do novo hub.
```

11. Após a implantação ser concluída, selecione "Ir para o recurso" para abrir o novo hub.

2. Registrar um novo dispositivo no Hub IoT

1. Acesse o portal do Azure através do seu navegador web.
2. No painel de controle do Azure, selecione o Hub IoT criado, por exemplo, chamado `Fullstack`.
3. No menu de navegação do Hub IoT, selecione “Dispositivos”.
4. Na área de Dispositivos, selecione **Adicionar Dispositivo.
5. Em Criar um dispositivo, forneça um nome para o novo dispositivo e selecione Salvar.
6. Após a criação do dispositivo, localize-o na lista do painel de Dispositivos e copie a cadeia de conexão primária.

Observação importante: O Registro de Identidade no Hub IoT do Azure é responsável por armazenar exclusivamente as identidades dos dispositivos, garantindo acesso seguro ao Hub IoT. Este registro mantém informações cruciais, incluindo IDs e chaves do dispositivo, utilizadas como credenciais de segurança.

3. Adicionar a extensão Hub IoT do Azure para Visual Studio Code

Procedimentos

1. Abra o Visual Studio Code.
2. Vá até a barra lateral e clique no ícone de exibição Extensões na barra de exibição ou selecione `Ctrl+Shift+X` para acessar o Marketplace.
3. Na barra de pesquisa, digite "Azure IoT Hub" e pressione Enter.
4. Localize a extensão Azure IoT Hub nos resultados e confirme se o identificador exclusivo da extensão selecionada, exibido na seção Mais Informações da página de detalhes da extensão, está definido como `vsciot-vscode.azure-iot-toolkit`.
5. Após a instalação, reinicie o VS Code, se necessário. Ao final, a página do Azure IoT Hub Extension deve ser exibida.

> Observações importantes:
> - Ao instalar a extensão Azure IoT Hub para o Visual Studio Code, você também incorpora a extensão Azure Account. A extensão Azure Account oferece autenticação única no Azure, facilitando a filtragem de assinaturas para outras extensões Azure. Além disso, ela disponibiliza o serviço Azure Cloud Shell no terminal integrado do VS Code.
> - A extensão Azure IoT Hub depende da Azure Account para conectividade. A desinstalação da Azure Account implicará na remoção da extensão Azure IoT Hub.

Conectar ao Hub IoT

1. No modo Explorer do VS Code, expanda a seção "Hub IoT do Azure" na barra lateral.
2. Selecione as reticências (...) na seção "Hub IoT do Azure" para abrir o menu de ações. Escolha "Selecionar Hub IoT" no menu.
3. Se não estiver conectado ao Azure, uma notificação pop-up aparecerá no canto inferior direito. Selecione "Entrar" e siga as

instruções para entrar no Azure.
4. Escolha sua assinatura do Azure na lista suspensa **"Selecionar Assinatura"**.
5. Selecione seu Hub IoT na lista suspensa **"Selecionar Hub IoT"**.
6. Os dispositivos do seu Hub IoT serão recuperados e exibidos na seção "Dispositivos" na barra lateral.

> Observação: Você também pode usar uma cadeia de conexão para acessar seu Hub IoT. Selecione "Definir Cadeia de Conexão do Hub IoT" no menu de ações e insira a cadeia de conexão de política "iothubowner" para o seu Hub IoT.

4. Conectar o simulador online Raspberry Pi ao Hub IoT do Azure

Procedimentos

1. Acesse o [Raspberry Pi Azure IoT Online Simulator](https://azure-samples.github.io/raspberry-pi-web-simulator/#GetStarted).

2. Substitua a cadeia de conexão no código de exemplo pela cadeia de conexão do dispositivo do Hub IoT que você copiou anteriormente.

3. Utilize o editor de código online para interagir com o Raspberry Pi.

Código de exemplo

```
javascript
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const BME280 = require('bme280-sensor');

const BME280_OPTION = {
  i2cBusNo: 1,
  i2cAddress: BME280.BME280_DEFAULT_I2C_ADDRESS()
};

const connectionString = 'HostName=nome-do-seu-hub.azure-
devices.net;DeviceId=dispositivo-
001;SharedAccessKey=xxxxxxxxxxxxxx';

const LEDPin = 4;

var sendingMessage = false;
var messageId = 0;
var client, sensor;
var blinkLEDTimeout = null;

function getMessage(cb) {
  messageId++;
  sensor.readSensorData()
    .then(function (data) {
      cb(JSON.stringify({
        messageId: messageId,
        deviceId: 'Raspberry Pi Web Client',
```

