Atividade 1
Node.js + Express.js
REST

Universidade Federal do Ceará
Projeto de Interfaces Web
Prof. Victor Farias

- **Data de entrega**: 09/12/20
- **Modo de entrega**
  - Código + vídeo de apresentação zipado pelo moodle.
- **Instruções para o código**
  - Excluir node_modules e adicionar ao arquivo zip final
- **Instruções para o vídeo de apresentação:**
  - Somente as funcionalidades que foram mostradas no vídeo receberam nota
  - Não precisa falar, basta gravar a tela
    - Lembrar de usar uma qualidade um pouco mais baixa pois o moodle só aceita arquivos até 20mb
    - Pode usar algum software como o obs ou gravar a tela do computador o celular
  - O objetivo é demonstrar as funcionalidades pedidas usando o postman ou software similar.
  - Roteiro do vídeo
    - Mostrar rapidamente a estrutura de pastas com o package.json criado
    - Inserir usuário pelo endpoint **POST** /api/usuários
    - Buscar todos usuários com **GET** api/usuarios
    - Buscar um usuário que existe com **GET** /api/usuarios
    - Buscar um usuário que não existe com **GET** /api/usuarios
    - Remover usuário com DELETE /api/usuários/:id
    - Buscar novamente todos os usuários **GET /**api/usuarios para mostrar que usuário foi removido
    - Executar mesmo roteiro acima para posts
  - Dica: Deixa todas as requisições prontas no postman e, depois, é só gravar executando elas.

- Tempo máximo do vídeo: 5 minutos

Questões

1. Montar projeto usando npm e estrutura de pastas mostrada em sala (1 ponto)
2. **Endpoints.** Construa os endpoints que se recebe, respeitando **exatamente** as estruturas do JSON de exemplo e **exatamente** nessas rotas. Os dados são armazenados em lista como visto em sala de aula.

Usuários

a. **POST** /api/usuarios (1 ponto)
   i. Recebe usuário e armazena. Ex:

```
{
   “id”: “1”,
   “nome”: “Victor”,
   “email”: “victor.aefarias@gmail.com”,
   “senha”: “123”
}
```

   ii. Retorna o mesmo usuário

b. **GET** /api/usuarios (1 ponto)
   i. Retorna todos os usuários**.** Ex:

```
[
   {
      “id”: “1”,
      “nome”: “Victor”,
      “email”: “victor.aefarias@gmail.com”,
      “senha”: “123”
   },
   {
      “id”: “5”,
      “nome”: “João”,
      “email”: “joao @gmail.com”,
      “senha”: “456”
   },
   ...
]
```

c. **GET** /api/usuarios/:id (1,5 pontos)
   i. Retorna usuário com um dado id**.** Ex para /api/usuarios/1

```
{
   “id”: “1”,
   “nome”: “Victor”,
   “email”: “victor.aefarias@gmail.com”,
   “senha”: “123”
}
```

   ii. Caso usuário não exista, retorna erro 404

d. **DELETE** /api/usuarios/:id (1 ponto)
   i. Remove usuário com id dado

Posts

e. **POST** /api/posts (1 ponto)
   i. Recebe post e armazena. Ex:

```
{
   “id”: “1”,
   “texto”: “Oi, tudo bem?”,
   “likes”: “6”
}
```

   ii. Retorna o mesmo post.

f. **GET** /api/posts (1 ponto)
   i. Retorna todos os posts**.** Ex:

```
[
   {
      “id”: “1”,
      “texto”: “Oi, tudo bem?”,
      “likes”: “6”
   },
   {
      “id”: “5”,
      “texto”: “Tudo bom! E vc?”,
      “likes”: “6”
   },
   ...
]
```

g. **GET** /api/posts/:id (1,5 pontos)

i. Retorna post com um dado id**.** Ex para /api/posts/1

```
{
    “id”: “1”,
    “texto”: “Oi, tudo bem?”,
    “likes”: “6”
}
```

h. **DELETE** /api/posts/:id (1 ponto)

i. Remove post com id dado