

# Iniciaremos em alguns instantes...

*“As aulas disponibilizadas por meios digitais envolvem horas de estudo, pesquisa e organização dos(as) professores(as) responsável(eis) e destinam-se, única e exclusivamente, aos alunos regularmente matriculados em cursos da Universidade Cruzeiro do Sul, para fins acadêmicos*

*A disponibilização/ divulgação dos links de acesso às aulas síncronas e/ ou registros das aulas a colegas não matriculados e a indivíduos estranhos à comunidade acadêmica da universidade implicará responsabilização civil e criminal conforme Artigo 46, inciso IV da Lei 9610/98, combinado com o artigo 184 do Código Penal, que estabelece direito autoral ao docente.*

*Assim, em atendimento à Lei e à valorização do trabalho docente, solicita-se que **não compartilhem as aulas com indivíduos que não estejam regularmente matriculados.**”*



A hand is pointing at a screen that displays blurred lines of JavaScript code. The code includes various strings and script tags, such as "type='text/javascript'", "</script>", and "script>". The background is dark, and the text is in various colors (green, blue, yellow).

# Técnicas de Programação



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Node.js parte III



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

## Rotas (Req)

No Req vai ficar tudo referente ao detalhes das requisições;

Vamos passar valores via GET (posteriormente aprenderemos por POST);

← → ↻ ⓘ localhost:7777/?nome=lozano&idade=37

Olá Mundo 2021!

Esses dados foram enviados e estão acessíveis dentro do req



Mas como acessá-los?

Vamos exibir na tela os dados nome e idade que foram enviado.

Edite o arquivo index.js inserindo o comando req.query + o parâmetro enviado.

```
4  router.get('/', (req, res) => {  
5      //res.send('Olá Mundo 2021!');  
6      res.send('Olá ' + req.query.nome);  
7  });
```






# Mostrando os dois valores

```
4 router.get('/',(req, res)=>{  
5   //res.send('Olá Mundo 2021!');  
6   res.send('Olá, '+req.query.nome+" ,voce tem " +req.query.idade+" anos de idade");  
7 });
```

← → ↻ ⓘ localhost:7777/?nome=lozano&idade=37

Olá, lozano ,voce tem 37 anos de idade





# Armazenando valores recebidos em variáveis

```
4 router.get('/',(req, res)=>{  
5     //res.send('Olá Mundo 2021!');  
6     let nome = req.query.nome;  
7     let idade = req.query.idade;  
8     res.send('Olá, '+nome+" ,voce tem " +idade+" anos de idade");  
9 });
```

← → ↻ ⓘ localhost:7777/?nome=lozano&idade=37

Olá, lozano ,voce tem 37 anos de idade



## Enviar parâmetros via POST

Para campos de senha, login, etc...;

Campos que não queremos que apareçam no navegador;

Enviamos internamente na requisição, através do método POST.

Vamos agora fazer uma pequena inclusão no nosso arquivo app.js:

```
JS app.js > ...
4
5 //configurações básicas do aplicativo
6 const app = express();
7 app.use('/',router); //foi passado 1 rota pois criamos apenas 1
8 💡
9 app.use(express.json());
10
11 module.exports = app; //exportamos o app, pois vamos importa-lo no servidor
12
13
```

Quando fazemos isso, dizemos para que as requisições POST sejam tratadas igual ao GET





```
15  router.get('/posts/12',(req, res)=>{    //numero aleatorio
16  |      res.send(' ');
17  });
```

Imagine um blog, ficaria inviável criarmos as rotas em números fixos, podemos randomizar parte da nossa rota, faremos isso posteriormente

Pegando parâmetros  
específicos na  
requisição

Vamos criar uma nova rota;

No arquivo index.js vamos inserir a  
seguinte rota:



# Pegando parâmetros específicos da requisição

```
15  router.get('/posts/:id',(req, res)=>{  
16      |      let id = req.params.id; params pega o id  
17      |      res.send('ID do post: '+id);  
18  |  });
```

