Projeto: Monitorador de Webservices Bancários

1. **História do usuário**

As instituições bancárias tem disponibilizado serviços de APIs para a emissão de Boletos, que substituem a comunicação já antiga e tradcional dos arquivos CNAB - que são arquivos texto trocados com as instituições para realizar a comunicação entre sistemas e os bancos.

Como as APIs são serviços SaaS disponíveis em nuvem, ocorrem instabilidades e problemas que impactam a emissão de títulos feitas pelo serviços que as consomem. Nesse contexto, a Tecnospeed se posiciona como um middleware que conecta o sistema a API bancária.

Quando a API bancária está passando por um problema, essa mesma indisponibilidade é apresentada para o nosso cliente pelo nosso serviço, o que gera dúvida nesse cliente se o problema está acontecendo de fato no banco, ou no serviço da Tecnospeed.

Por isso, nós da Tecnospeed gostaríamos de criar uma ferramenta pública que faz o monitoramento dos principais serviços de APIs bancárias, para dar transparência ao cliente sobre onde está ocorrendo o problema: se é em nosso produto, ou na instituição bancária.

Além da Tecnospeed, há milhares de outros desenvolvedores que se conectam com essas APIs bancárias, então além de beneficiar a Tecnospeed e seus clientes, toda a comunidade de desenvolvedores que precisam emitir boletos também serão beneficiadas por essa ferramenta.

2. **Tecnologias a serem utilizadas**

- Backend: NodeJS ou Typescript
- Front-end: VUEJS ou ReactJS
- Banco de dados: Postgrees

3. **Requisitos**

Os bancos nos quais os serviços a serem monitorados devem ser:

- 001 - Banco do Brasil (v2)
- 341 - Itaú (v2 e Francesa)
- 756 - Sicoob (v2 e v3)
- 748 - Sicredi (v2 e v3)
- 104 - Caixa
- 033 - Santander (v2)
- 041 - Banrisul
- 077 - Inter

Todos referentes à Cobrança Bancária. Cada banco e cada versão, deverão ser criadas contas/convênios para cada uma a partir de uma software house e cedente para essa finalidade.