```
        temperature: data.temperature_C,
        humidity: data.humidity
      }), data.temperature_C > 30);
    })
    .catch(function (err) {
      console.error('Failed to read out sensor data: ' + err);
    });
}

function sendMessage() {
  if (!sendingMessage) { return; }

  getMessage(function (content, temperatureAlert) {
    var message = new Message(content);
    message.properties.add('temperatureAlert',
temperatureAlert.toString());
    console.log('Sending message: ' + content);
    client.sendEvent(message, function (err) {
      if (err) {
        console.error('Failed to send message to Azure IoT Hub');
      } else {
        blinkLED();
        console.log('Message sent to Azure IoT Hub');
      }
    });
  });
}

function onStart(request, response) {
  console.log('Try to invoke method start(' + request.payload +
')');
  sendingMessage = true;

  response.send(200, 'Successfully start sending message to cloud',
function (err) {
    if (err) {
      console.error('[IoT hub Client] Failed sending a method
response:\n' + err.message);
    }
  });
}

function onStop(request, response) {
  console.log('Try to invoke method stop(' + request.payload + ')');
  sendingMessage = false;

  response.send(200, 'Successfully stop sending message to cloud',
function (err) {
    if (err) {
      console.error('[IoT hub Client] Failed sending a method
response:\n' + err.message);
    }
  });
}

function receiveMessageCallback(msg) {
  blinkLED();
  var message = msg.getData().toString('utf-8');
```

```
    client.complete(msg, function () {
      console.log('Receive message: ' + message);
    });
  }

  function blinkLED() {
    if (blinkLEDTimeout) {
      clearTimeout(blinkLEDTimeout);
    }
    wpi.digitalWrite(LEDPin, 1);
    blinkLEDTimeout = setTimeout(function () {
      wpi.digitalWrite(LEDPin, 0);
    }, 500);
  }

  wpi.setup('wpi');
  wpi.pinMode(LEDPin, wpi.OUTPUT);
  sensor = new BME280(BME280_OPTION);
  sensor.init()
    .then(function () {
      sendingMessage = true;
    })
    .catch(function (err) {
      console.error(err.message || err);
    });

  client = Client.fromConnectionString(connectionString, Protocol);

  client.open(function (err) {
    if (err) {
      console.error('[IoT hub Client] Connect error: ' + err.message);
      return;
    }

    client.onDeviceMethod('start', onStart);
    client.onDeviceMethod('stop', onStop);
    client.on('message', receiveMessageCallback);
    setInterval(sendMessage, 2000);
  });
```

Visualização em Tempo Real de Dados de Temperatura e Umidade

Este projeto consiste em um aplicativo web que permite visualizar em tempo real os dados de temperatura e umidade enviados por dispositivos IoT conectados a um Azure IoT Hub.

Configuração do Ambiente Local

Pré-requisitos

- Node.js (versão 16 LTS ou superior)
- npm (Node Package Manager)
- CLI do Azure configurada e conectada à sua conta Azure

Instalação de Dependências

Para instalar as dependências necessárias, execute o seguinte comando no diretório raiz do projeto:

bash
npm install

set IotHubConnectionString=YOUR_IOT_HUB_CONNECTION_STRING
set EventHubConsumerGroup=YOUR_CONSUMER_GROUP_NAME

npm start

az iot hub consumer-group create --hub-name YOUR_IOT_HUB_NAME --name YOUR_CONSUMER_GROUP_NAME

az iot hub show-connection-string --hub-name YOUR_IOT_HUB_NAME --policy-name service

az webapp create -n YOUR_WEB_APP_NAME -g YOUR_RESOURCE_GROUP_NAME -p YOUR_APP_SERVICE_PLAN_NAME --runtime "NODE:16LTS" --deployment-local-git

az webapp config appsettings set -n YOUR_WEB_APP_NAME -g YOUR_RESOURCE_GROUP_NAME --settings EventHubConsumerGroup=YOUR_CONSUMER_GROUP_NAME IotHubConnectionString="YOUR_IOT_HUB_CONNECTION_STRING"

az webapp show -n YOUR_WEB_APP_NAME -g YOUR_RESOURCE_GROUP_NAME --query state