Atividade 3
Node.js, Express.js, Mongoose, BCrypt e JWT
V1.2

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

- **Data de entrega**: 27/01/21
- **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. Gerar o token token a partir do endpoint **POST** /api/usuarios/signin para usuário inserido no ponto 1

3. Inserir um post para esse com **POST** /api/posts para o usuário passando o token de autenticação dele
4. Inserir um comentário para o post do ponto 3 com **POST /**api/comentarios passando o token de autenticação do usuário

   Fazendo buscas
5. Buscar todos os usuários com **GET /**api/usuarios passando o token
6. Buscar todos os posts com **GET /**api/posts passando o token
7. Buscar todos os comentários com **GET** /api/comentarios
8. Mostrar as 3 coleções (usuário, posts e comentários) no MongoDB usando o MongoShell ou alguma aplicação parecida

   Removendo dados
9. Insira um segundo usuário e obtenha o token dele
10. Tente remover o comentário feito no ponto 4 usando **DELETE** /api/comentarios/<id_comentario> com o token do segundo usuário (deve dar erro)
11. Tente remover o comentário feito no ponto 4 usando **DELETE** /api/comentarios/<id_comentario> com o token do primeiro usuário (deve ser ok)
12. Tente remover post feito no ponto 3 usando **DELETE** /api/usuarios/<id_post> com o token do segundo usuário (deve dar erro)
13. Tente remover post feito no ponto 3 usando **DELETE** /api/usuarios/<id_post> com o token do primeiro usuário (deve ser ok)
14. Tente remover o primeiro usuário usando **DELETE** /api/usuarios/<id_usuario> usando o token do segundo usuário (deve dar erro)
15. Tente remover o primeiro usuário usando **DELETE** /api/usuarios/<id_usuario> usando o token do segundo usuário (deve ser ok)
16. Mostrar as coleções com os dados removidos

- Dica: Deixa todas as requisições prontas no postman e, depois, é só gravar executando-as.
- Tempo máximo do vídeo: 5 minutos

Questões

1. Todos os endpoints agora recebem e validam o token JWT menos em POST /usuarios/signin e POST /usuarios. Caso o token seja válido, executa o endpoint normalmente. Caso seja invalido, retornar mensagem de erro invalido para o usuário. (2 pontos)
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.

   <u>Usuários</u>

   a. **POST** /api/usuarios (2 pontos)
      i. Recebe usuário e armazena em banco
      ii. Deve ser armazenada somente a assinatura hash da senha em banco
      iii. Exemplo do json de requisição:

      ```
      {
        "nome": "Victor",
        "email": "victor.aefarias@gmail.com",
        "senha": "123"
      }
      ```

      iv. Retorna o mesmo usuário sem senha

   b. **POST** /api/usuarios/signin (3 pontos)
      i. Recebe email e senha do usuário
      ii. Verifica se a senha recebida corresponde a assinatura hash do usuário no banco de dados
      iii. Exemplo do json de requisição:

      ```
      {
        "email": "victor.aefarias@gmail.com",
        "senha": "123"
      }
      ```

      iv. Retorna o token com o id do usuário como payload

   c. **DELETE** /api/usuarios/:id (1 ponto)

i. Remove usuário com id dado
ii. Só permite a remoção se o usuário a ser deletado for o usuário com o id fornecido no token

d. Manter o resto dos enpoints adicionando o recebimento e validação do token

Posts

e. DELETE /api/posts/:id (1 ponto)

i. Remove post com id dado
ii. Só permite a remoção se o post pertencer ao usuário logado

f. Manter o resto dos enpoints adicionando o recebimento e validação do token

Comentarios

g. **DELETE** /api/comentarios/:id (1 ponto)

i. Exclui comentário com id dado.
ii. Só permite excluir comentário de o comentário pertence ao usuário logado

EXTRA!

Posts

a. POST /api/posts (0,5 ponto)

i. Recebe post, armazena em banco com id do usuário como o id do usuário do token
ii. Exemplo json da requisição:

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

iii. Retorna o mesmo post

Comentarios

a. POST /api/comentarios (0,5 ponto)

iii. Recebe comentário, armazena em banco com id do usuário como o id do usuário do token
iv. Exemplo json de requisição:

```
{
    “texto”:”Tudo certo e contigo?”,
```

```
        “id_post”: 1,
}
```