← → ↻ ⓘ localhost:7777/posts/1

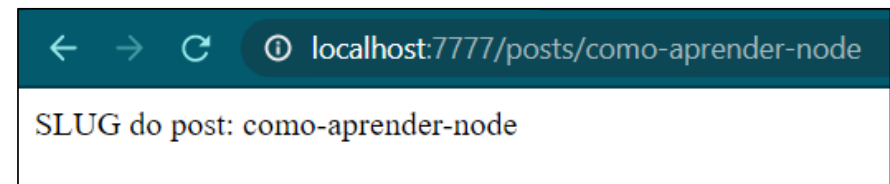
ID do post: 1

← → ↻ ⓘ localhost:7777/posts/50

ID do post: 50



```
router.get('/posts/:slug', (req, res) => {  
  let slug = req.params.slug;  
  res.send('SLUG do post: ' + slug);  
});
```



# Slug

Titulo do post como url.

Podemos acessar os parâmetros tanto pelo Slug quando pelo nome que quisermos, no nosso exemplo usamos o nome id



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

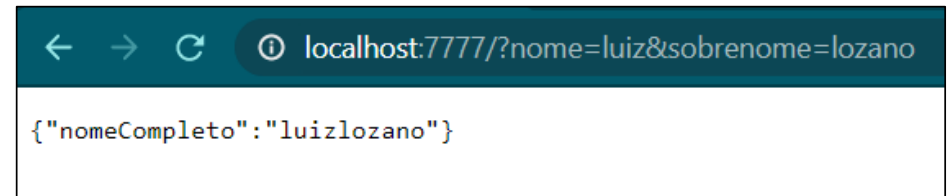
## Rotas (Res)

Podemos enviar o json ao invés de mandar um texto;

Json não pode ser usado junto com o send;

Vamos no arquivo index.js e vamos inserir o seguinte código:

```
3   const router = express.Router();
4   router.get('/', (req, res) => {
5       let nome = req.query.nome;
6       let sobrenome = req.query.sobrenome;
7
8       res.json({
9           nomeCompleto: nome + ' ' + sobrenome
10      }); //objeto {}
11
12  });
```



← → ↻ ⓘ localhost:7777/?nome=luiz&sobrenome=lozano

```
{\"nomeCompleto\": \"luizlozano\"}
```

Retornou um json, um json é 100% compatível com webservices, por exemplo



## Rota (Res) - Genérico

- O valor que mandar será transformado em json.

```
4 router.get('/', (req, res) => {  
5   |   res.json(req.query);  
6   });
```

← → ↻ ⓘ localhost:7777/?nome=luiz&sobrenome=lozano

```
{"nome": "luiz", "sobrenome": "lozano"}
```

← → ↻ ⓘ localhost:7777/?nome=luiz&sobrenome=lozano&idade=37

```
{"nome": "luiz", "sobrenome": "lozano", "idade": "37"}
```



# Observação

## Requisições

- Req (req.query)
- Post (req.body)
- Parâmetros da URL (req.params)

## Envios

- Send
- Json



# Configurando Template

```
utes > JS index.js > ...  
1  const express = require('express')  
2  
3  const router = express.Router();  
4  router.get('/', (req, res) => {  
5  
6  });  
7  
8  module.exports = router;
```

Até agora usamos Strings para nossas rotas, portanto, apresentamos apenas alguns pequenos textos como respostas às nossas solicitações. Porém é necessário trabalharmos com templates, pra que nosso projeto fique mais parecido com um sistema.

Iremos trabalhar com layouts, portanto, com a parte visual do sistema.

Vamos deixar nosso arquivo index.js limpo, conforme o código.



# Template Engine - Mustache

```
PS C:\projetoANF> npm install mustache-express --save  
  
added 5 packages, and audited 205 packages in 3s  
  
13 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities
```

São bibliotecas completas com foco em lidar com o layout dos sistemas.

Existem diversas bibliotecas, com suas particularidades de configuração de layout.

Iremos utilizar a biblioteca Mustache, pois é muito similar ao html.

