API - Register Point

Ibukun C. D. Adjitche

[1] Boa Vista – Roraima – Brazil

adjitchedidier1@gmail.com

Resumo. *Este documento é um relatório, contextualizando o problema referente à marcação de ponto e apresentando os estudos e as condições de como foi implementado o serviço backend da API Register Point que servirá para receber as informações de funcionário marcando seus pontos no caso de uma empresa. No primeiro instante será apresentado a natureza do problema, em seguida a arquitetura sugerida para resolução do problema e processo para chegar nesta solução; e por fim as conclusões finais.*

1. Introdução

O meio de controlar a jornada de trabalho dos funcionários no Brasil, se realiza via um sistema de marcação de ponto. Esse sistema de marcação é geralmente feita em um intervalo de 5 minutos. Todos os funcionários, a princípio, devem já ter finalizado a sua marcação.

Esse cenário é usado com milhares de colaboradores de centenas empresas pelo Brasil, ao mesmo tempo simultaneamente. Isto é um problema que realça o risco de sobrecarregar os servidores e a perda importante dos dados de marcação dos funcionários.

A solução pensada e desenvolvida para resolver esse problema funcional e não funcional, que as empresas podem ser sujeitas para registrar as marcações de ponto, é o uso de uma fila que armazenaria à primeira vista as requisições de marcação de ponto. Nas próximas linhas abordaremos o serviço de fila escolhido e o processo de implementação da solução resolvendo o problema.

2. Entendendo o Problema

Diversos profissionais de tecnologias apontam a implementação de um algoritmo de fila para resolver problemas de sobrecarga dos servidores ou a execução de várias Thread para cada requisição. O que acontece é que esse tipo de abordagem funciona dependendo da natureza do problema. Por exemplo, caso exista o acesso em uma área compartilhada em requisições executadas em paralelo, ocorre um risco de perda de informações ou a parada do servidor. Neste caso o uso de uma fila é necessário para guardar as informações e executá-las por suas vezes.

A natureza do problema de marcação de pontos, é que, os dados de marcação a serem recebidos pelo sistema são diferentes, independente e única em cada requisição. O que poderia nos levar a pensar no uso de uma programação multi-processo ou Thread para resolver o congestionamento, fazendo uso de todas as capacidades do servidor. Mas podemos nos deparar com perda de informações nesta avalancha de requisição ou caso de um erro acontecer na rede. Motivo pelo qual o uso de um algoritmo de fila encontra se importante na primeira vista antes de pensar e todas as outras formas de uma escalabilidade horizontal.

3. AWS-Simple Queue Service

O uso de algoritmos de fila necessita funções robustas assegurando a recuperação certa e intacta dos dados salvos nelas. Razão pelo qual foi implementado a api Register Point com os serviços da Fila da AWS, que de forma segura na nuvem, conserva os dados e tem toda uma política de resgate das informações quando a mensagem da fila está sendo requisitada por uma api e não está totalmente consumida. A menos que ela esteja apagada.

Assim, é criado uma conta na aws para o uso do serviço AWS-Simple Queue Service. É criado em seguinte uma fila de modo Standard chamada PointRegistrer com os seguintes parâmetros principais:

- Período de retenção da mensagem : 4 dias;
- Tempo limite de visibilidade padrão: 30 segundos;
- Tamanho máximo da mensagem: 256 KB;
- Atraso de entrega: 0 segundos;e
- Tempo de espera de recebimento da mensagem: 10 segundos

O período de retenção de mensagem é o tempo em que pode-se guardar na fila essa mensagem para ser consumida, caso contrário ela desaparecerá. O tempo limite de visibilidade padrão trata de deixar invisível para outros consumidores, por um tempo determinado a mensagem consumida e essa mensagem volta a ser visível na fila se passar do tempo. O atraso de entrega é quando a fila demora um tempo para colocar a mensagem mandada para ela. O tempo de espera de recebimento da mensagem significa que a fila vai aguardar um tempo antes de retornar ao usuário ou a API os elementos contidos nela.

4. API Register Point

A api Register Point tem como objetivo ser uma camada de controle para efetuar de forma organizada e eficiente o consumo de avalancha requisições de marcação de ponto. A tecnologia usada para desenvolvê-lo é o framework **NodeJs** com o **Typescript**, usando alguns conceitos da arquitetura *"SOLID"* e da programação Orientada Objeto com as seguintes bibliotecas baixadas pelo gerenciador de instalação ***Yarn***:

- **Express**: um gerenciador de requisições de diferentes verbos HTTP em diferentes URLs;
- **Axios**: uma biblioteca de requisição HTTP dentro do Nodejs;
- **Dotenv**: uma biblioteca que carrega as variáveis de ambiente;
- **Express-async-errors**: um módulo do express para tratar os erros;
- **Aws-sdk**: fornece uma conexão Api Javascript para os serviços da AWS;

O sistema operacional usado é o ***Linux*** e as disposições das pastas do projeto seguem as seguintes configurações como mostra a figura 1:

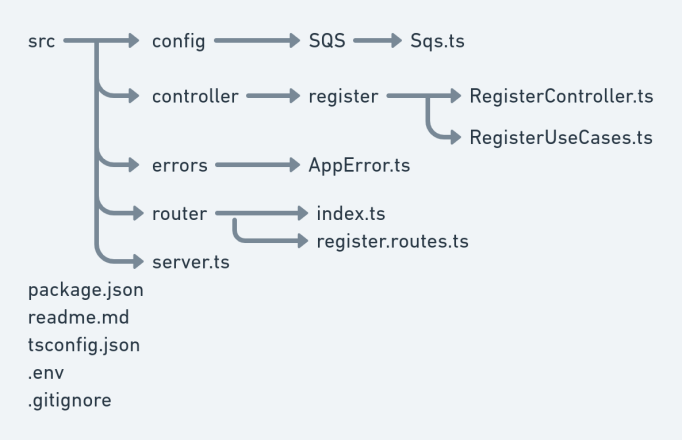

Figura 1. Disposição das Pastas

Na pasta **src** está contido todas as camadas pelo qual passará a execução das requisições do projeto: **config**: nela foi criado a pasta SQS que possuía o arquivo Sqs.ts, onde está escrita os métodos e as configurações de acesso ao api da AWS para uso do serviço da fila.

controller: nesta pasta temos uma pasta ***"registrer"*** que possui os arquivos ***registrerController*** e ***registrerUseCases***. o arquivo ***registerController*** tem como funcionalidade de agrupar ou reunir todas as lógicas sequenciais antes do consumo da requisição. Por exemplo, nesta pasta, é chamado às funções de conexão a fila, verificado os conteúdos desta fila e depois é mandado para consumo. A resposta do consumo, em seguida, é montada dentro deste arquivo e enviado pela requisição HTTP aberta. No arquivo ***registreUseCases***, é feita a persistência ou o consumo em particular das requisições extraídas da fila tratada no arquivo ***registerController***. O ***registerUseCases*** manda de volta o resultado do consumo para o ***registerController***

errors: Esta pasta contém a configuração de apresentação ou tratamento de erros que poderão acontecer ao decorrer da execução da api. **router**: Nesta pasta, é apontada a pasta habilitada para resolver uma determinada requisição interceptada pelo servidor rodando.

O arquivo ***server.ts*** reúne todas as configurações que inicializa o servidor para acertar ou não, qualquer requisição direcionada a sua porta. Neste projeto a porta escolhida é o **3003** e poderia ser acessível pelo seguinte link `http://localhost:3003` com o servidor rodando.

O código fonte desta api está disponibilizado no github: `https://github.com/IbukunChife/Senior-Desafio.git`

Basta clonar essa pasta no seu sistema operacional, entrar na pasta e executar ***"yarn install"*** para instalar as dependências.

