



16/8/2019

Treinamento Angular 8

COTI INFORMATICA



WWW.COTIINFORMATICA.COM.BR

AV RIO BRANCO, 185 SALA 307 – CENTRO – RIO DE JANEIRO - RJ

Sumário

O que é Jasmine?	2
A função <code>describe()</code>	2
A função <code>it()</code>	2
A função <code>expect()</code>	3
Exemplo prático: uma Pilha	3
O que é Karma?	5
Jsonwebtoken	6
Tecnicamente o que é ?	7
Header	7
Payload	8
Signature	9
Usando o token	9
JSON Web Tokens.....	10
NgPrime.....	12
PROJETO SERVER NODE MYSQL:.....	14
PROJETO CLIENT DETALHES	19
PROJETO SERVIDOR:.....	34
Package,json.....	36
PROJETO ANGULAR:	38

O que é Jasmine?



O Jasmine é uma biblioteca de testes JavaScript que tem suporte ao BDD (Behaviour Driven Development), ele é utilizado junto com o (TDD) Test Driven Development. Todo desenvolvedor sabe que o teste é uma das partes fundamentais no desenvolvimento de software, mas infelizmente nem sempre é possível implementar ele no seu dia dia.

A função `describe()`

Uma suite de teste no Jasmine começa com uma chamada à função `describe()`. Essa função tem 2 parâmetros: o primeiro é o nome de sua suite de teste (esse nome vai aparecer no resultado da execução dos testes); o segundo parâmetro é uma função com código que implementa a suite de teste em si. Veja o código abaixo retirado do arquivo de testes exemplos do Jasmine, o `PlayerSpec.js`.

```
describe("Player", function() {  
    // aqui são implementados os testes em si  
});
```

A função `it()`

Os specs (termo utilizado pelo Jasmine para se referir aos testes a serem executados) são criados por meio da função `it()`. Assim como a `describe()`, esta função recebe 2 argumentos. O primeiro é o nome do teste a ser executado. O segundo argumento é a função de teste, onde fica o próprio código de teste em si: a preparação, a chamada do teste e a verificação do resultado. Veja o exemplo abaixo.

```
describe("Player", function() {

    it("should be able to play a Song", function() {
        // aqui fica o código de teste
    });

});
```

A função expect()

Por fim é dentro da função que é o segundo argumento de `it()` colocamos o código de teste. Chamamos a função a ser testada e verificamos o resultado dela por meio de expectations. Para isso utilizamos a função `expect()` encadeada com algum matcher. Veja o exemplo abaixo.

```
describe("Player", function() {

    it("should be able to play a Song", function() {
        var player = new Player();
        var song = new Song();
        player.play(song);
        expect(player.currentlyPlayingSong).toEqual(song);
    });

});
```