Processo de configuração é o mesmo para qualquer biblioteca que você for utilizar como Engine.

Vamos instalar o Mustache para Express (pois estamos usando o express).





## Configurando Mustache

No arquivo app.js

mst é a extensão do arquivo visual que iremos utilizar.

```
JS app.js > ...
3  const mustache = require('mustache-express');
4  const router = require('./routes/index');
5
6  //configurações básicas do aplicativo
7  const app = express();
8  app.use('/', router);
9
10 app.use(express.json());
11
12 app.engine('mst', mustache()); //configuramos o motor, extensão que iremos usar
13 app.set('view engine', 'mst');//setar motor visual
14
15 module.exports = app;
```



## Configurando Mustache

Crie uma pasta no projeto chamada views.

Nessa pasta iremos colocar todos os arquivos visuais.

No arquivo app.js iremos setar essa pasta das views que criamos.

```
12  app.engine('mst', mustache()); //configuramos o motor, extensão que iremos usar
13  app.set('view engine', 'mst');//setar motor visual
14  //pega o diretorio absoluto do projeto e aumenta p/pasta views concantenando
15  app.set('views', __dirname + '/views');
16  module.exports = app;
```



# Agora iremos começar a utilizar o mustache

Iremos agora atualizar nossa rota principal no arquivo index.js

```
routes > JS index.js > ...  
1   const express = require('express');  
2  
3   const router = express.Router();  
4   router.get('/', (req, res) => {  
5     res.render('home'); // para renderizar  
6   });  
7  
8   module.exports = router;
```



## Utilizando o mustache

Na pasta views, vamos criar um arquivo chamado home.msf (para cada rota que iremos criar vamos ter que ter o arquivo visual).

Msf significa mustache.

≡ home.mst X

views > ≡ home.mst

```
1 <h1>Olá turma de ADS do Anália Franco</h1>
2 <p>Nossa aula de Técnicas de Programação</p>
```

localhost:7777

localhost:7777

# Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

## Utilizando o mustache

O html nos proporciona infinitas possibilidades.

Vamos agora trabalhar com o segundo parâmetro do render, até agora utilizamos apenas o primeiro. Nessa parâmetro podemos mandar diversos dados para o View.

```
4  router.get('/', (req, res) => {  
5      res.render('home', {  
6          'nome': 'Lozano',  
7          'idade': 37  
8      }); // para renderizar  
9  });
```



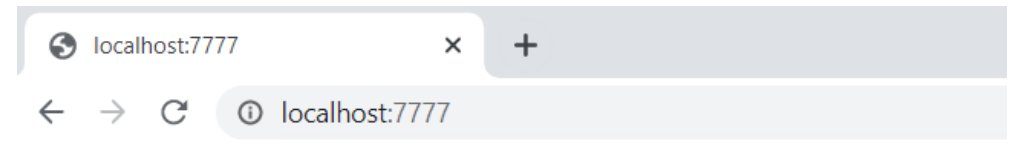
## Utilizando o mustache

Agora para receber os dados enviados pelo segundo parâmetro do render.

Dentro das `{{}}` conseguimos usar as variáveis.

views > home.mst

```
1 <h1>Olá turma de ADS do Anália Franco</h1>
2 <p>Nossa aula de Técnicas de Programação<p>
3 <p>Nome: {{nome}} tem {{idade}} anos de idade</p>
```



## Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Lozano tem 37 anos de idade



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

## Utilizando Mustache

Criando objeto e enviando o objeto no segundo parâmetro do render.

Ao testar, teremos o mesmo resultado.

```
4  router.get('/', (req, res) => {  
5      let obj = {  
6          'nome': 'Lozano',  
7          'idade': '37'  
8      };  
9      res.render('home', obj); // para renderizar  
10 }
```



## Configurando VSCode

≡ home.mst X

views > ≡ home.mst

```
1 <h1>Olá turma de ADS do Anália Franco</h1>
2 <p>Nossa aula de Técnicas de Programação<p>
3 <p>Nome: {{nome}} tem {{idade}} anos de idade</p>
```

Ln 3, Col 50 Spaces: 4 UTF-8 CRLF Plain Text 🔍 🔔

Podemos notar que o arquivo mst está em formato texto, vamos fazer com que o VSCode reconheça essa código como html.




