

FIC 2 – Módulo II

# Aula 13 – Autenticando usuários no Node.js

Criação de páginas  
customizadas com Node.js

*Leonardo Dalvi Pio*

# Autenticando usuários no Node.js

- Sistema de login (utilizando sessões).
- Autenticação de usuário sem utilizar banco de dados.
- No *Replit*, se o pacote/recurso (***pagkage.json***, ***express***, ***express-session***, ***body-parser***, ***path***) não estiver disponível, ele será baixado automaticamente.



# Autenticando usuários no Node.js

- No Replit, crie uma pasta nomeada "Aula 13".
- Dentro da pasta, crie um novo arquivo (New Repl).  
Selecione o Template 'Node.js' e atribua o nome do arquivo "index.js" (arquivo do tipo .js).



# Autenticando usuários no Node.js

- Iniciaremos criando as constantes, importando dos módulos abaixo, utilizando a porta padrão = (443).

```
const express = require('express')
const session = require('express-session')
const bodyParser = require('body-parser')
const path = require('path')

const app = express()
const porta = 443
```

- Incluiremos o comando `'.listen()'` na porta de comunicação (porta 443) com o Servidor, e a mensagem "Servidor rodando" será apresentada (com o comando: `console.log`)."

```
app.listen(porta, () => { console.log('Servidor rodando') })
```





# Autenticando usuários no Node.js

- Utilizaremos o comando abaixo para que o 'express' utilize a constante 'session' (do módulo 'express-session'), informando o segredo da sessão (palavra-chave) para utilizarmos na autenticação do usuário.
- É criada uma chave exclusiva do usuário. Aqui foi utilizado um valor aleatório ('1234567890') como exemplo.

```
app.use(session({ secret: '12134567890' })))
```

- Criaremos as variáveis de login e senha.

```
var login = 'admin'  
var senha = '1234'
```



# Autenticando usuários no Node.js

- Utilizaremos o código abaixo para renderizar o código e informar o local dos arquivos (*HMTL*).

```
app.engine('html', require('ejs').renderFile)
app.set('view engine', 'html')
app.set('views', path.join(__dirname, './'))
```

- Um dos módulos amplamente usados para fazer a renderização em Node.js é o módulo EJS (*Embedded JavaScript template*). EJS é uma linguagem de modelagem simples que permite gerar marcação HTML com JavaScript simples. Para saber mais, acesse o material em: <https://ejs.co/>.
- Utilizaremos o 'template engine', que oferece a possibilidade de criação de páginas das nossas aplicações em *Node.js* de forma dinâmica sem depender das limitações do *HTML*.

# Autenticando usuários no Node.js

- Nele primeiramente passamos qual 'template engine' vamos utilizar. Como primeiro parâmetro passamos o termo 'view engine', e como segundo, 'ejs'. Desta forma o Express entende que queremos usar o 'EJS'.
- Logo após apontar a 'template engine' que será utilizada, é necessário apontar também onde estarão os arquivos de interface. Para isso vamos adicionar outro 'app.set()', passando o termo 'views' como primeiro parâmetro e o diretório como segundo parâmetro, como no exemplo, './views'.
- Portanto, agora podemos criar uma rota 'get' e nela vamos passar qual 'view' será retornada quando houver essa requisição. Para maiores informações sobre o EJS, acesse: <https://expressjs.com/en/guide/using-template-engines.html>



# Autenticando usuários no Node.js

- Criaremos uma rota (raiz, '/') para a página principal do código, utilizando o 'app.get' com a *Arrow Functions*, passando o 'req' e 'res' como parâmetro.
- Criaremos um 'if' verificando se o usuário está logado ou não. Se estiver logado, a página 'logado.html' será aberta, caso contrário, será a est 'home.html'. Utilizaremos o comando 'req.session.login' para verificar essa opção.

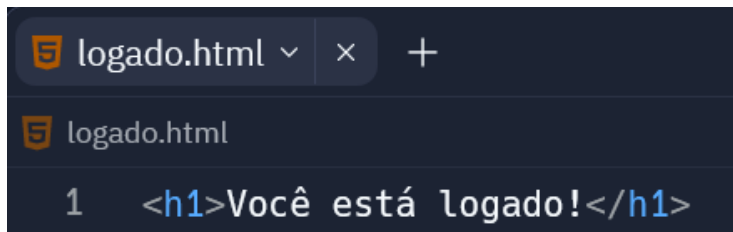
```
app.get('/', (req, res) => {  
  if (req.session.login) {  
    res.render('logado')  
  }  
  else{  
    res.render('home')  
  }  
})
```

Perceba que criamos a rota principal e utilizamos o comando 'res.render()', ou seja, ao acessar a rota /, será exibida a 'view' logado ou home.



# Autenticando usuários no Node.js

- Criaremos um arquivo *HTML*, 'logado.html', somente para exibir a mensagem: "Você está logado!". Essa página será aberta após a autenticação do usuário.



```
logado.html × +
logado.html
1 <h1>Você está logado!</h1>
```

- Criaremos um arquivo *HTML*, 'home.html', como página principal/raiz. Essa página será utilizada para autenticação do usuário.