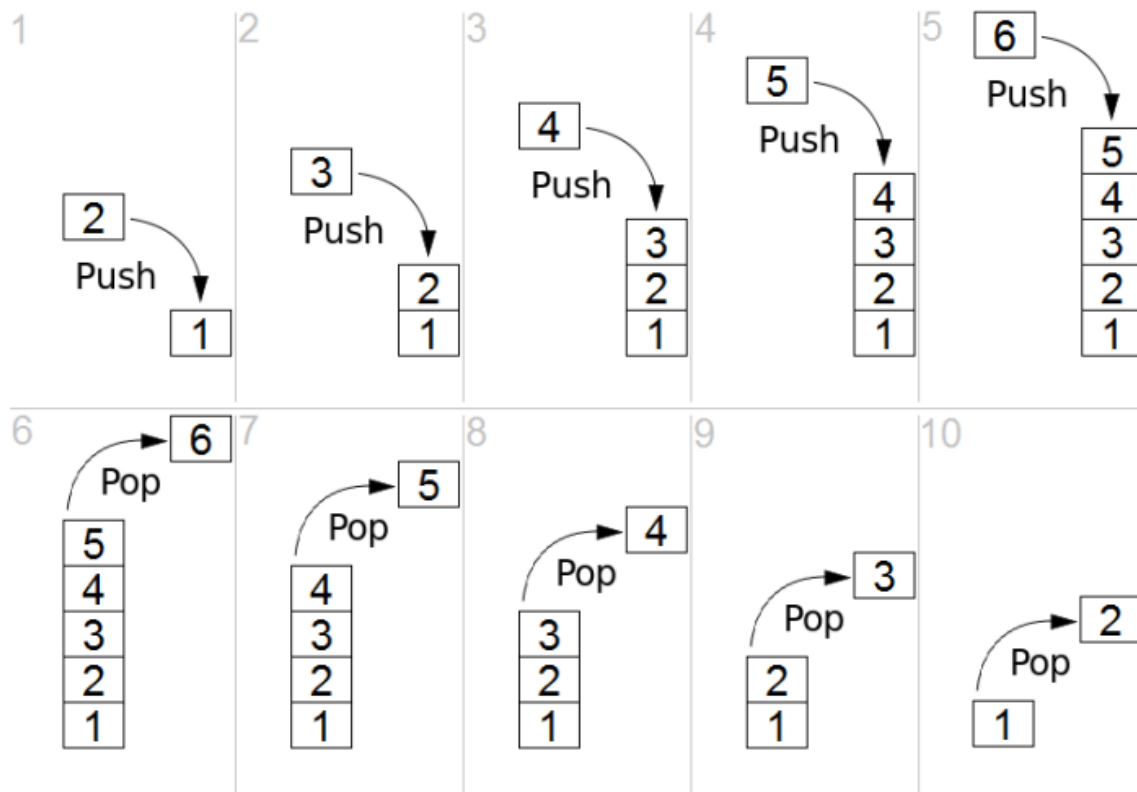
A linha em destaque é onde é feita a verificação. A função `expect()` recebe o valor atual de execução; a função `toEqual()` é um matcher (o Jasmine possui vários matchers pré-definidos) que recebe como argumento o valor esperado.

Agora que vimos as principais funções utilizadas para a escrita de testes no Jasmine, vamos fazer um exemplo utilizando esses conceitos em conjunto.

Exemplo prático: uma Pilha

Vamos escrever código de teste para a estrutura de dados conhecida como Pilha. Para os que não conhecem, a pilha é uma estrutura de dados que implementa o algoritmo Last In, First Out (LIFO). É como uma pilha de pratos organizada para ser lavada; os primeiros pratos a serem tirados para lavar são os que estão no topo da pilha.

A figura abaixo demonstra o funcionamento de uma pilha que implementa o LIFO. Veja que os elementos vão sendo inseridos sempre no topo da pilha (parte de cima da figura). E são removidos sempre do topo (parte de baixo da figura).



- **push:** adiciona um novo elemento à pilha;
- **pop:** retira e retorna o elemento do topo da pilha (o último elemento que foi adicionado);
- **peek:** permite verificar o elemento que está no topo da pilha, mas sem tirá-lo de lá;
- **size:** função auxiliar que permite verificar a quantidade de elementos presentes na pilha;
- **isEmpty:** outra função auxiliar que retorna se a pilha está vazia.

O que é Karma?



O Karma, construído em NodeJS, com o objetivo de facilitar a execução de seus testes. Ele pode ser instalado facilmente pelo NPM (Node Package Manager) e conta com suporte para os frameworks de teste Jasmine, QUnit e Mocha, porém é bem simples de extendê-lo para que ele rode também outros tipos de testes.