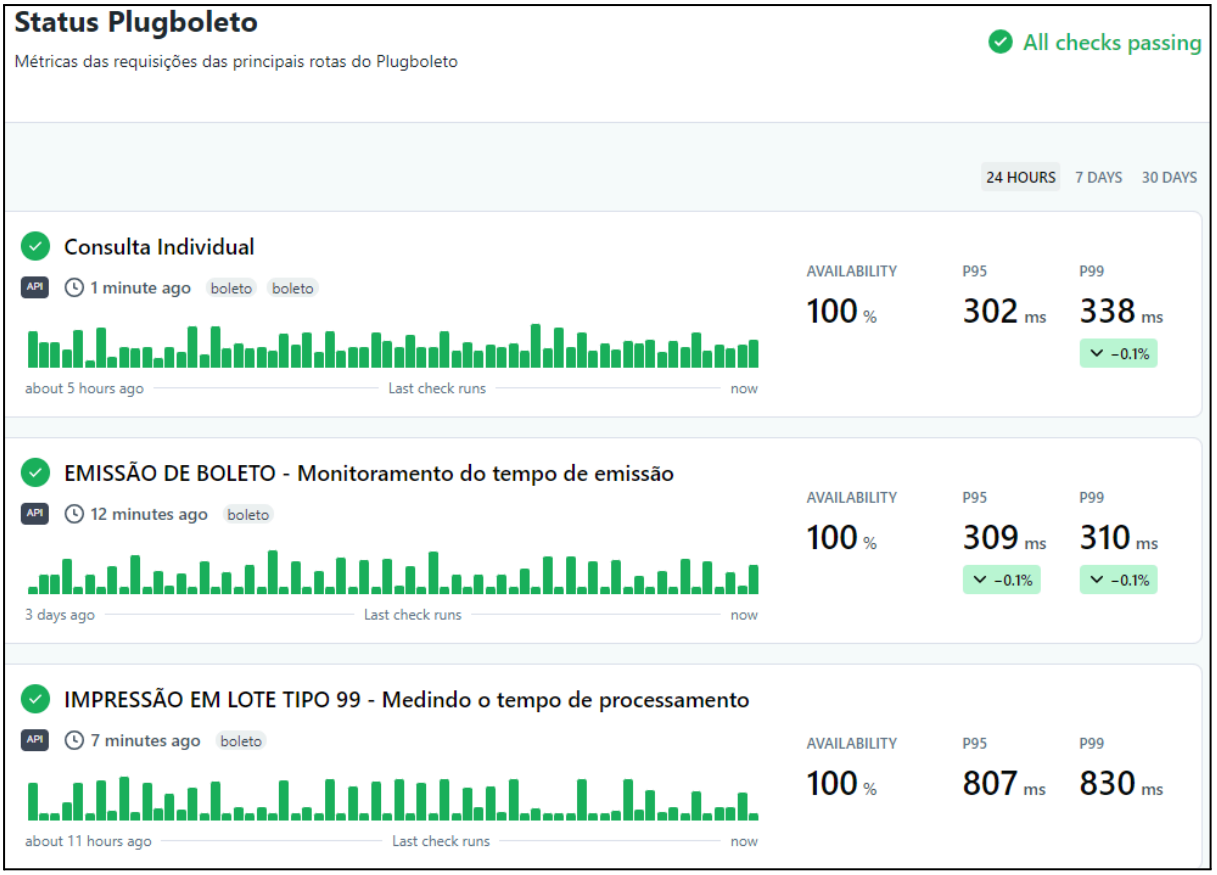
A seguir, listamos os seguintes requisitos obrigatórios:

RQ 01 - Ter o monitoramento dos serviços de Envio/Registro de Boleto

O monitoramento **positivo** é quando a API bancária retornar os status 200, 400, 401, 403 e 422. Enquanto **negativo** quando retornar ECONRESET, EHOUSTUNREACH, 500 e 504.

O monitoramento deve ser dividido por serviço (Cobrança Bancária), por Banco (Banco do Brasil, Itaú,...) e o cliente pode optar por escolher o serviço/banco que ele deseja visualizar.

Exemplos de layout:



https://plugboletostatus.checkly-dashboards.com/?duration=1d

RQ 02 - Ter o monitoramento dos serviços de Consulta de Boleto.

O monitoramento **positivo** é quando a API bancária retornar os status 200, 400, 401, 403 e 422. Enquanto que **negativo** quando retornar ECONRESET, EHOUSTUNREACH, 500 e 504.

RQ 03 - Ter um front-end público que o usuário pode acessar livremente para visualizar os status bancários selecionando por banco/versão.

O front deve exibir um gráfico contendo todas as requisições realizadas no período, segmentado por bancos e exibindo a situação amigável e o tempo de cada requisição como *tooltip*. Ele deve ser responsivo e leve, para ser carregado tanto em computadores, quanto celulares.

RQ 04 - Registrar o histórico das requisições em gráficos que mostrem periodicidade de 24h, 7 dias e 30 dias.

O monitoramento deve exibir tanto se a resposta tem sucesso ou erro, quanto o tempo de resposta de cada requisição. Uma requisição deve ser feita no intervalo de cada 5 minutos e esses registros devem ser gravados.

RQ 05 - Quando ocorrer erros, registrar o histórico desses erros

Deve existir uma tabela para o usuário, onde mostra a resposta retornada pelo banco quando houver um status de erro. Nessa tabela, devem estar os últimos 10 erros

para aquele serviço e banco que foi identificado, bem como a mensagem retornada pelo servidor e a data e hora em que ele ocorreu.

