Nome: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Aproveitamento: \_\_\_\_\_\_  
E-mail: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**AVALIAÇÃO NodeJS**

As atividades descritas a seguir contemplam e pretendem avaliar o aproveitamento do seguinte curso: NodeJS: Inovando com JavaScript no backend, realizado na plataforma Alura durante o programa.

**Proposta:** Analisar e desenvolver um site para cadastramento de médicos no Central do Bem Estar juntamente com API REST para alimentar app mobile.

**Instruções Gerais**

1) A avaliação é *individual* e com consulta (aos cursos, a documentação oficial e o material produzido a partir deles disponível no repositório individual);

2) Baixar o conteúdo de [https://github.com/besilva/pdb-n](https://github.com/myreli/pdb-python)odejs que serve como material de apoio.

3) É **obrigatório** o uso do padrão DAO mostrado no curso, boas práticas de roteamento, usando a url, verbo HTTP e o Content Type, validação dos dados vindo do front, websocket, middlewares, laços, estruturas condicionais, funções e outros recursos vistos no curso. Além disso, espera-se código otimizado, lógica de programação, blocos reutilizáveis e boa organização;

4) O SGBD é escolha do aluno, sendo permitido usar MySQL, MongoDB ou MariaDB.

5) Ao concluir a avaliação, enviar por e-mail para [bernardo.silva@compasso.com.br](mailto:myreli.barros@compasso.com.br) o link do repositório, o arquivo zip do projeto (SeuNome.zip) e quaisquer observações necessárias com o assunto **[PDB][2018] Avaliação NodeJS**.

*“Just do it” - Shia Labeouf*

**Atividade:**

1) verificar a instalação de Git, Visual Studio Code, NodeJS e o banco escolhido na máquina e reportar quaisquer problemas;

2) clonar o repositório privado individual do projeto e baixar o material para a avaliação, lá você encontrará um os comandos de insert caso use um banco SQL e os json caso opte por NoSQL para popular o banco de dados;

3) Telas:

Segue link para **ideia** de como deve ser as telas(não é obrigatório seguir as mesmas):

<https://marvelapp.com/40agcjg>

A página de cadastro foi feito um esboço, para poupar tempo de fazer front para se focar mais no NodeJS e se encontra no repositório do git.

4) Contextualizando:

Seu cliente é uma rede chamada Central de Bem Estar, lá eles cadastram médicos para agendarem consultas, para uma melhor experiência do usuário é cadastrado a(s) especialidade(s) e a(s) ênfase(s) do médico para o usuário escolher o que melhor pode atender seu problema.

Exemplo: Doutora Maria tem especialidade em Traumatologia com ênfase na parte da coluna e do ombro.

Por outro lado doutor João tem especialidade em Ginecologia e Obstetrícia, porém não tem ênfase em nada.

5)Exercícios:

Os itens obrigatórios para o do site são:

* 1. Cadastro, busca, listagem, edição e remoção dos médicos
  2. Cada médico tem um CRM, nome, data de nascimento, **dados para contato** e **Especialidade(s)**
  3. Sendo os **dados para contato:** telefone e endereço
  4. Sendo a **Especialidade:** nome e as **Ênfase(s)** daquela especialidade
  5. Sendo a **Ênfase:** apenas o nome

Resumindo quando um médico é cadastrado é obrigatório cadastrar seus dados pessoais (CRM, nome, data de nascimento) e um dado de cadastro (telefone e endereço), sendo possível mais de um dado de cadastro, além disso deve ser obrigatório associar o médico a no mínimo uma Especialidade já cadastrada ou a várias especialidades que ele tiver, quando a especialidade é selecionada deve mostrar as ênfases relacionadas a ela(s) para ele selecionar caso tenha alguma.

Na parte da busca deve ser possível filtrar os resultados pelo nome e/ou especialidade do médico.

Ao clicar em editar deve ser possível acessar o formulário de cadastro com os dados do médico selecionado

Ao clicar em deletar deve ser enviado para uma rota onde será executado o comando de deletar e após a conclusão deve ser redirecionado de volta para a listagem.

Além disso, sempre que um médico for adicionado o sistema deve exibir um alerta para os outros funcionários não perderem tempo tentando cadastrar repetidamente o mesmo médico.

Por fim, seu cliente pediu para ter um login para acessar as páginas do sistema referentes ao médico, logo deve ser feito uma validação para verificar se o usuário está logado conforme ele acessa uma rota, caso não esteja, o mesmo deve ser redirecionado para o login.

6)Desafios:

Esta etapa é **extra**, só deve ser feito se o aluno quiser e sobrar tempo.

**Refinando ainda mais a busca:**

Atualmente, o sistema só permite buscar pelo nome do médico e pela especialidade, mas para melhorar ainda mais a busca seria interessante, ao selecionar uma especialidade ter um select para selecionar as ênfases daquela especialidade para achar o médico especifico para o problema.

**Deletando usando Ajax:**

**IMPORTANTE!** Caso opte por fazer esse desafio, faça um commit antes de começa-lo para garantir que a versão do delete implementada anteriormente não seja comprometida, e me avise no email da prova que no git constam as duas versões.

O desafio: O comportamento padrão da aplicação é que ao clicar no botão deletar do médico na página a aplicação é direcionada para uma rota que executa o deletar e retorna pra listagem. O desafio é implementar o delete de uma forma mais interessante, usando o Ajax passando o verbo **Delete** do http, caso tenha sucesso o registro da tabela de listagem deve ser removido e caso ocorra um erro deve mostrar que houve um problema interno através de um alerta para o usuário do sistema.