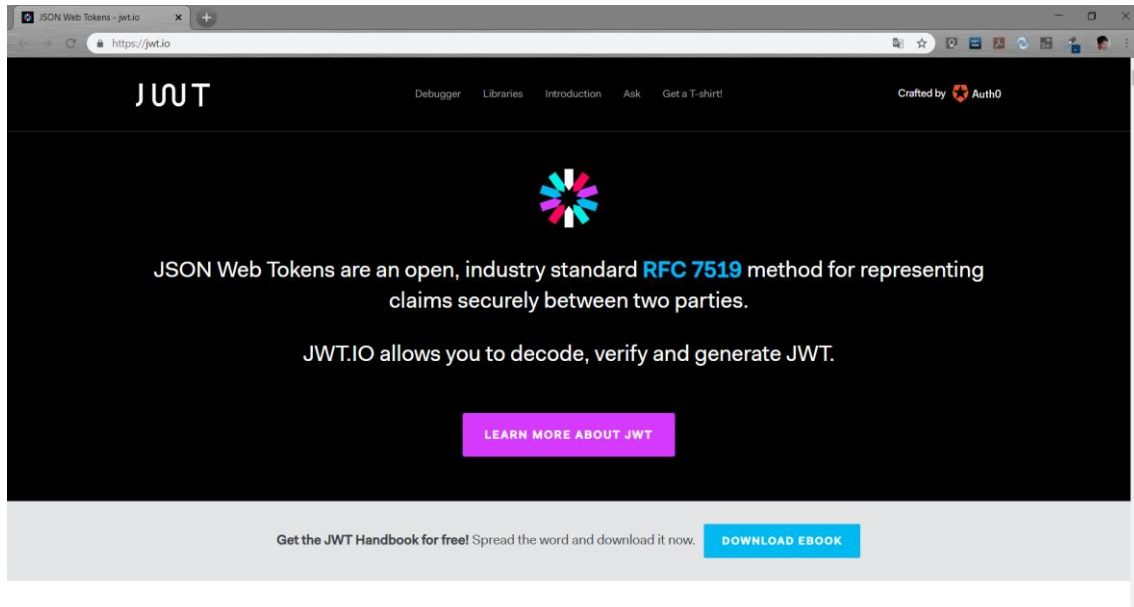
Com o Karma é possível marcar alguns arquivos para serem observados, subir um pequeno servidor local e então, em qualquer mudança nesses arquivos, o Karma irá rodar a sua suíte de testes automaticamente e te apresentar os resultados. Também é bem simples configurá-lo em ferramentas de integração contínua como o Jenkins ou Travis.

Além de ter uma configuração muito simples e interativa, rodar os seus testes em vários browsers, também há vários plugins para integrar a execução em outras ferramentas, geralmente utilizadas para build, como Grunt e Maven (no caso de projetos que também utilizam Java).

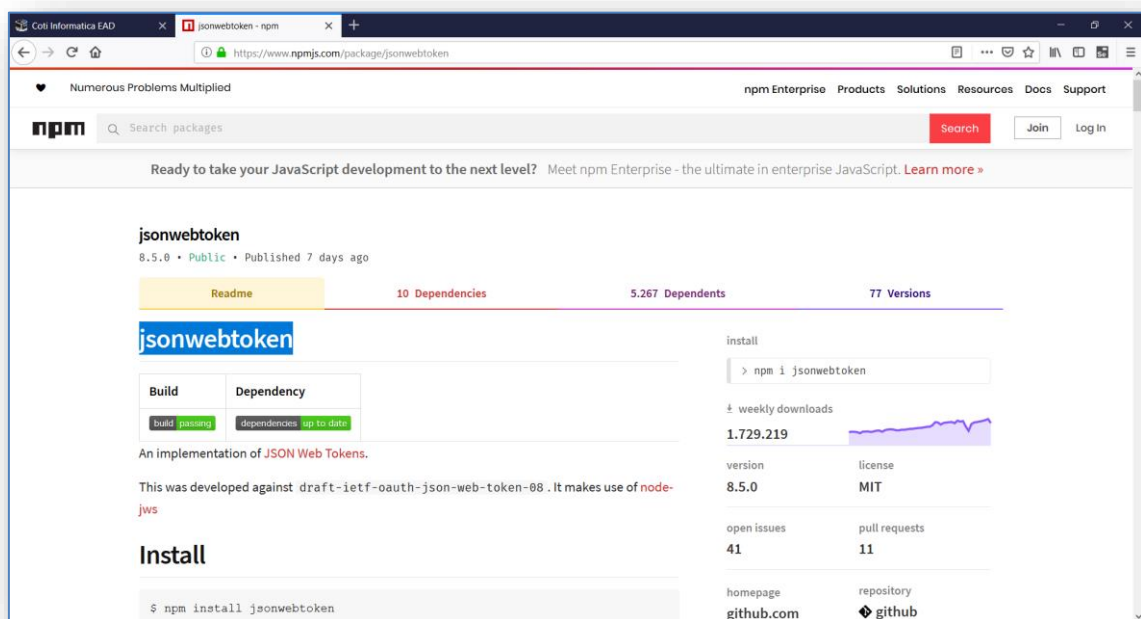
Muitas vezes ocorre uma confusão sobre as mudanças nos testes causadas ao utilizar Karma, porém, é válido ressaltar que ele não irá simplificar a maneira como seus testes são escritos, ou ajudar você a construir ótimas suítes só pelo simples fato de usá-lo. O que o Karma faz é apenas rodar os testes de maneira rápida e simples, e aposto que isso pode ajudar muito a sua vida durante o desenvolvimento de aplicações com JavaScript.

