

Hammer Consult - Desafio técnico

2020

1 Apresentação

Os desafios propostos aqui tem o intuito de medir o desempenho do candidato nas tarefas do dia a dia na empresa como um desenvolvedor completo. Esse teste pretende atingir todos os níveis de conhecimento, portanto, faça as tarefas que conseguir e deixe de lado o que não conseguir desenvolver. Neste desafio testaremos o conhecimento do candidato em estilos, layout, HTML5, CSS3, animações, lógica, TypeScript, Angular, básico de back-end, RestAPIs, protocolo HTTP, boas práticas de programação e alguns outros conceitos.

2 Aplicação

Como antes mencionado, a aplicação será separada em duas partes, teste de layout e teste de lógica:

2.1 Teste de layout

Nessa parte, o front-end não fará uso de nenhum serviço ou rota externa, apenas estilos e um pouco de lógica no próprio código do front-end. Disponibilizamos imagens de um protótipo e uma breve explicação funcional na qual a aplicação desenvolvida deverá se basear para a entrega do desafio, simulando o ambiente real de trabalho do candidato.

2.2 Teste de lógica

Já aqui a parte de lógica irá utilizar serviços que o candidato devera criar em um back-end, utilizando linguagem e tecnologia de sua preferência. O back-end terá três rotas principais que serão descritas mais adiante no documento. Especificações de funcionamento e um exemplo funcional do mesmo também serão disponibilizados para o candidato na seção adequada do documento. O front-end da parte de lógica poderá ter a aparência que o candidato desejar, se adequando da melhor forma com os estilos propostos para o teste de layout.

2.3 Desenvolvimento

O projeto deverá ser desenvolvido usando o repositório Git fornecido pela empresa onde se encontra este arquivo. Usando as melhores práticas de versionamento possíveis que o candidato desejar, o candidato deverá fazer um Pull Request da sua branch de desenvolvimento para a "master" deste repositório quando o projeto estiver pronto, sendo avaliado posteriormente por nossa equipe. (O Pull Request não será aceito, apenas avaliado)

2.4 Tecnologias

O candidato deverá utilizar Angular 6+ no front-end, que é o framework mais utilizado pela empresa em seus projetos, com SCSS para estilos e os frameworks de teste padrões do Angular para testes unitários e testes ponta a ponta. Em relação ao back-end, o candidato pode usar a tecnologia que preferir, sem restrições.

2.5 Código

O desenvolvimento do código da aplicação deverá seguir algumas regras. Sempre use todos os recursos disponíveis da linguagem, do framework e das tecnologias da melhor forma, criando a melhor implementação possível de todas as funcionalidades. Lembre-se que os projetos são realizados em time e alguém dará manutenção para o seu código posteriormente, portanto, tenha em mente boas práticas de programação e crie um código legível por outros programadores. Crie testes unitários e testes de ponta a ponta para o front-end, para garantir a funcionalidade da sua aplicação mesmo que alguém desenvolva outras funcionalidades nela.

3 Teste de layout

Para o teste de layout, o candidato deverá desenvolver um velocímetro digital veicular, conforme figura presente na próxima página.



Figura 1. Painel virtual automotivo

Requisitos do teste:

1. O velocímetro deverá ter animações de movimentação, como se estivesse em um veículo de verdade.
2. Os ponteiros deveram se mexer simulando a aceleração e desaceleração sem freio de um veículo real com uma marcha apenas, usando as animações do painel detalhadas neste [vídeo](#). Apenas o velocímetro deve ser animado, todas as outras animações do painel não precisam ser levadas em consideração.
3. As rotações do motor do veículo devem acompanhar seu desenvolvimento no marcador de velocidade.
4. O centro do painel deverá possuir o conteúdo estático mostrado na figura, uma faixa simbólica onde o carro deve andar.
5. As cores devem ser as mesmas ou similares, não fornecemos cores, portanto, o candidato pode analisar a foto ou usar cores similares que achar parecido.
6. Os ícones não precisam ser os mesmos da imagem mas devem estar na mesma posição e mesma cor com os mesmos efeitos, não precisam ser os exatos mesmos ícones, podem ser quaisquer ícones que o candidato escolher.