Técnicas de Programação. - Prof. Msc. Luiz C M Lozano



# Configurando VSCode

Instale o plugin mustache.



**Mustache** dawwhite.mustache  
Dan White | 93.589 | ★★★★★ | Repository  
Syntax highlighting for mustache  
[Install](#) ⚙️

```
home.mst X
views > home.mst
1 <h1>Olá turma de ADS do Anália Franco</h1>
2 <p>Nossa aula de Técnicas de Programação<p>
3 <p>Nome: {{nome}} tem {{idade}} anos de idade</p>
```

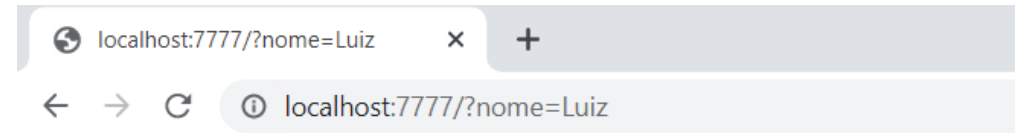


Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

Passando dados pelo navegador

Podemos comunicar o index.js com o arquivo da view, isso abre um grande leque de possibilidades.

```
routes > JS index.js > router.get('/') callback > [obj] > 'nome'
1  const express = require('express');
2
3  const router = express.Router();
4  router.get('/', (req, res) => {
5    let obj = {
6      'nome': req.query.nome,
7      'idade': '37'
8    };
9    res.render('home', obj); // para renderizar
10 };
11
12 module.exports = router;
```



## Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 37 anos de idade

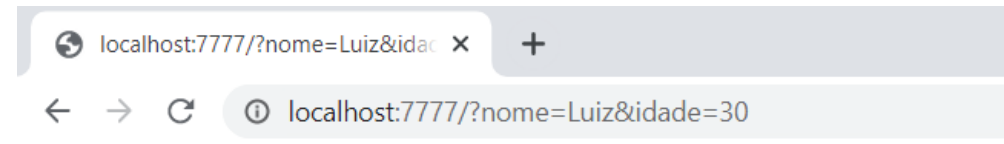


Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

Passando dados pelo navegador

Podemos comunicar o index.js com o arquivo da view, isso abre um grande leque de possibilidades.

```
routes > JS index.js > router.get('/') callback > [obj] > 'idade'
1  const express = require('express');
2
3  const router = express.Router();
4  router.get('/', (req, res) => {
5    let obj = {
6      'nome': req.query.nome,
7      'idade': req.query.idade
8    };
9    res.render('home', obj); // para renderizar
10 });
11
12 module.exports = router;
```



## Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 30 anos de idade



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Conhecendo o Mustache

Altere o arquivo da view mst, a função mostrar deve ter o mesmo nome da variável mostrar booleana dentro do index.js.

```
4   {{#mostrar}}
5   <p>Nome: {{nome}} tem {{idade}} anos de idade</p>
6   {{/mostrar}}
```

```
routes > JS index.js > router.get('/') callback > [obj]
1   const express = require('express');
2
3   const router = express.Router();
4   router.get('/', (req, res) => {
5     let obj = {
6       nome: req.query.nome,
7       idade: req.query.idade,
8       mostrar: false
9     };
10    res.render('home', obj); // para renderizar
11  });
12
13  module.exports = router;
14
```

localhost:7777/?nome=Luiz&idade=30

**Olá turma de ADS do Anália Franco**

Nossa aula de Técnicas de Programação



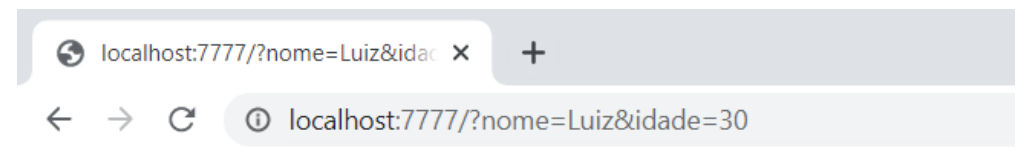
Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

## Conhecendo o Mustache

Alterando a variável mostrar para true, a função volta a exibir.