Jsonwebtoken

<https://jwt.io/>



<https://www.npmjs.com/package/jsonwebtoken>



Tecnicamente o que é ?

Após o registro quando o usuário faz o pedido de login

1. O servidor verifica se o usuário é legítimo e responde com um token(JWT) contendo a identidade do usuário.

2. O token em resposta é armazenado localmente no sistema do cliente e o usuário é permitido dentro do aplicativo.

3. Quando o usuário faz alterações em seu perfil, seu perfil [dados + token] é enviado para o servidor.

4. O servidor primeiro verifica se o pedido contém o token (responde com um erro se não for passado). O token é então verificado, uma vez feito, os dados do perfil da carga são verificados e as respectivas alterações são feitas no banco de dados.

5. É o mesmo para todas as outras ações feitas pelo usuário.

6. Quando o usuário “desconecta” o token de identificação é destruído do local.

O JWT é um padrão ([RFC-7519](#)) de mercado que define como transmitir e armazenar objetos JSON de forma compacta e segura entre diferentes aplicações. Os dados nele contidos podem ser validados a qualquer momento pois o token é assinado digitalmente.

Ele é formado por três seções: Header, Payload e Signature.

Header

O Header é um objeto JSON que define informações sobre o tipo do token (typ), nesse caso JWT, e o algoritmo de criptografia usado em sua assinatura (alg), normalmente [HMAC SHA256](#) ou [RSA](#).

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```


Payload

O Payload é um objeto JSON com as Claims (informações) da entidade tratada, normalmente o usuário autenticado.

Essas claims podem ser de 3 tipos:

- Reserved claims: atributos não obrigatórios (mas recomendados) que são usados na validação do token pelos protocolos de segurança das APIs.

sub (subject) = Entidade à quem o token pertence, normalmente o ID do usuário;

iss (issuer) = Emissor do token;

exp (expiration) = Timestamp de quando o token irá expirar;

iat (issued at) = Timestamp de quando o token foi criado;

aud (audience) = Destinatário do token, representa a aplicação que irá usá-lo.

Geralmente os atributos mais utilizados são: sub, iss e exp.

- Public claims: atributos que usamos em nossas aplicações. Normalmente armazenamos as informações do usuário autenticado na aplicação.

name

roles

permissions


- Private claims: atributos definidos especialmente para compartilhar informações entre aplicações.

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Por segurança recomenda-se não armazenar informações confidenciais ou sensíveis no token.

Signature

A assinatura é a concatenação dos hashes gerados a partir do Header e Payload usando base64UrlEncode, com uma chave secreta ou certificado RSA.