4 Teste de lógica

Neste teste, o candidato deve desenvolver um back-end com três rotas, servindo três serviços que o front irá utilizar para montar tabelas e formulários dinâmicos.

4.1 Tabelas

Para as tabelas, o back-end deverá retornar um array de objetos com o número de propriedades variável em cada um deles, e com isso, o front deverá montar uma tabela com o maior número de colunas possível baseado nos resultados retornados pelo back-end, de forma que nenhuma coluna fique vazia em todas as linhas. O front deverá preencher a tabela com os dados, preenchendo em cada linha as colunas com os resultados disponíveis. Caso a coluna em questão não exista nas propriedades do objeto da linha, o candidato pode deixar a coluna em branco.

Requisitos do teste:

1. O número de linhas da tabela deverá sempre ser igual ao comprimento do vetor retornado pelo serviço do back-end.
2. O número de colunas da tabela deverá sempre ser igual ao tamanho do conjunto de todas as propriedades retornadas pelo serviço juntas, sem repetir.
3. O nome das colunas deverá corresponder ao nome de cada propriedade retornada pelo serviço, convertendo o nome da propriedade de Lower Camel Case pra texto padrão, com espaços.
4. Nenhuma coluna poderá ficar vazia em todas as linhas.

4.2 Formulário

Para o formulário, o back-end deverá ter uma rota que retornará um array de objetos, definindo quantos campos o formulário terá, qual o tipo de cada um e qual o valor deles. Os campos podem ser do tipo arquivo, número, texto ou uma seleção de opções predefinidas. O usuário poderá editar esses ver e editar esses valores. No caso do tipo arquivo, o usuário poderá adicionar um arquivo de sua escolha e baixá-lo, se assim desejar.

Caso o tipo seja arquivo, o arquivo virá pré-selecionado e quando o usuário clicar no campo, ele será redirecionado para uma rota do back-end que permitirá que o usuário baixe o arquivo em questão.

Os estilos ficam por conta do candidato, sempre criando uma interface aceitável e moderna, condizente com a etapa anterior do desafio, com o mesmo padrão de cores e identidade visual. É possível pegar alguma base assistindo o resto do [vídeo](#) citado no teste de layout anterior.

Requisitos do teste:

1. O formulário deverá ter exatamente o mesmo número de campos retornados pelo serviço do back-end.
2. Os campos do formulário deverão ter o respectivo tipo retornado pelo serviço.

3. Cada campo deverá ter seu respectivo título retornado pelo serviço.
4. Os campos deverão mostrar o valor retornado pelo serviço e permanecer desabilitados.
5. Os campos deverão ter interações especiais dependendo do tipo, como o tipo arquivo levando pra um link de download do mesmo.

4.3 Back-end

O back-end não possui muitas restrições, pode ser desenvolvido como o candidato quiser desde que tenha o mesmo comportamento do back-end de exemplo hospedado pela empresa presente neste [link](#).

Como já mencionado anteriormente, tecnologia livre, o candidato pode usar o que preferir.

Lembre-se que o candidato não deve em hipótese alguma usar o back-end de exemplo em seu projeto, pode usar para testes de desenvolvimento, mas deve entregar uma versão usando o próprio back-end criado e hospedado por conta própria.

O back-end deverá redirecionar para o Swagger quando acessarmos a raíz de seu endereço, assim como o back-end que fornecemos de exemplo para o candidato.

5 Hospedagem

O usuário deverá hostear o front-end e o back-end do projeto em domínios diferentes e informar a empresa o link para as duas partes. Também deverá colocar essas informações no README.MD do projeto.

5.1 Front-end

A SPA desenvolvida deverá ter seu front-end hospedado na empresa de hospedagem que o candidato preferir, com rotas funcionais.

5.2 Back-end

O back-end da aplicação poderá ser hospedado também na empresa de preferência do candidato, fazendo as conexões corretas entre as duas partes do projeto (front e back) hospedadas em endereços diferentes.