RECODE pro

node.js express

Trabalhando com o framework

O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.

Agora instale o Express no diretório app e salve-o na lista de dependências. Por exemplo:

```
$ npm install express --save
```

Para instalar o Express temporariamente não o inclua na lista de dependências, omita a opção --save:

```
$ npm install express
```

https://expressjs.com/pt-br/starter/installing.html

RECODE pro

node.js express

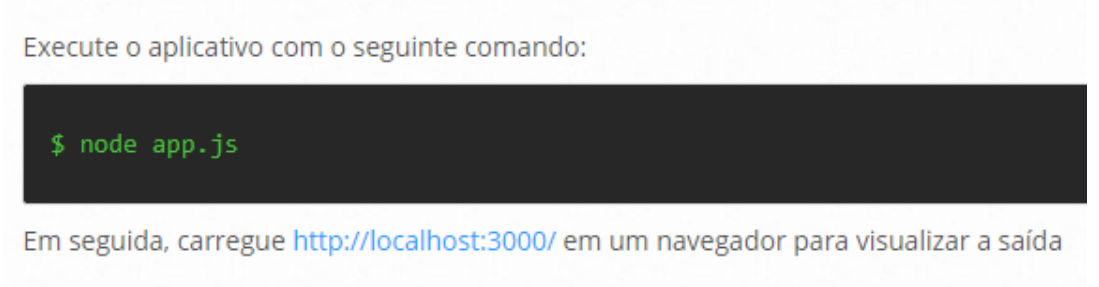
Exemplo Hello World usando express

Crie um novo projeto chamado app.js com seguinte código.

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```



O aplicativo inicia um servidor e escuta a porta 3000 por conexões. O aplicativo responde com “Hello World!” à solicitações para a URL raiz (/) ou ***rota***. Para todos os outros caminhos, ele irá responder com um **404 Não Encontrado**.

https://expressjs.com/pt-br/starter/hello-world.html

RECODE pro

node JS express

Roteamento Básico

O ***Roteamento*** refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) e um método de solicitação HTTP específico (GET, POST, e assim por diante).
Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.
A definição de rotas aceita a seguinte estrutura

```
app.METHOD(PATH, HANDLER)
```

Onde:

- app é uma instância do express.
- METHOD é um método de solicitação HTTP.
- PATH é um caminho no servidor.
- HANDLER é a função executada quando a rota é correspondida.

https://expressjs.com/pt-br/starter/basic-routing.html

RECODE pro

node JS express

Roteamento Básico

Responder com `Hello World!` na página inicial:

```
app.get('/', function (req, res) {
  res.send('Hello World!');
});
```

Responder a uma solicitação POST na rota raiz (/) com a página inicial do aplicativo:

```
app.post('/', function (req, res) {
  res.send('Got a POST request');
});
```

https://expressjs.com/pt-br/starter/basic-routing.html

node.js express

Roteamento Básico

Responder a uma solicitação PUT para a rota /user:

```
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
});
```

Responder a uma solicitação DELETE para a rota /user:

```
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
});
```

https://expressjs.com/pt-br/starter/basic-routing.html

RECODE pro

node JS express

Roteamento Básico renderizando aquivos HTML

```
const express = require('express');
const app = express();

app.get("/", function(req, res){
  res.sendFile(__dirname + "/html/index.html");
});

app.get("/sobre", function(req, res){
  res.sendFile(__dirname + "/html/sobre.html");
});

app.get('/blog', function(req,res){
  res.send("Bem vindo ao meu blog!");
});
app.listen(8081, function(){console.log("Servidor Rodando!");});
```

https://expressjs.com/pt-br/starter/basic-routing.html

RECODE pro

node.js express

Exercício – refaça o exercício anterior usando o express

Crie 3 arquivos HTML: artigos.html, contato.html e erro.html

Coloque qualquer conteúdo para cada pagina html;

Ao digitar no browser o path: ***/artigos*** deve renderizar **artigos.html**

A regra anterior também se aplica para o arquivo **contato.html**;

Ao digitar qualquer path diferente de ***/artigos*** e ***/contato*** deve renderizar **erro.html**;

A rota principal **"/"** deve renderizar **artigos.html**;

https://expressjs.com/pt-br/starter/basic-routing.html

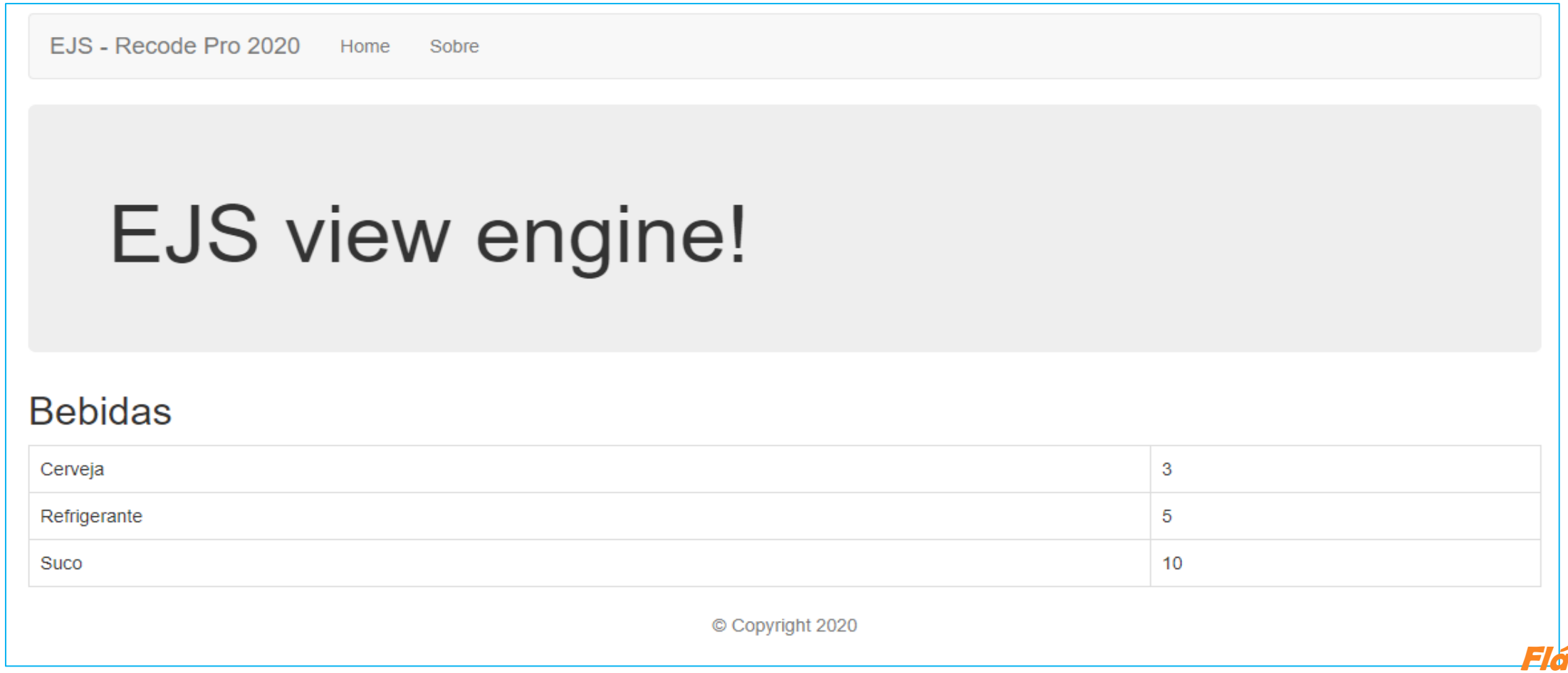
RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap



Flávio Mota

RECODE pro

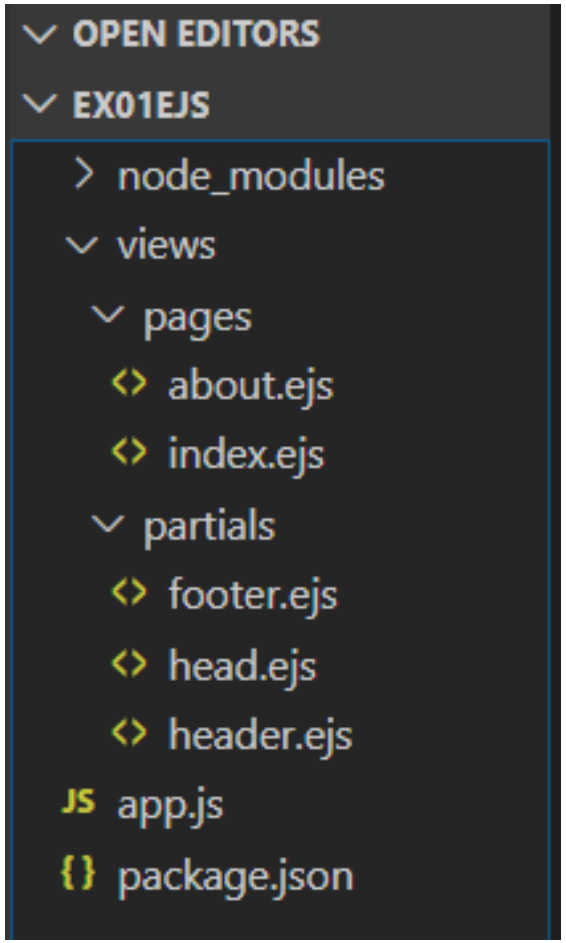
<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando módulos, EJS, Express e BootStrap

Estrutura de Arquivos



Para instalar as dependências digite:

npm install express --save

npm install ejs --save

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Com tudo instalado, vamos configurar a aplicação para utilizar o EJS e definir nossas rotas para as duas páginas que precisaremos, index (*full width*) e about (*sidebar*). Faremos tudo isso em nosso arquivo *app.js*.

Aqui definimos que a aplicação utilizará a porta 8080 e que nossa *view engine* será o *EJS*. É importante lembrar que o *res.render()* irá procurar nossas páginas dentro do diretório *views*.

```
//define uma instância para o express
var express = require('express');
var app = express();

//define o EJS como nossa view engine
app.set('view engine', 'ejs');
//Usar res.render para carregar arquivos de view ejs
//index page
app.get('/', function(req, res){

    var bebidas =
    [{ nome: 'Cerveja', total: 3 },
     { nome: 'Refrigerante', total: 5 },
     { nome: 'Suco', total: 10 }];

    res.render('pages/index', {
        bebidas: bebidas
    });
});
//about page
app.get('/about', function(req, res){
    res.render('pages/about');
});

app.listen(8080);
console.log('8080 é a porta mágica');
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Partials: head.ejs, header.ejs, e footer.ejs

Como na maioria das aplicações que construímos, muito código é reutilizado. Vamos definir três *partials* que utilizaremos em todo nosso site: *head.ejs, header.ejs,* e *footer.ejs*.

head.ejs

```
<!-- views/partials/header.ejs -->
<nav class="navbar navbar-default" role="navigation">
    <div class="container-fluid">

        <div class="navbar-header">
            <a class="navbar-brand" href="#">
                EJS - Recode Pro 2020
            </a>
        </div>

        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/about">Sobre</a></li>
        </ul>

    </div>
</nav>
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Partials: head.ejs, header.ejs, e footer.ejs

Como na maioria das aplicações que construímos, muito código é reutilizado. Vamos definir três *partials* que utilizaremos em todo nosso site: *head.ejs, header.ejs,* e *footer.ejs*.

header.ejs

```
<!-- views/partials/head.ejs -->
<meta charset="UTF-8">
<title>Template com EJS</title>
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<style>
    body { padding-top:50px; }
</style>
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Partials: head.ejs, header.ejs, e footer.ejs

Como na maioria das aplicações que construímos, muito código é reutilizado. Vamos definir três *partials* que utilizaremos em todo nosso site: *head.ejs, header.ejs,* e *footer.ejs*.

footer.ejs

```
<!-- views/partials/footer.ejs -->

<p class="text-center text-muted">© Copyright 2020</p>
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Usando partials EJS

Agora que temos nossas *partials* definidas, tudo o que precisamos é carregar os arquivos sempre que necessário. Para incluir uma *partial* basta digitar o seguinte código:

<% include NOME_ARQUIVO %>

index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <% include ../partials/head %>
</head>
<body class="container">
    <header>
        <% include ../partials/header %>
    </header>
    <main>
        <div class="jumbotron">
            <h1>EJS view engine!</h1>
        </div>
        <h2>Bebidas</h2>
        <table class="table table-bordered">
            <tbody>
            <% bebidas.forEach(function(bebida){ %>
                <tr>
                    <td><%= bebida.nome %></td>
                    <td><%= bebida.total %></td>
                </tr>
            <% }); %>
            </tbody>
        </table>
    </main>
    <footer>
        <% include ../partials/footer %>
    </footer>
</body>
</html>
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Usando partials EJS

Agora que temos nossas *partials* definidas, tudo o que precisamos é carregar os arquivos sempre que necessário. Para incluir uma *partial* basta digitar o seguinte código:

<% include NOME_ARQUIVO %>

about.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <% include ../partials/head %>
</head>
<body class="container">
    <header>
        <% include ../partials/header %>
    </header>
    <main>
        <div class="row">
            <div class="col-sm-8">
            </div>
            <div class="col-sm-4">
                <div class="well">
                    <h3>Recode Pro 2020 - Professores Flávio Mota / Flavio Sousa</h3>
                </div>
            </div>
        </div>
    </main>
    <footer>
        <% include ../partials/footer %>
    </footer>
</body>
</html>
```

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Trabalhando com o EJS view engine

Vamos criar o seguinte projeto usando EJS, Express e BootStrap

Passando dados para a view

app.ejs

```
app.get('/', function(req, res){

    var bebidas =
    [{ nome: 'Cerveja', total: 3 },
     { nome: 'Refrigerante', total: 5 },
     { nome: 'Suco', total: 10 }];

    res.render('pages/index', {
        bebidas: bebidas
    });
});
```

index.ejs

```
<h2>Bebidas</h2>
<table class="table table-bordered">
    <tbody>
    <% bebidas.forEach(function(bebida){ %>
        <tr>
            <td><%= bebida.nome %></td>
            <td><%= bebida.total %></td>
        </tr>
    <% }); %>
    </tbody>
</table>
```

Essa lista será utilizada na página index

RECODE pro

<%= EJS %>

Embedded JavaScript templating.

Exercício

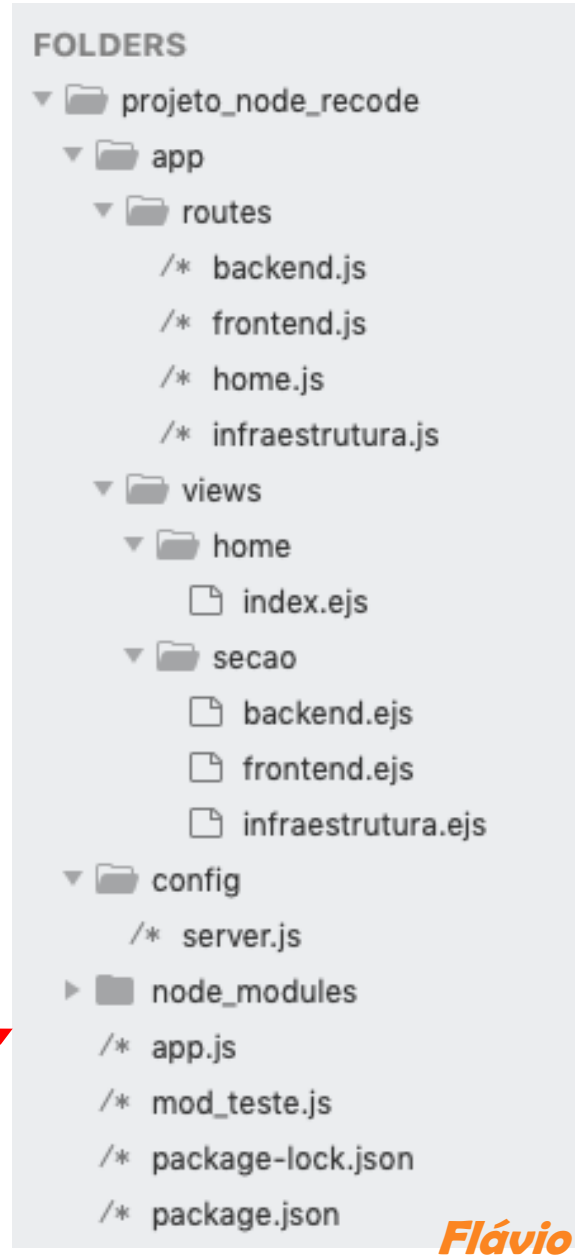
```
var http = require('http');

var server = http.createServer(function(req, res){
   var categoria = req.url;

   if(categoria == '/front-end'){
      res.end("<html><body>Tecnologias Front-End: TypeScript, Angular, React..</body></html>");
   }else if(categoria == '/back-end'){
      res.end("<html><body>Tecnologias Back-End: NodeJS, Python, PHP, MySQL...</body></html>");
   }else if(categoria == '/infraestrutura'){
      res.end("<html><body>Azure Cloud, Linux, MySQL Server...</body></html>");
   }else{
      res.end("<html><body>Programador Full Stack</body></html>");
   }
});

server.listen(3000);
```

Organize o projeto antigo como a estrutura ao lado, usando módulos, express e ejs



Flávio Mota