Desenvolvedor Full Stack IA - Desafio prático

No mundo de gerenciamento de *software* existe algo terrível conhecido como inferno das dependências (*dependency hell*). Quanto mais o sistema cresce, mais pacotes são adicionados a ele e maior será a possibilidade de, um dia, você encontrar-se neste poço de desespero.

Como uma solução para este problema, existe um conjunto simples de regras que ditam como os números das versões são atribuídos e incrementados. Por exemplo: considere o formato de versão X.Y.Z (Maior.Menor.Correção). Correção de falhas (*bug fixes*) que não afetam o sistema, incrementa a versão de Correção, adições/alterações compatíveis com as versões anteriores do *software* ou *firmware* incrementa a versão Menor, e alterações incompatíveis com as versões anteriores incrementa a versão Maior.

No contexto de IoT, a atualização *Over-The-Air*, usualmente chamada de OTA, facilita a atualização dos dispositivos pois permite o envio de atualizações de *firmware* e *software*, para dispositivos embarcados de forma remota. Aqui no Instituto Atlântico temos tido muito trabalho em gerenciar as versões semânticas dos arquivos que geramos para nossos dispositivos IoT, que recebem atualização via OTA. Vimos que você tem potencial para construir um *software* para controle de versão dos nossos arquivos *firmware* e muito provavelmente fazer parte do nosso time!

Aqui vai algumas estórias de usuário para que você tenha noção do que estamos pensando:

- Eu, como engenheiro, gostaria de manter (CRUD) *firmware* com **nome do projeto**, **versão** e **nome da placa compatível**;
- Eu, como engenheiro, gostaria de ter a garantia que **somente usuários autorizados** podem fazer *upload* e *download* dos arquivos de *firmware*.
- Eu, como engenheiro, gostaria de **garantir a integridade*** do arquivo de *firmware*, a fim de impedir que alguém não autorizado tenha acesso às informações contidas na imagem (.bin)
- Eu, como engenheiro, gostaria que ao abrir a plataforma de controle de versão de *firmware*, o sistema listasse para mim todos os arquivos (.bin ou .zip) por **projeto** e **placa compatível;**

***Você é livre para escolher qualquer estratégia para proteger a integridade dos arquivos: usar uma biblioteca de criptografia na linguagem que você escolheu; compactar o arquivo em .zip e protegê lo com senha, etc.*

Bem simples né? Pra deixar esse desafio mais interessante, considere algumas regras de negócio:

- O nome do arquivo de *firmware* deve ser salvo no seguinte formato: **nome_do_projeto_v0_0_1.bin**
- O sistema não deve aceitar o *upload* de um arquivo em formatos diferente de **.bin** ou **.zip** (ou formatos resultantes de um processo de criptografia)
- A versão do arquivo deve refletir a versão Maior (MAJOR), Menor (MINOR) e de Correção (PATCH)

Mas o que realmente interessa pra nós?

Queremos avaliar:

- A qualidade e reusabilidade do código que você escreve, não o funcionamento;
- Boas práticas do paradigma de programação que você escolher (Orientação a Objetos, Programação Funcional, etc);
- O cenário apresentado possui 4 estórias de usuário, nós ficaremos satisfeitos se você escolher **pelo menos 2 estórias** e implementá-las, mas quanto mais implementar, mais felizes ficaremos. Cabe a você como Analista, decidir as estórias que são mais importantes e factíveis dentro do prazo.

Você ganhará pontos **extras** se:

- Dockerizar os componentes da sua solução;
- Fazer o *deploy* da sua aplicação (heroku, AWS ou afins);
- Disponibilizar o código em um repositório (github, bitbucket ou afins)

Sabemos que o tempo pode ser curto. Então vamos fazer assim: faça aquilo que você conseguir, do melhor jeito que você puder! Pode entrar em contato com a gente se tiver **qualquer** dúvida ou tiver algum problema com relação ao tempo de entrega do desafio.

Ah! E caso venha te interessar, esta é a *stack* de tecnologias que damos preferência, porém se sinta livre para usar a stack que você desejar:

- Java, Node JS e Python no *backend*;
- Angular, React e Vue.JS no *frontend*;

Boa sorte! Estamos ansiosos (: