Atividade 2

Node.js, Express.js e Mongoose

REST

Universidade Federal do Ceará

Projeto de Interfaces Web

Prof. Victor Farias

* **Data de entrega**: 13/06/21
* **Peso 2**
* **Modo de entrega**
  + Código + vídeo de apresentação zipado pelo moodle
  + O vídeo também pode ser subido no YouTube e envia o link no arquivo zip
* **Instruções para o código**
  + Excluir node\_modules e adicionar os outros arquivos ao 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
  1. Lembrar de usar uma qualidade um pouco mais baixa, caso envie o vídeo pelo moodle, pois o moodle só aceita arquivos de até 20mb
  2. Pode usar algum software como o obs ou gravar a tela do computador ou celular
* O objetivo é demonstrar as funcionalidades pedidas usando o postman ou software similar.
* **Roteiro do vídeo**

**Montando o banco**

* 1. Inserir um primeiro usuário pelo endpoint **POST** /api/usuários
  2. Inserir um post para esse usuário (usar o id do usuário que o banco gerou automaticamente) com **POST** /api/posts
  3. Inserir um segundo usuário com **POST** /api/usuários
  4. Inserir um post para o segundo usuário **POST** /api/posts
  5. Adicionar um comentário desse segundo usuário no post feito pelo primeiro usuário usando **POST** /api/comentarios

**Fazendo buscas**

* 1. Buscar todos os usuários com **GET /**api/usuarios
  2. Buscar apenas o primeiro usuário pelo id com **GET /**api/usuarios/<id\_primeiro\_usuario>
  3. Buscar todos os posts do primeiro usuário com **GET /**api/usuarios/<id\_primeiro\_usuario>/posts
  4. Buscar todos os posts com **GET /**api/posts
  5. Buscar apenas o post do primeiro usuário com **GET /**api/posts/<id\_post> (pegar id na mão)
  6. Buscar todos os comentários do post do primeiro usuário com **GET** /api/posts/<id\_post>/comentarios
  7. Buscar todos os comentários com **GET** /api/comentarios
  8. Mostrar as 3 coleções no MongoDB usando o MongoShell ou alguma aplicação parecida

**Removendo dados**

* 1. Remover o comentário feito usando **DELETE** /api/comentarios/<id\_comentario>
  2. Remover o post do primeiro usuário como **DELETE** /api/posts/<id\_post>
  3. Remover primeiro usuário com **DELETE** /api/usuarios/<id\_usuario>
  4. Mostrar as coleções com os dados removidos
* Dica: Deixar todas as requisições prontas no postman e, depois, é só gravar executando-as.
* Tempo máximo do vídeo: 5 minutos

**Questões**

1. Crie os modelos segundo o seguinte esquema (1 ponto)

**Usuário** (id, nome, email, senha)

**Post** (id, texto, likes, id\_usuario)

**Comentário** (id, texto, id\_post, id\_usuario)

1. Crie as views para renomear \_id para id em todas as entidades e para evitar de enviar senha do usuário para o cliente (1 ponto)
2. **Endpoints.** Construa os endpoints que se recebe, respeitando **exatamente** as estruturas do JSON de exemplo e **exatamente** nessas rotas. Agora os dados serão armazenados no MongoDB.  
    **Usuários**
   1. **POST** /api/usuarios (0,5 ponto)
      1. Recebe usuário e armazena em banco
      2. Exemplo do json de requisição:

{

“nome”: “Victor”,

“email”: “victor.aefarias@gmail.com”,

“senha”: “123”

}

* + 1. Retorna o mesmo usuário
  1. **GET** /api/usuarios (0,5 pontos)
     1. Retorna todos os usuários
     2. Exemplo do json de resposta:

[

{

“id”: “1”,

“nome”: “Victor”,

“email”: “victor.aefarias@gmail.com”,

“senha”: “123”

}**,**

{

“id”: “5”,

“nome”: “João”,

“email”: “joao @gmail.com”,

“senha”: “456”

}**,**

**...**

]

* 1. **GET** /api/usuarios/:id (0,5 ponto)
     1. Retorna usuário com um dado id
     2. Exemplo json de resposta para /api/usuarios/1:{

“id”: “1”,

“nome”: “Victor”,

“email”: “victor.aefarias@gmail.com”,

“senha”: “123”

}

* + 1. Caso usuário não exista, retorna erro 404
  1. **GET** /api/usuarios/:id/posts (1 ponto)
     1. Retorna todos os posts do usuário com o id dado.
     2. Exemplo json de resposta /api/usuarios/1/posts:

[

{

“id”: “1”,

“texto”: “Oi, tudo bem?”,

“likes”: “6”,

“id\_usuario”: 1

}**,**

{

“id”: “8”,

“texto”: “Olá?”,

“likes”: “8”,

“id\_usuario”: 1

}**,**

**...**

]

* 1. **DELETE**  /api/usuarios/:id (0,5 ponto)
     1. Remove usuário com id dado

**Posts**

* 1. **POST** /api/posts (0,5 ponto)
     1. Recebe post e armazena em banco
     2. Exemplo json da requisição:

{

“texto”: “Oi, tudo bem?”,

“likes”: “6”,

“id\_usuario”: 1

}

* + 1. Retorna o mesmo post.
  1. **GET** /api/posts (0,5 ponto)
     1. Retorna todos os posts
     2. Exemplo json de resposta:

[

{

“id”: “1”,

“texto”: “Oi, tudo bem?”,

“likes”: “6”,

“id\_usuario”: 1

}**,**

{

“id”: “5”,

“texto”: “Olá amigos!”,

“likes”: “4”,

“id\_usuario”: 2

}**,**

**...**

]

* 1. **GET** /api/posts/:id (0,5 ponto)
     1. Retorna post com um dado id
     2. Exemplo json de retorno para /api/posts/1{

“id”: “1”,

“texto”: “Oi, tudo bem?”,

“likes”: “6”

}

* 1. **GET** /api/posts/:id/comentarios (1 ponto)
     1. Retorna todos os comentários do post com id dado
     2. Exemplo json de resposta para /api/posts/1/comentarios

[

{

“id”:6,

“texto”:”Tudo certo e contigo?”,

“id\_post”: 1,

“id\_usuario”: 2

},

{

“id”:6,

“texto”: ”Tudo blz ma!”,

“id\_post”: 1,

“id\_usuario”: 3

}

...

]

* 1. **DELETE**  /api/posts/:id (0,5 ponto)
     1. Remove post com id dado

**Comentarios**

* 1. **POST /**api/comentarios (1 ponto)
     1. Recebe comentário e armazena em banco
     2. Exemplo json de requisição:

{

“texto”:”Tudo certo e contigo?”,

“id\_post”: 1,

“id\_usuario”: 2

}

* 1. **GET** **/**api/comentarios (0,5 ponto)
     1. Retorna todos os posts
     2. Exemplo json de retorno:

[

{

“id”:6,

“texto”:” Tudo certo e contigo?”,

“id\_post”: 1,

“id\_usuario”: 2

},

{

“id”:7,

“texto”: ”Casal lindo!”,

“id\_post”: 2,

“id\_usuario”: 1

}

...

]

* 1. **DELETE** /api/comentarios/:id (0,5 ponto)
     1. Exclui comentário com id dado.