Tudo que estiver dentro da função mostrar no mst será apresentado ou não, de acordo com o booleano.

```
1  const express = require('express');
2
3  const router = express.Router();
4  router.get('/', (req, res) => {
5    let obj = {
6      nome: req.query.nome,
7      idade: req.query.idade,
8      mostrar: true
9    };
10   res.render('home', obj); // para renderizar
11 });
12
13 module.exports = router;
```



### Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 30 anos de idade



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

Lendo um  
Array/Lista

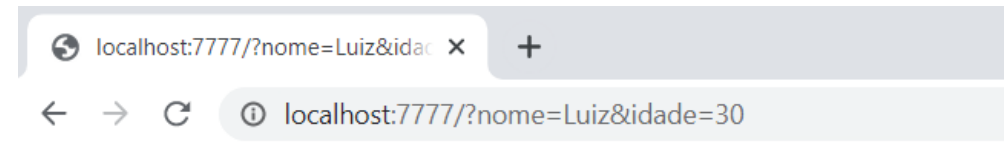
```
routes > JS index.js > router.get('/') callback > [obj] > disciplinas

5     let obj = {
6         nome: req.query.nome,
7         idade: req.query.idade,
8         mostrar: true,
9         disciplinas: [
10             { nome: 'TDP', qt: '20 aulas' },
11             { nome: 'POO', qt: '18 aulas' }
12         ]
13     };
14     res.render('home', obj); // para renderizar
15 });
16
17 module.exports = router;
18
```



# Lendo um Array/Lista

```
views > home.mst
1 <h1>Olá turma de ADS do Anália Franco</h1>
2 <p>Nossa aula de Técnicas de Programação</p>
3
4 {{#mostrar}}
5 <p>Nome: {{nome}} tem {{idade}} anos de idade</p>
6 {{/mostrar}}
7
8 <ul>
9 {{#disciplinas}}
10 <li>Nome: {{nome}} - Qtde: {{qt}}</li>
11 {{/disciplinas}}
12 </ul>
```



## Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 30 anos de idade

- Nome: TDP - Qtde: 20 aulas
- Nome: POO - Qtde: 18 aulas



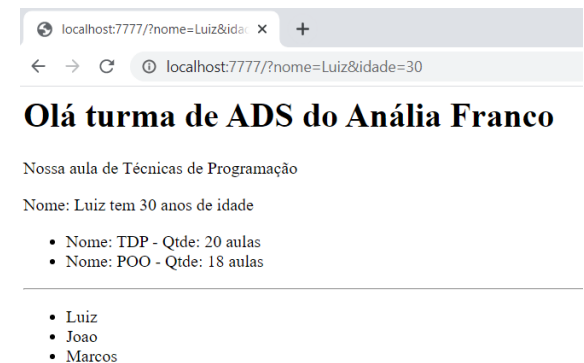
## Listando um Array

No caso não temos um objeto pra acessar, como nome, qt...

Com isso utilizamos um ponto `{{.}}`, com isso ele pega o conteúdo da vez.

```
routes > JS index.js > router.get('/') callback > [obj] > professores
5   let obj = {
6     nome: req.query.nome,
7     idade: req.query.idade,
8     mostrar: true,
9     disciplinas: [
10      {nome: 'TDP', qt: '20 aulas'},
11      {nome: 'POO', qt: '18 aulas'}
12    ],
13    professores: ['Luiz', 'Joao', 'Marcos']
14  };
15  res.render('home', obj); // para renderizar
16  });
17
18  module.exports = router;
19
```

```
14   <hr/>
15   <ul>
16     {{#professores}}
17     <li>{{.}}</li>
18     {{/professores}}
19   </ul>
```

