Teste – DEV Pleno

PARTE 01:

a) Unicamente para receber os dados de medição eu criaria uma rota do tipo POST para receber no body o modelo da medição e em query receberia um Id que faria menção ao sensor do qual a medição pertence, essa rota em si faria o vinculo da medição à um sensor e armazenando-a em banco.

b) Eu criaria um modelo que representaria o dado do sensor recebendo as informações, porém no caso da criação de uma nova medição eu não faria o envio do ID para a rota, já que a API criada deveria ter total responsabilidade por fazer o controle da chave de seus próprios registros.

c) Eu usaria o MongoDB, já que estamos falando de vários sensores e de ainda mais equipamentos dentro de um mesmo local o processo deve ser performático, então usando um banco não relacional podemos fazer manipulações de informações no banco sem exigir muito da memória do servidor e assim poupando performance para a alta escala de dados.

PARTE 02:

a) Ao verificar as rotas que criei na API poderá ser notado que temos as controllers de equipamento, sensor e enfim medições. Assim podemos criar um equipamento e vincular o seu ID à um sensor, assim fazemos a amarração entre eles e ainda sim respeitando a premissão de "não relacional" do MongoDB, já em relação as medições, as mesmas estão sendo armazenadas majoritariamente dentro de seus devidos sensores, seguindo a premissa de que foi interpretado que não é lógico haver uma medição sem seu sensor de origem.

b) Levando em consideração que já fizemos a amarração entre as entidades usadas na aplicação, fiz uma rota do modelo GET que trará os sensores que carregam o vinculo com o equipamento enviado em sua query. Logo que verificado que o equipamento está armazenado e ele possui sensores, fiz uma listagem de suas medições e ordenados por data de criação e limitei aos 10 primeiros de cada sensor.

PARTE 03:

a) Levando em consideração a performance, fiz um JOB usando a técnologia Hangfire para que seja executado um serviço por trás das funções frente do projeto. Levando esse JOB em um nível mais bem explicado, ele faz a pesquisa de todos os sensores e seus ultimos 50 registros, assim consigo verificar as regras dos ultimos 5 e também a média de todos eles e se a mesma se encaixa na margem de erro citada na problemática, satisfazendo os dois cenários é feito o envio de e-mail informando o destinátario da situação.

c) Feito o teste para que seja testado todos os cenários citados acima, usando a técnologia xUnit, Mock e entre outras. Pode ser melhor análisando olhando para a camada de Presentation, dentro da pasta de numero 1.2 chamada de test.

Parte 04:

R. Nesse caso, acredito que caberia fazer a avaliação de uma mensageria para criação de medições, assim teríamos certeza de o que a API não está conseguindo receber ficará armazenado na fila para ser processado!
Duas técnologias que tive o prazer de trabalhar que funcionaram bem para casos de alta escalabilidade são o MassTransit para o controle da fila e o RabbitMQ para a aplicação do Publisher do lado da aplicação dos sensores e o Consumer para a API em questão