Irriga GLOBAL | Field Innovation.

Teste técnico

Cargo desejado: Analista de Sistemas Back-end
Linguagem: Javascript (Node.js)
Cargo desejado: Desenvolvedor Back-End.
Prazo: 30 de Dezembro de 2019, 11:00 AM BRT.

Descrição

Como Analista de Sistemas Back-end, no Irriga Global, você vai desempenhar diversas atividades para integração de APIs, tanto de terceiros quando as nossas próprias integrações da plataforma com aplicações web e móveis. Para isso você irá desenvolver um micro serviço de estações meteorológicas e alimentá-lo com dados.

Critérios observados

+ Implementação correta dos requisitos;
+ Organização de código e utilização de padrões;
+ Domínio da linguagem;
+ Documentação;
+ Escolha de bibliotecas/frameworks;
+ Performance;
+ Reaproveitamento de código;
+ Implementação de testes;
+ Criatividade.

Requisitos

+ **Primeira etapa**: Você deverá criar uma API REST de estações meteorológicas. Esta API fornecerá dados tanto para aplicações móveis quanto para páginas web. Outros serviços podem utilizá-la como forma de interagir com este micro serviço.
+ **Segunda etapa**: Você deverá criar um script que ao ser executado, busca as informações em um sistema terceiro (https://darksky.net/) e utiliza a API criada anteriormente para inserir os dados.

Requisitos de ambiente:

Node.js (v12)
MySQL: (v5.7 ou v8)

+ Primeira etapa:

Criar uma API REST para um micro serviço de dados meteorológicos. Esta API deverá conter 5 *endpoints,* descritos abaixo e seguir a estrutura definida do banco de dados.

Banco de dados

O banco de dados deverá seguir a estrutura contida no arquivo **dbSchema-IRRIGA.sql**.
O nome do banco de dados é IRRIGA. A tabela que contém dados de estações é **weather_stations**.
Cada estação meteorológica tem sua própria tabela de dados, seguindo o padrão **weather_data_<id>,** em que <id> é o identificador da estação. Ex.: Uma nova estação é criada com identificador **5**. Uma tabela para ela deverá ser criada com o nome **weather_data_5**.

POST /weather-stations

Criação de uma nova estação meteorológica.

Você deve informar no corpo da requisição, um objeto JSON com os seguintes atributos:
id: um identificador inteiro positivo único (opcional). Se nenhum for informado, você deve gerar um. Ex.: 1
name: um nome, do tipo *string*, para identificação da estação. Ex.: Santa Maria
latitude: coordenada geográfica da estação meteorológica, formato decimal. Ex.: -29.000000
longitude: coordenada geográfica da estação meteorológica, formato decimal. Ex.: -53.000000
altitude: altitude em metros. Ex.: 100
timezone: fuso horário. Ex.: Europe/Budapest

Se criado com sucesso, a requisição deve retornar status 201. Se uma estação com o mesmo ID já existir, deve retornar erro 400.

GET /weather-stations

Lista todas as estações meteorológicas disponíveis.

O retorno deve conter um array JSON com as estações meteorológicas e status 200.

GET /weather-stations/:id

Lista as informações de uma estação meteorológica, através de seu identificador definido por :id.
Deve retornar um array JSON e status 200. Se a estação não for encontrada, deve retornar status 404.

POST /weather-data/:id

Insere dados meteorológicos para uma estação específica, definida em :id.

Você deve informar no corpo da requisição um array de objetos no formato JSON, onde cada objeto deve estar de acordo com a seguinte estrutura:
moment: momento que o dado foi medido. Ex.: “2019-12-23 10:00:00”.
air_temperature: temperatura do ar, em Celcius. Ex.: 25.
air_humidity: umidade relativa do ar, em %. Ex.: 90.
wind_speed: velocidade do vento, em m/s: Ex: 5.
rainfall: chuva, em mm. Ex: 20.

Obs.: o valor da data em **moment** deve ser salvo com fuso horário 0 (UTC).

Ao inserir os dados, deve retornar status 201. Não deve permitir inserir novamente um dado para a mesma estação e mesma data, retornando erro 409.

GET /weather-data/:id

Busca os dados meteorológicos de uma estação meteorológica, definida em :id.

Obs.: a data de cada dado **deve estar no fuso horário da estação (weather_stations.timezone)**.
Ex.: uma estação com fuso horário America/Sao_Paulo, sem horário de verão, deve apresentar seus dados com 3 horas a menos com relação ao GMT.

Deve ser retornado um array JSON, com status 200 se dados foram encontrados. Se a estação não existir, o status 404 deve ser retornado.

+ **Segunda etapa:**
  Criar um script que ao ser executado faça download de dados meteorológicos da plataforma Darksky e os insira por meio da API criada anteriormente.

Documentação da API: https://darksky.net/dev/docs
Endpoint desejado: dados de previsão horária, forecast (hourly).
Chave para requisições (secret key**)**: cc57eded744c264838f0f10fec22fca4

Para cada estação meteorológica cadastrada na API anteriormente criada (GET /weather-stations), você deve buscar dados de previsão meteorológica horária e inseri-los no micro serviço (POST /weather-data/:id). Cada estação contém uma latitude e longitude, que devem ser usadas na requisição.

Os dados meteorológicos que buscamos são:

+ Temperatura do ar (Celcius);
+ Umidade relativa do ar (%);
+ Velocidade do vento (m/s);
+ Volume de chuva (mm);
+ Data e hora da medição.

O script deve ser salvo com o nome **darksky.js** e será executado com o comando:
node darksky.js

Obs.: Este script não deve se comunicar com o banco de dados diretamente. Todas as interações devem ser feitas utilizando a API criada na primeira etapa.
O volume de chuva não é fornecido pela API diretamente, você deve ler a documentação do Darksky para entender como buscá-lo/calculá-lo.
Alguns dados podem necessitar conversão de escala.

O que você deve entregar:

Você deverá responder o email que foi enviado com esta definição do teste, informando o link do **repositório Github** contendo os códigos-fonte da aplicação. O script e a API devem estar no mesmo repositório.

Neste repositório deverá ter um documento **README.md**, descrevendo os passos de como executar o projeto.