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)  secret base64 encoded
```

Essa assinatura é utilizada para garantir a integridade do token, no caso, se ele foi modificado e se realmente foi gerado por você.

Isso previne ataques do tipo man-in-the-middle, onde o invasor poderia interceptar a requisição e modificar seu conteúdo, desta forma personificando o usuário com informações falsas. Caso o payload seja alterado, o hash final não será válido pois não foi assinado com sua chave secreta. Apenas quem está de posse da chave pode criar, alterar e validar o token.

Usando o token

Ao fazer login em um serviço de autenticação um token JWT é criado e retornado para o client. Esse token deve ser enviado para as APIs através do header Authorization de cada requisição HTTP com a flag Bearer, conforme ilustra o diagrama abaixo.

Authorization: Bearer <token>

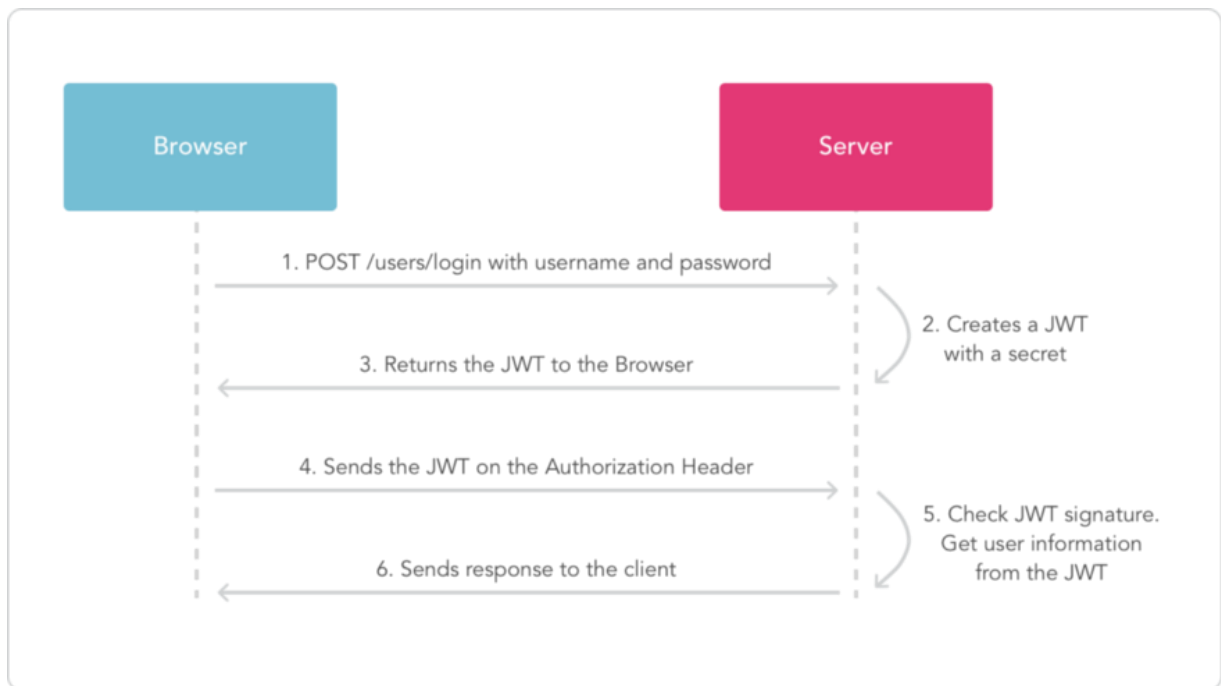


Diagrama de sequência usando token JWT

Em posse do token, a API não precisa ir até o banco de dados consultar as informações do usuário, pois contido no próprio token JWT já temos suas credenciais de acesso.

JSON Web Tokens

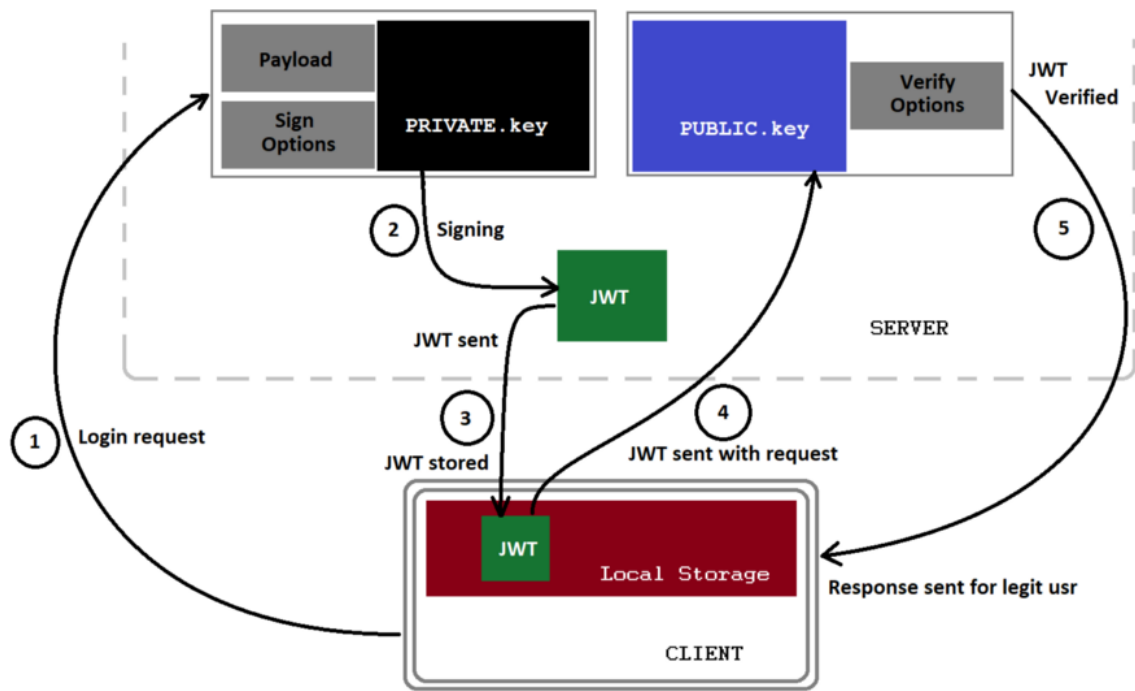
JWT, resumidamente, é uma string de caracteres codificados que, caso cliente e servidor estejam sob HTTPS, permite que somente o servidor que conhece o 'segredo' possa ler o conteúdo do token, e assim confirmar a autenticidade do cliente.

Ou seja, quando um usuário se autentica no sistema (com usuário e senha), o servidor gera um token com data de expiração pra ele. Durante as requisições seguintes do cliente, o JWT é enviado no cabeçalho da requisição e, caso esteja válido, a API irá permitir acesso aos recursos solicitados, sem a necessidade de se autenticar novamente.



O conteúdo do JWT é um payload JSON que pode conter a informação que você desejar, que lhe permita mais tarde conceder autorização a determinados recursos para determinados usuários. Minimamente ele terá o ID do usuário autenticado, mas pode conter muito mais do que isso. Saiba a diferença, autenticação é você provar que você é você mesmo. Já autorização, é você provar que possui permissão para fazer ou ver o que você está tentando.

Antes de emitir o JWT, é necessário que o usuário passe por uma autenticação tradicional, geralmente com usuário e senha. Essa informação fornecida é validada junto a uma base de dados e somente caso ela esteja ok é que geramos o JWT para ele.



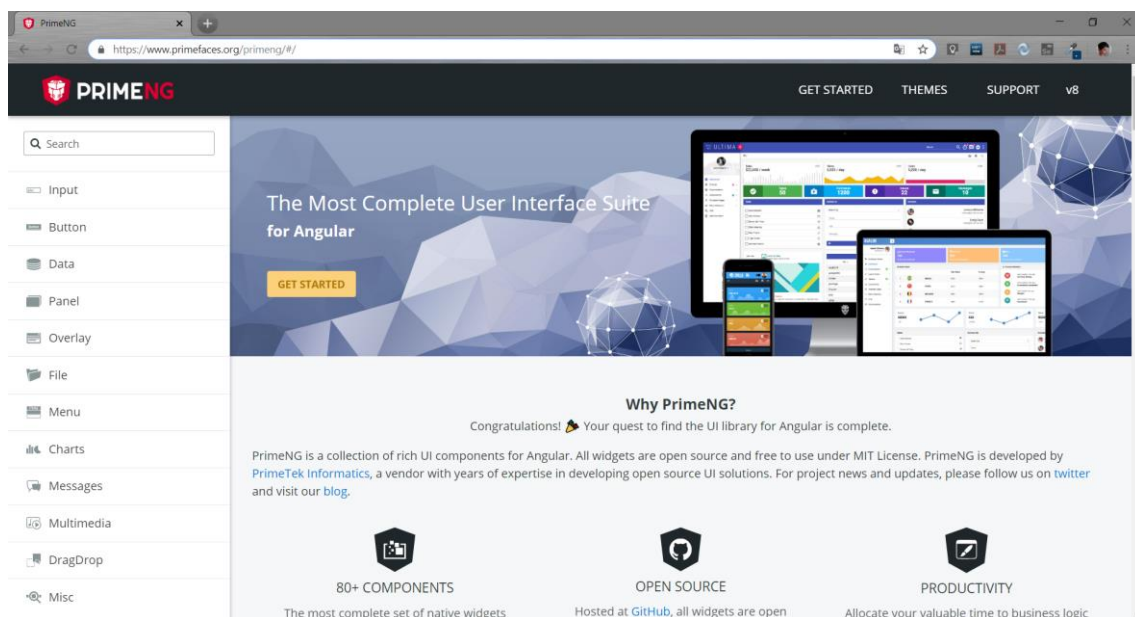
NgPrime



<https://www.primefaces.org/primeng/#/>

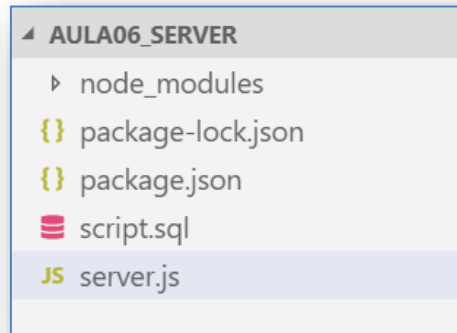
PrimeNG é uma coleção de componentes de UI ricos para Angular. Todos os widgets são open source e gratuitos para uso sob licença MIT. O PrimeNG é desenvolvido pela PrimeTek Informatics, um fornecedor com anos de experiência no desenvolvimento de soluções de UI de código aberto.

O PrimeNG possui cerca de 80 componentes de código aberto e gratuito com o uso sob a licença MIT. Foi desenvolvido pela PrimeTek Informatics e todos seus widgets estão hospedados no GitHub e a grande maioria são nativos. É uma biblioteca rica em componentes de entendimento relativamente simples com códigos limpos e bem organizados. Apesar disso possui alguns defeitos com relação a consistência de seus componentes e nem sempre fornece todas as variáveis necessárias para uma determinada atividade. Em contraste está sempre gerando novas versões e há correções dos erros que encontram.



PROJETO SERVER NODE MYSQL:

Estrutura do projeto depois de finalizado:



server.js

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();

const mysql = require('mysql');
app.use(bodyParser.json());