RQ 06 - Ter um banco de dados com os registros para montagem da interface e tratamento dos dados

Criar tabela com todas as requisições realizadas e mantendo como identificadores o tipo (registro ou consulta), código de cada banco, *status code* recebido e data de criação. Além disso, devemos armazenar o tempo da requisição em um outro campo. Dentro de um campo *jsonb*, devemos armazenar o payload da resposta. Criar index para os identificadores para serem consultados.

…ou…

Criar um banco de dados onde cada instituição bancária possuirá 2 tabelas. Uma para consulta e outra para registro (exemplo: “BB_Registro, BB_Consulta, Caixa_Registro, Caixa_Consulta,...);

Dentro de cada tabela deverão existir no mínimo 3 colunas, sendo uma para o horário em que o teste foi realizado, outra para o resultado (online ou offline) e outra para medir o tempo de resposta (em segundos). Novas colunas podem ser adicionadas livremente caso identifique a necessidade;

A cada teste realizado as colunas e tabelas devem ser persistidas conforme o resultado do teste e os dados apresentados no front deverão utilizar os dados destas tabelas.

RQ 07 - Medir a resposta e o tempo da resposta

Em caso de sucesso da requisição (ou seja, o servidor respondeu como esperado), deve-se registrar o status de sucesso aliado ao tempo de resposta do servidor do banco. Em caso de falha, deve ser identificada a falha, bem como o seu tempo de resposta (se houver) e também o registro do erro deve ser feito na tabela conforme RQ 05.

RQ 08 - Ter espaços de publicidade que façam referência ao produto Plugboleto do Plugbank

Devem ter no front-end, ao menos um espaço publicitário onde seja possível inserir um banner (imagem) que contenha um link clicável para uma nova janela que redirecione para o produto Plugboleto: https://tecnospeed.com.br/boleto/

O espaço publicitário deve ser inserido de forma responsiva, sendo exibida também em dispositivos móveis e sem prejudicar o desempenho da página.

O front-end deve seguir a identidade visual da Tecnospeed.

4. Materiais de apoio

Recursos de Referência:

- Monitor SEFAZ da Tecnospeed: https://monitor.tecnospeed.com.br/

Utilizar o Monitor da SEFAZ como referência para a construção do Monitorador Bancário, lembrando-se de utilizar a logo e identidade da Tecnospeed e do Plugbank.

- Documentação do Plugboleto

Utilizar o nosso produto para fazer os disparos de requisições na nossa API. Assim, utilizam-se semelhantes para envios. Documentação pública: atendimento.tecnospeed.com.br.

- Repositório de demonstrações de uso do Plugboleto

Caso haja dúvidas sobre a forma de implementar o produto, possuímos um repositório de códigos que podem ser utilizados como exemplo. Repositório público: https://github.com/tecnospeed/Componente-Boleto/tree/master/demonstracoes/API

- PageSpeed Test: PageSpeed Insights (web.dev)

Para aprimorar a escalabilidade e responsividade, pode ser utilizada a ferramenta PageSpeed Test para avaliar o projeto. Lembrando que para que ele funcione, é necessário que o projeto esteja hospedado em um ambiente em nuvem.

- Identidade visual da Tecnospeed

Link com arquivos da identidade visual da Tecnospeed para utilizar no projeto: https://drive.google.com/drive/folders/1Oky--tKCYfEZBj5p9DtJp0EMY6fVSJmX?usp=drive_link

5. Diferenciais no projeto

Serão diferenciais nos projetos apresentados:

- Ter a maior quantidade de bancos e serviços disponíveis no produto (conforme a lista apresentada nos requisitos).
- Fidelidade no uso das tecnologias listadas aqui com as apresentadas no projeto
- Maior proximidade/similaridade com o layout e design system da Tecnospeed
- Aplicação de técnicas de escalabilidade dos serviços, para suportar grandes capacidades de acesso
- Responsividade do front-end para dispositivos móveis
- Ter aplicação de técnicas de SEO (Search Engine Optimization) para facilitar o usuário encontrar o monitor no Google.