# Autenticando usuários no Node.js

- Codificação de 'home.html':

```
<html lang="pt-br">
<head>
  <title>Página Inicial</title>
</head>
<body>
<h2>Área de Login</h2>
  <form method = "post">
    <input type = "text" name = "login">
    <input type = "password" name = "password">
    <input type = "submit" bane = "acao" value = "Logar">
  </form>
</body>
</html>
```

# Autenticando usuários no Node.js

- Utilizaremos o comando `req.session.login` para exibir (via `console.log`) o usuário autenticado (via requisição) no Console do *Replit*.

```
app.get('/', (req, res) => {  
  if (req.session.login) {  
    res.render('logado')  
    console.log('Usuário logado: ' + req.session.login)  
  }  
  else {  
    res.render('home')  
  }  
})
```

> Console x Shell x +

`express-session deprecated` undefined resave option  
; provide resave option `index.js:9:9`  
`express-session deprecated` undefined saveUninitial  
ized option; provide saveUninitialized option `inde  
x.js:9:9`  
Hint: hit control+c anytime to enter REPL.  
Servidor rodando  
**Usuário logado: admin**  
□

# Autenticando usuários no Node.js

- Para configurar a autenticação, utilizaremos o comando `'app.post'`. Criaremos uma rota (página inicial) para realizar essa ação com requisição (req) e resposta (res) na *Arrow Functions*.
- Os comandos `'req.body.password'` e `'req.body.login'` são utilizados para receber a requisição de senha (password) e login.

```
app.post('/', (req,res) => {  
  if (req.body.password == senha && req.body.login == login){  
    req.session.login = login  
    res.render('logado')  
  }  
  else{  
    res.render('home')  
  }  
})
```

# Autenticando usuários no Node.js

- Se login e senha forem iguais aos informados nas variáveis, a autenticação é validada e, como resposta, a página 'logado.html' é aberta/renderizada (com o comando `res.render('logado')`). Caso contrário, abre-se a página 'home.html'.
- Se for validado, é necessário criar uma sessão de login com o comando `req.session.login = login`.

```
app.post('/', (req,res) => {  
  if (req.body.password == senha && req.body.login == login){  
    req.session.login = login  
    res.render('logado')  
  }  
  else{  
    res.render('home')  
  }  
})
```



# Autenticando usuários no Node.js

- Para a verificação (login e senha) foi utilizado o módulo 'body-parser'. A constante 'app' utilizará o comando abaixo para usar o que foi informado no campo de login e senha.

```
app.use(bodyParser.urlencoded({ extended: true })))
```

- Crie o código acima em:

```
1  const express = require('express')
2  const session = require('express-session')
3  const bodyParser = require('body-parser')
4  const path = require('path')
5
6  const app = express()
7  const porta = 443
8
9  app.use(session({ secret: '12134567890' })))
10
11  app.use(bodyParser.urlencoded({ extended: true })))
12
13  var login = 'admin'
14  var senha = '1234'
```

Pronto!

Já temos uma *API* que autentica usuários no Node.js!

# Autenticando usuários no Node.js

Codificação final (1/2):

```
1  const express = require('express')
2  const session = require('express-session')
3  const bodyParser = require('body-parser')
4  const path = require('path')
5
6  const app = express()
7  const porta = 443
8
9  app.use(session({ secret: '12134567890' }))
10
11 app.use(bodyParser.urlencoded({ extended: true }))
12
13 var login = 'admin'
14 var senha = '1234'
15
16 app.engine('html', require('ejs').renderFile)
17 app.set('view engine', 'html')
18 app.set('views', path.join(__dirname, './'))
```

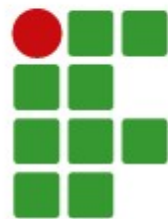
# Autenticando usuários no Node.js

Codificação final (2/2):

```
20 ✓ app.get('/', (req, res) => {
21   if (req.session.login) {
22     res.render('logado')
23     console.log('Usuário logado: ' + req.session.login)
24   }
25   else {
26     res.render('home')
27   }
28 })
29
30 ✓ app.post('/', (req, res) => {
31   if (req.body.password === senha && req.body.login === login) {
32     //logado com sucesso
33     req.session.login = login
34     res.render('logado')
35   }
36   else {
37     res.render('home')
38   }
39 })
40 app.listen(porta, () => { console.log('Servidor rodando') })
```

# Testes!

- Para testar o código, inicie com o botão 'Run' do *Replit*.
- Na aba 'Console' é possível observar a mensagem: "Servidor rodando"!
- Na aba 'Webview' é possível observar a mensagem: "Área de Login".
- Informe login (admin) e senha (1234).
- Observe que, se informar login e senha corretos, a mensagem "Você está logado!" será exibida.
- Pronto! Você conseguiu configurar o acesso à sua API Node.js.



**INSTITUTO  
FEDERAL**  
Espírito Santo