app.use(function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*');
  res.header("Access-Control-Allow-Credentials", "true");
  res.header('Access-Control-Allow-Methods',
    'GET,PUT,POST,DELETE,OPTIONS');
  res.header('Access-Control-Allow-Headers', 'X-Requested-
    With, Content-Type, X-Codingpedia, Authorization');
  next();
});

function execSqlQuery(sqlQry, res) {
  const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    port: '3306',
    password: 'coti',
    database: 'dbCincob'
  })
```

```

        connection.query(sqlQry, function (error, results, fields) {
            if (error) {
                res.json(error);
            } else {
                res.json(results);
            }
            connection.end();
            console.log('Executou ...');
        })
    }

    app.post('/aluno', (req, res) => {
        var nome = req.body.nome;
        var email = req.body.email;
        var disciplina = req.body.disciplina;
        var nota1 = req.body.nota1;
        var nota2 = req.body.nota2;
        var situacao = req.body.situacao;
        execSqlQuery(`insert into aluno (nome, email, disciplina,
        nota1, nota2, situacao) values ('${nome}' , '${email}' ,
        '${disciplina}' , '${nota1}' , '${nota2}' , '${situacao}')`,
        res);
    });

    app.get('/aluno', (req, res) => {
        execSqlQuery(`select * from aluno `, res);
    });

    app.get('/aluno/:id', (req, res) => {
        var id = parseInt(req.params.id);
        execSqlQuery(`select * from aluno where idAluno=` + id,
        res);
    });

    app.delete('/aluno', (req, res) => {
        var id = parseInt(req.body.id);
        execSqlQuery(`delete from aluno where idAluno=` + id, res);
    });

    var server = app.listen(3006, 'localhost', function () {
        var host = server.address().address;
        var port = server.address().port;
    });

