Reavivamento de sistema legado através do uso de API Python

Lucas Lima da Cunha

1. Cenário

No cenário proposto foi apresentado um sistema legado de marcação de pontos cujo problema central foi a demora de processamento de requisições, potencializado pela alta carga de requisições em curtos espaços de tempo, em torno dos horários de entrada e saída de funcionários.

2. Proposta

A partir desta situação foi pensado e desenvolvido uma API, utilizando Python, Flask e algumas outras bibliotecas do Python para intermediar as requisições entre o usuário e o sistema legado de forma que não fosse necessário esperar pela longa conclusão da ação, terceirizando isso para a API, que armazenaria as requisições em um bando de dados próprio e a cada 30 minutos, tempo parametrizado, na primeira requisição por algum usuário, a aplicação leria o seu banco de dados e enviaria as informações pendestes para o sistema legado.

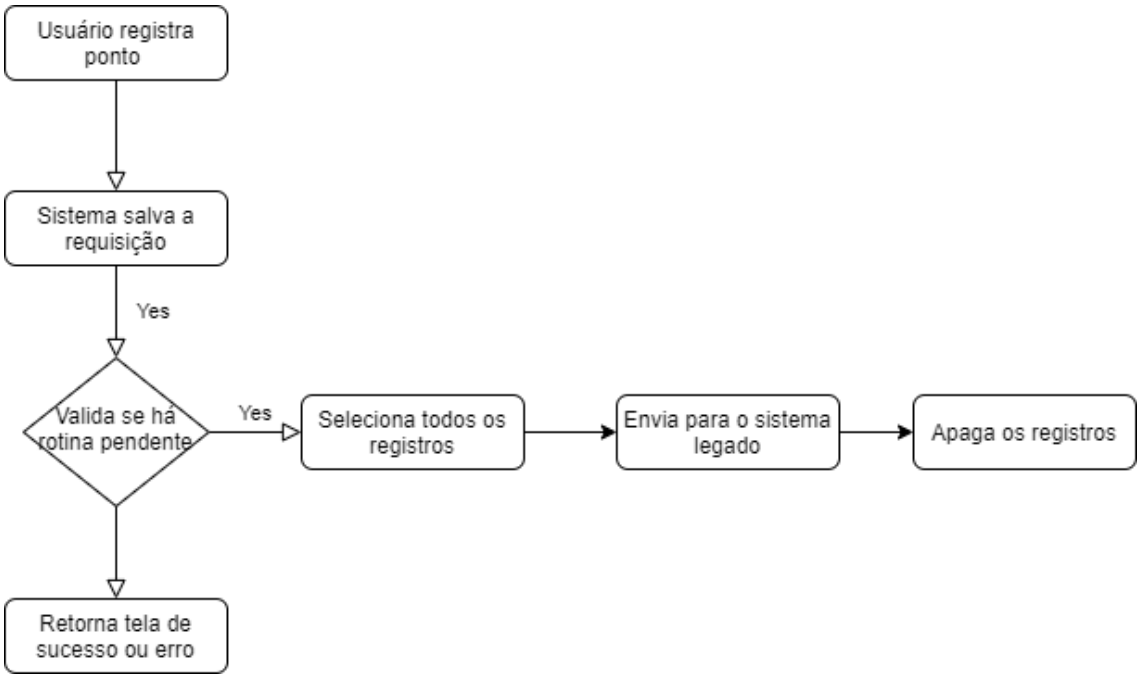

Figura 1 fluxograma

O sistema é composto por quatro arquivos, uma classe python que, cria a conexão com um banco de dados, cria uma tabela nova, e tem um método de registrar

ponto, que segue o fluxograma acima além de rotas para as três telas que compõem o sistema, a tela principal em que é feito o registro, a tela de sucesso e a de erro.

3. Resultado

Como nos testes foi utilizado um banco de dados hospedado no HostGator e a aplicação localmente, o tempo de espera da requisição ainda não é ideal, e pela instabilidade da conexão com o servidor variou entre 1 a 4 segundos por requisição (para comparação, comentando as duas linhas que registravam a requisição no banco de dados foi possível executar 1000 requisições em 3 segundos). Apesar disso, como no cenário proposto foi descrito um tempo de requisição de 10 segundos, já seria uma melhoria considerável levar até 4 segundos para realizar a mesma ação.

4. Próximos Passos e Melhorias

O código foi desenvolvido para ser consumido como um site, porém, algumas alterações nos returns das funções seriam capazes de retornar um JSON, que poderia ser consumido como API por praticamente todo dispositivo, como IoT em uma máquina de ponto.

Fica em aberto também melhorias no código como rodar a aplicação em um servidor aws, isso renderia um grande ganho de performance, utilizando o código back-end e o banco de dados no mesmo serviço, encurtando o tempo de requisições além de todas as vantagens de ser um sistema serverless e facilmente escalável.