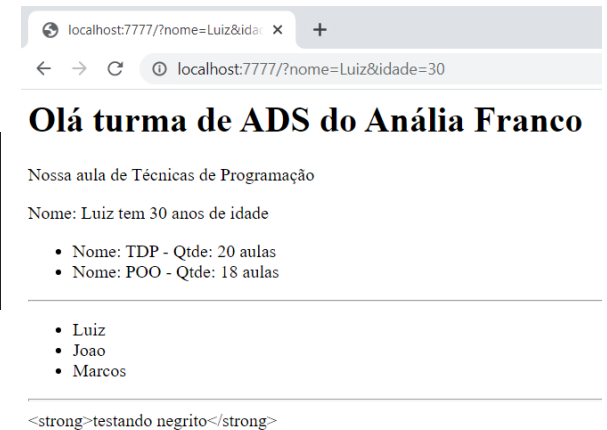




# Exibindo html de uma String

```
5 let obj = {
6   nome: req.query.nome,
7   idade: req.query.idade,
8   mostrar: true,
9   disciplinas: [
10    {nome: 'TDP', qt: '20 aulas'},
11    {nome: 'POO', qt: '18 aulas'}
12  ],
13  professores: ['Luiz', 'Joao', 'Marcos'],
14  teste: '<strong>testando negrito</strong>'
15 };
16 res.render('home', obj); // para renderizar
17 });
18
19 module.exports = router;
```

```
21 <hr/>
22 {{teste}}
```

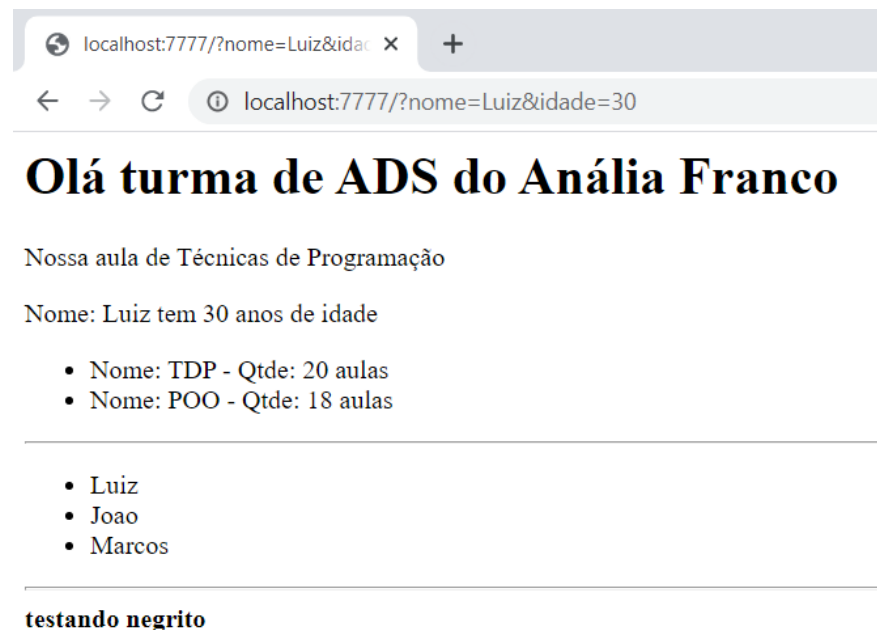


## Usando html de uma String

Se ao atualizar a página, não funcionar, é porque você apenas atualizou o mst, portanto vá no index.js, coloque algum espaço e salve, depois atualize que irá funcionar (provável bug do node).

Coloque mais uma chave para ele interpretar o html corretamente.