O funcionamento da api segue as seguintes etapas: todas as requisições de marcação de ponto que serão feitas entram automaticamente em uma fila única na AWS com as configurações citadas acima. Em seguida, a api vai requisitar o tamanho dos elementos dentro da fila. Por cada elemento neste instante a fila retornará se acionada uma requisição de consumo das informações nela. Quando essa informação é consumida com

sucesso, é requisitado o delete da informação na fila, até que o tamanho total anteriormente fornecido pela fila seja percorrido. A api devolverá, ao término, os resultados de todas execuções realizadas com sucesso das requisições contida na fila (como ilustra a figura 2):

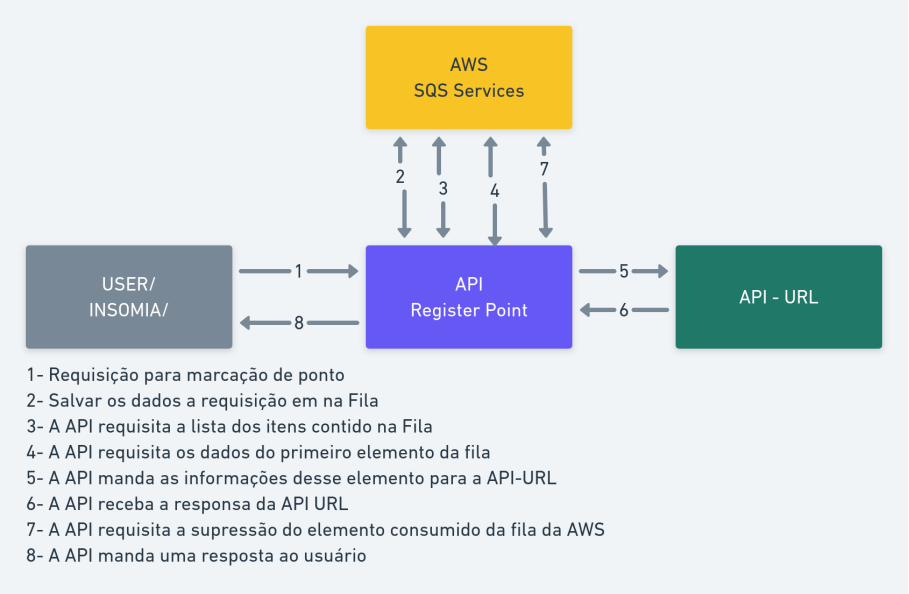

Figura 2. Fluxo de funcionamento da API Register Point

5. Conclusões

A lógica de desenvolvimento da api Register Ponto satisfaz os requisitos que permitem o descongestionamento dos servidores e inclusive a segurança dos dados das requisições. A api aceita todas as requisições e espera um determinado tempo antes de iniciar o consumo destas requisições. Isto garante que todas as requisições, inclusive o próprio último seja considerado no processo de consumo. As funcionalidades da fila permitirão que uma requisição consumida com uma outra api seja invisível para uma outra api Register Point. O que assegura, a não duplicidade de requisição ao lado consumidor ou provavelmente no Banco de Dados. O aspecto que precisa ser melhorado na api Register Point, é a forma pelo qual, a resposta final do consumo dos dados, esteja apresentada. Isto é, a forma pelo qual, o usuário que marca o seu ponto possa ter apenas o status de sucesso da sua ação, e não receber a lista de sucesso de todos os elementos contidos na fila.

Referências

[1] Luiz Duarte, Tutorial de Node.js com filas na AWS SQS. Fonte: https://www.luiztools.com.br/post/tutorial-de-node-js-com-filas-na-aws-sqs/

[2] Amazon Simple Queue Service, Documentation . Fonte: https://docs.aws.amazon.com/pt_br/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

[3] Wesley WILLIANS, Full Cycle. Amazon SQS: Trabalhando com filas na AWS. Fonte: https://www.youtube.com/watch?v=3M4Vyf-AwPc