

Web Coding Challenge

DESCRIÇÃO DO PROJETO

Você está planejando uma grande conferência de programação e recebeu diversas propostas de palestras, mas você está com problemas para organizá-las de acordo com as restrições de tempo do dia. Então, você decide escrever um programa para fazer isso por você.

FUNCIONALIDADES

1. Criar um cadastro das palestras, sendo uma estrutura de CRUD (Create, Read, Update e Delete) em um banco de dados.
2. Criar uma API REST com um endpoint que receba um arquivo contendo uma listagem de palestras e seus respectivos tempos de duração (Dados de Entrada existentes no final desse arquivo).
3. O endpoint criado deve organizar a listagem de palestras recebidas de acordo com as seguintes regras:
 - A conferência tem várias tracks, cada qual tendo uma sessão pela manhã e outra pela tarde.
 - Cada sessão contém várias palestras.
 - Sessões pela manhã começam às 9h e devem terminar às 12h, para o almoço.
 - Sessões pela tarde começam às 13h e devem terminar a tempo de realizar o evento de networking.
 - O evento de networking deve começar entre 16h e 17h.
 - Nenhum dos nomes das palestras possui números.
 - A duração de todas as palestras são fornecidas em minutos ou definidas como lightning (palestras de 5 minutos).
 - Os palestrantes serão bastante pontuais, então não há a necessidade de intervalos entre as palestras.
4. Criar uma aplicação simples de frontend utilizando Ruby on Rails, mas podendo ser também em Vue.js, onde o usuário possa cadastrar, alterar, remover as palestras ou fazer o upload de um arquivo e submetê-lo para o endpoint descrito no ponto anterior. A resposta do endpoint deve ser usada para exibir a programação do evento.

ESPECIFICAÇÕES

Você deve produzir uma solução para o problema acima utilizando **Ruby (Versão 3.0.2)** e banco de dados **PostgreSQL (Versão 12 em diante)**.

As respostas da API devem seguir o padrão do REST usando JSON, não definiremos um padrão para a resposta, logo sinta-se livre para montá-las da melhor forma possível e seguindo os padrões da comunidade. No final deste arquivo será descrito um exemplo de caso para o problema, junto com uma solução para o mesmo.

Também não será definido um modelo de dados sendo somente as informações contidas no arquivo de exemplo como obrigatórias, logo sinta-se livre para criá-lo da melhor forma possível e seguindo as boas práticas de modelagem de dados.

Além de verificar se sua solução satisfaz a especificação descritas, também avaliaremos outros aspectos, como o design de sua solução e seu domínio do paradigma da linguagem. Nós esperamos que você encaminhe um código que acredite ser de qualidade, um código que funcione e que tenha sido evoluído no decorrer de seu desenvolvimento.

Outro requisito é o envio dos testes que você produziu para verificar sua solução. Independente de serem feitos antes ou depois de criada a implementação, queremos ter a chance de observar sua habilidade em produzi-los e verificar as regras do problema.

ENTREGA

1. Crie um repositório público no Github com o código de sua aplicação;
2. Crie um arquivo **README** na raiz do projeto descrevendo instruções de configuração e build caso existam e listando as bibliotecas usadas no projeto;
3. Caso tenha algum comentário ou considerações sobre sua implementação, adicione no README anteriormente criado.
4. Envie por email o link do seu repositório para o email: alexandre.melo@stant.com.br
5. O prazo máximo para o recebimento da implementação é de **7 dias** contados a partir da data de recebimento deste documento.

CASOS PARA O PROBLEMA

Dados de Entrada:

Diminuindo tempo de execução de testes em aplicações Rails enterprise 60min

Reinventando a roda em ASP clássico 45min

Apresentando Lua para as massas 30min

Erros de Ruby oriundos de versões erradas de gems 45min

Erros comuns em Ruby 45min

Rails para usuários de Django lightning

Trabalho remoto: prós e cons 60min

Desenvolvimento orientado a gambiarras 45min

Aplicações isomórficas: o futuro (que talvez nunca chegaremos) 30min

Codifique menos, Escreva mais! 30min

Programação em par 45min

A mágica do Rails: como ser mais produtivo 60min

Ruby on Rails: Por que devemos deixá-lo para trás 60min

Clojure engoliu Scala: migrando minha aplicação 45min

Ensinando programação nas grotas de Maceió 30min

Ruby vs. Clojure para desenvolvimento backend 30min

Manutenção de aplicações legadas em Ruby on Rails 60min

Um mundo sem StackOverflow 30min

Otimizando CSS em aplicações Rails 30min



Resultado esperado após a organização:

Track A:

09:00 Diminuindo tempo de execução de testes em aplicações Rails enterprise 60min

10:00 Reinventando a roda em ASP clássico 45min

10:45 Apresentando Lua para as massas 30min

11:15 Erros de Ruby oriundos de versões erradas de *gems* 45min

12:00 Almoço

13:00 Ruby on Rails: Por que devemos deixá-lo para trás 60min

14:00 Erros comuns em Ruby 45min

14:45 Programação em par 45min

15:30 Ensinando programação nas grotas de Maceió 30min

16:00 Ruby vs. Clojure para desenvolvimento backend 30min

16:30 Otimizando CSS em aplicações Rails 30min

17:00 Evento de Networking

Track B:

09:00 Trabalho remoto: prós e cons 60min

10:00 A mágica do Rails: como ser mais produtivo 60min

11:00 Aplicações isomórficas: o futuro (que talvez nunca chegaremos) 30min

11:30 Codifique menos, Escreva mais! 30min

12:00 Almoço

13:00 Desenvolvimento orientado a gambiarras 45min

13:45 Clojure engoliu Scala: migrando minha aplicação 45min

14:30 Um mundo sem StackOverflow 30min

15:00 Manutenção de aplicações legadas em Ruby on Rails 60min

16:00 Rails para usuários de Django lightning

17:00 Evento de Networking