22 {{{teste}}}



localhost:7777/?nome=Luiz&idade=30

### Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 30 anos de idade

- Nome: TDP - Qtde: 20 aulas
- Nome: POO - Qtde: 18 aulas

---

- Luiz
- Joao
- Marcos

---

testando negrito

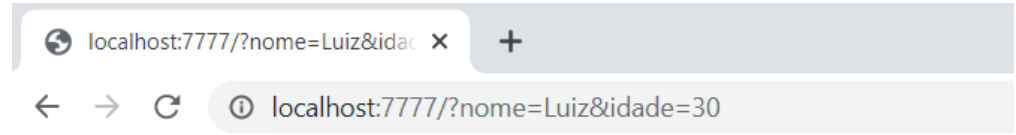


Suponho que a lista esteja vazia

Se a lista de professores estiver vazia, esse comando faz ele monitorar e enviar uma mensagem quando ela ficar vazia (teste com valores e depois vazia).

O chapéu significa que são situações que não teve o loop.

```
16 {{#professores}}
17 <li>{{.}}</li>
18 {{/professores}}
19 </ul>
20
21 {{^professores}}
22 <p>Não existem professores cadastrados</p>
23 {{/professores}}
24
```



## Olá turma de ADS do Anália Franco

Nossa aula de Técnicas de Programação

Nome: Luiz tem 30 anos de idade

- Nome: TDP - Qtde: 20 aulas
- Nome: POO - Qtde: 18 aulas

Não existem professores cadastrados

**testando negrito**



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Comentários no mst

---

`{{ ! }}`

29 `{{! esse trecho não será executado}}`



## Vamos limpar nosso arquivo mst

Deixar apenas um h1 com a mensagem página home.

Vamos criar como se fossem cabeçalhos, para serem usados em todas as páginas, pois quando for alterado gera uma alteração em cascata.

No node chamamos isso de partials.

```
🍷 home.mst JS index.js  
views > 🍷 home.mst  
1 <h1>Página Home </h1>  
2  
3
```

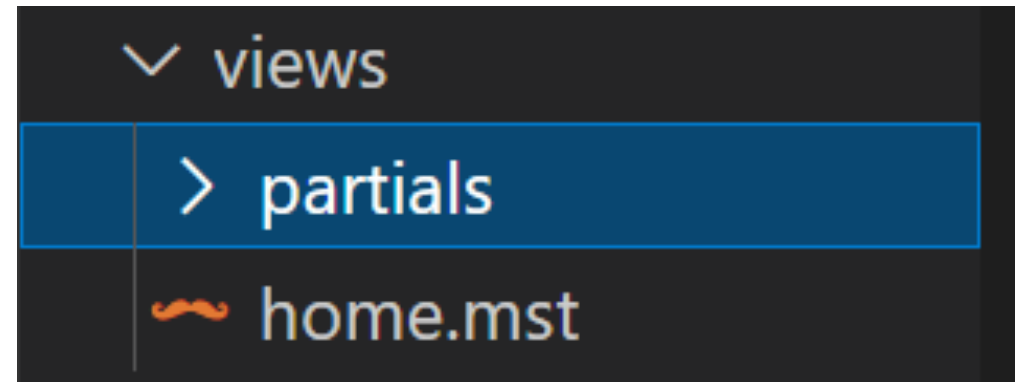


## Edite o arquivo app.js

Envie parâmetros para o mustache.

No primeiro parâmetro iremos dizer onde esta a pasta dos  
partials.

```
JS app.js > ...
3  const mustache = require('mustache-express');
4  const router = require('./routes/index');
5
6  //configurações básicas do aplicativo
7  const app = express();
8  app.use('/',router);
9
10 app.use(express.json());
11
12 app.engine('mst', mustache(__dirname + '/views/partials'));
13 app.set('view engine', 'mst');//setar motor visual
14 //pega o diretorio absoluto do projeto e aumenta p/pasta v
15 app.set('views', __dirname + '/views');
16 module.exports = app;
```



header.mst

Dentro da pasta partials, crie o arquivo headers.mst

Esse será nosso arquivo de partials para cabeçalhos.

```
🍷 header.mst ●  
  
views > partials > 🍷 header.mst  
1    <h1>Cabeçalho</h1>  
2    <hr/>
```



## Edite o arquivo app.js

O segundo parâmetro do mustache quer dizer a extensão desses partials.

A partir de agora, Podemos usar o header em qualquer lugar das páginas.

```
JS app.js > ...
3  const mustache = require('mustache-express');
4  const router = require('./routes/index');
5
6  const app = express();
7  app.use('/', router);
8
9  app.use(express.json());
10
11 app.engine('mst', mustache(__dirname + '/views/partials', '.mst'));
12 app.set('view engine', 'mst');
13
14 app.set('views', __dirname + '/views');
15 module.exports = app;
```



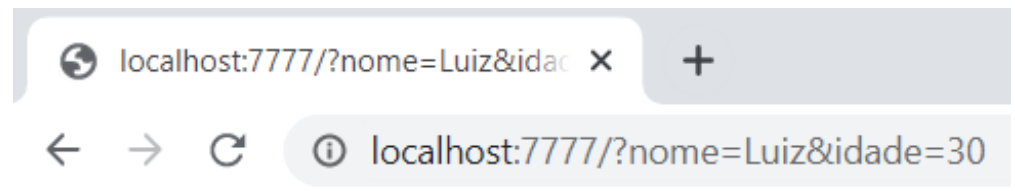


## Usando o header no home

Deixe o arquivo index.js da seguinte forma.

Note que ao executar, irá aparecer o home que está na página home.

```
routes > JS index.js > router.get('/') callback
1  const express = require('express');
2
3  const router = express.Router();
4  router.get('/', (req, res) => {
5    let obj = {
6      nome: req.query.nome
7    };
8    res.render('home', obj);
9  });
10
11 module.exports = router;
```



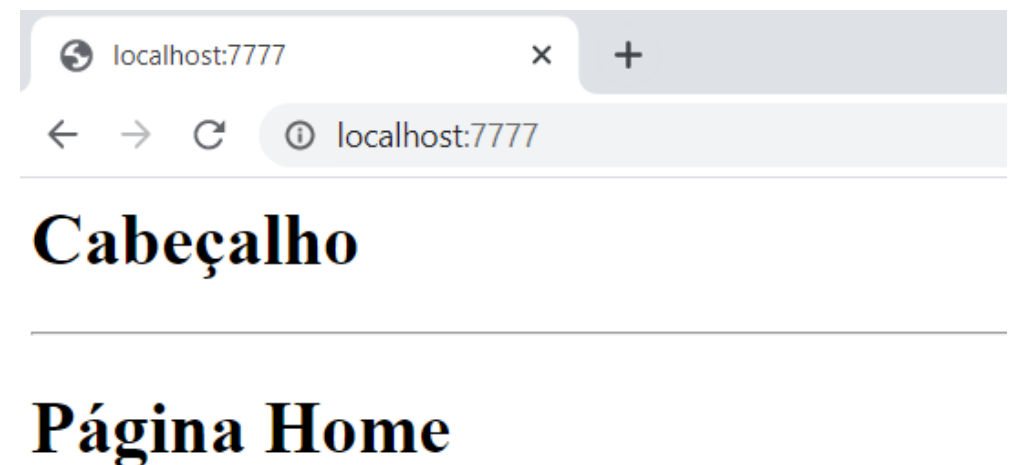
# Página Home



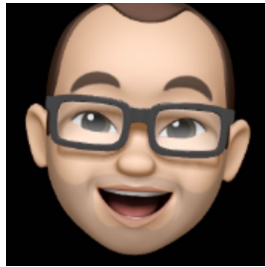
# Usando o header no home

Abre e fecha chaves duplas, símbolo de maior, espaço e o nome do partial que irá usar.

```
home.mst X
views > home.mst
1  {{> header}}
2
3  <h1>Página Home </h1>
4
```



Siga-me...



<https://www.facebook.com/professor.lozano>



<https://www.instagram.com/professor.lozano>



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Créditos:

Material parcialmente elaborado  
pelo Prof. Alexander Gobatto e  
adaptado pelo Prof. Luiz Lozano



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano