integrado

Teste Back-end

PARTE 1

Criação de script para ler API JSON e popular o banco de dados:

O objetivo do teste é varrer uma API que retorna todas as universidades contidas em cada país da lista fornecida e salvar estas informações no banco de dados.

Requisitos:

- O código deve ser desenvolvido utilizando Node.JS e MongoDB.
- API para busca das informações: http://universities.hipolabs.com/search?country=uruguay
- Devem ser listadas as universidades dos seguintes países:

```
[
  "argentina",
  "brasil",
  "chile",
  "colombia",
  "paraguai",
  "peru",
  "suriname",
  "uruguay"
]
```

- Criar uma collection no MongoDB para armazenar as universidades separadamente.

integrado

PARTE 2

Criação de API REST para gestão das universidades cadastradas

O objetivo do teste é criar uma API que viabilize um CRUD (create, retrieve, update, delete) das universidades anteriormente cadastradas no MongoDB.

Requisitos:

- Método para listagem de Universidades:
  - GET /universities
    - Lista todas as universidades do banco de dados.
    - Retornar os campos _id, nome, país e estado.
  - Permitir filtro por país. Exemplo GET /universities?country=brazil
  - O método de listagem deverá retornar no máximo 20 registros por página.
  - Deverá ser possível informar a página na requisição para ter acesso a todos os registros.

- Método para buscar a universidade por _id.
  - GET /universities/:id (busca dados da universidade indicada pelo _id)
    - Retorna todos os dados armazenados da universidade em questão.

- Método para cadastro de Universidades:
  - POST /universities. Campos:
    - alpha_two_code
      - Sigla do país com 2 caracteres
    - web_pages
      - Lista com as URL's da universidade
    - name
      - Nome da universidade por extenso
    - country
      - Nome do país por extenso
    - domains
      - Lista de domínios da universidade
    - state-province
      - Sigla do estado onde fica a universidade se houver
  - Utilizar como referência os campos recebidos da API da parte 1 e já armazenados no banco de dados.
  - Para evitar duplicidade no banco de dados, deverá impedir que seja cadastrada uma nova universidade caso já possua alguma com o mesmo País, Estado e Nome.

- Método para atualização de Universidades:
  - PUT /universities/:id. Campos:
    - web_pages
      - Lista com as URL's da universidade

        - name
            - Nome da universidade por extenso
        - domains
            - Lista de domínios da universidade
- Método para remoção de Universidades:
    - DELETE /universities/:id

OBSERVAÇÕES

- Poderá utilizar as bibliotecas que desejar para o Node.JS.
- Seu projeto deve conter um README.md.
- Seu projeto deve ser enviado para um repositório GIT para avaliação. Faça commits regulares.
- O sistema rodará em um ambiente com ubuntu server 20.04 LTS em uma máquina virtualizada na Amazon AWS.
- Você deverá se preocupar em como esse serviço rodará em produção. Esperamos poder rodar o seu software em um servidor limpo seguindo somente as instruções fornecidas no README.md do projeto.

Avaliação

A avaliação será feita por desenvolvedores acostumados em colocar integrações no ar. Consideraremos a simplicidade da sua solução em conjunto com a efetividade da mesma. Não existe um único jeito certo de se fazer um sistema contanto que este seja efetivo, eficiente e fácil de manter. Atente aos seguintes pontos quando estiver desenvolvendo seu projeto:

- Simplicidade: se dois métodos resolvem o mesmo problema, preferimos o mais legível e simples. Atente para a facilidade de deploy e operação da sua aplicação.
- Desempenho: todos os testes rodarão em uma instância padrão na Amazon, o desempenho relativo da sua solução frente a outras será considerado. Lembrando que não estamos em busca do código mais rápido, mas definitivamente não queremos operar o mais lento
- Code style: não preferimos jeito A ou jeito B, desde que haja consistência ao longo do código.