```



```
    console.log('Listening at http://%s:%s', host, port);  
  })
```

script.sql

```
# npm install -S nodemon  
# npm install -f nodemon  
# npm install -S mysql  
  
#drop database dbCincob;  
  
create database dbCincob;  
use dbCincob;  
  
create table aluno(idAluno int primary key auto_increment,  
  nome varchar (50),  
  email varchar (50) unique,  
  disciplina varchar (50),  
  nota1 float,  
  nota2 float,  
  situacao varchar (20));  
  
  insert into aluno values  
  (null,'luís','luís@gmail.com','java',7,8,'aprovado');  
  
insert into aluno values  
  (null,'xuma','xuma.com','java',6,6,'reprovado');  
  
insert into aluno values  
  (null,'lu','lu@gmail.com','java',9,8,'aprovado');  
  
select * from aluno;
```

No banco

```
MySQL 5.7 Command Line Client - Unicode
mysql>
mysql> select * from aluno;
+-----+-----+-----+-----+-----+-----+-----+
| idAluno | nome | email          | disciplina | nota1 | nota2 | situacao |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | luis | luis@gmail.com | java       | 7     | 8     | aprovado |
| 2 | xuma | xuma.com       | java       | 6     | 6     | reprovado |
| 3 | lu  | lu@gmail.com   | java       | 9     | 8     | aprovado |
+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
mysql> _
```

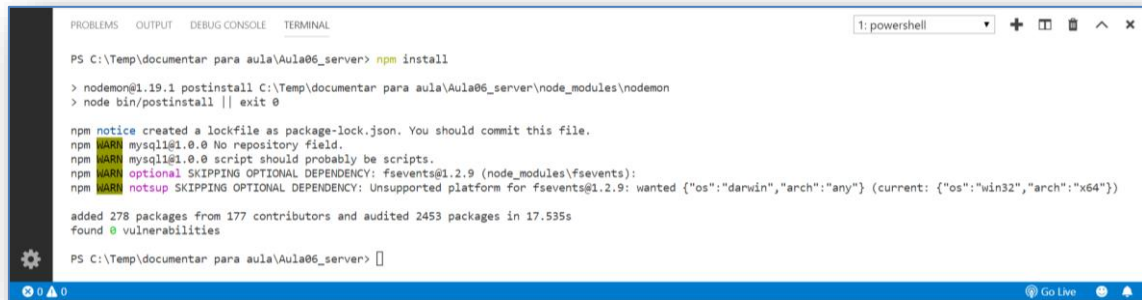
package.json

```
{
  "name": "mysql1",
  "version": "1.0.0",
  "description": "Projeto Mysql",
  "main": "server.js",
  "script": {
    "test": "echo ' Error: Banco nao especificado ' && exit
1"
  },
  "author": "aluno",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.18.3",
    "express": "^4.16.3",
    "mongodb": "^2.0.0",
    "mongoose": "^3.0.5",
    "mysql": "^2.16.0",
    "nodemon": "^1.18.10"
  }
}
```

Instalar as dependências.

Abrir o terminal e digitar:

npm install



```
PS C:\Temp\documentar para aula\Aula06_server> npm install

> nodemon@1.19.1 postinstall C:\Temp\documentar para aula\Aula06_server\node_modules\nodemon
> node bin/postinstall || exit 0

npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN mysql@1.0.0 No repository field.
npm WARN mysql@1.0.0 script should probably be scripts.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

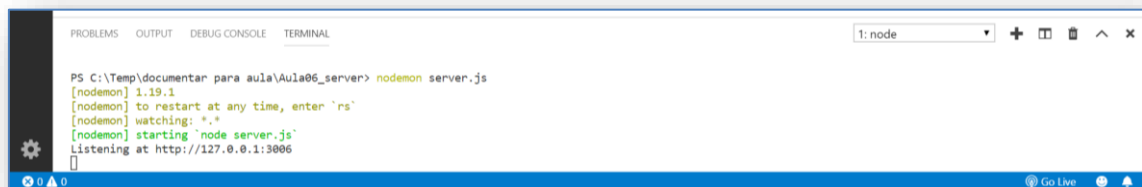
added 278 packages from 177 contributors and audited 2453 packages in 17.535s
found 0 vulnerabilities

PS C:\Temp\documentar para aula\Aula06_server>
```

Iniciar o servidor:

Digitar no terminal:

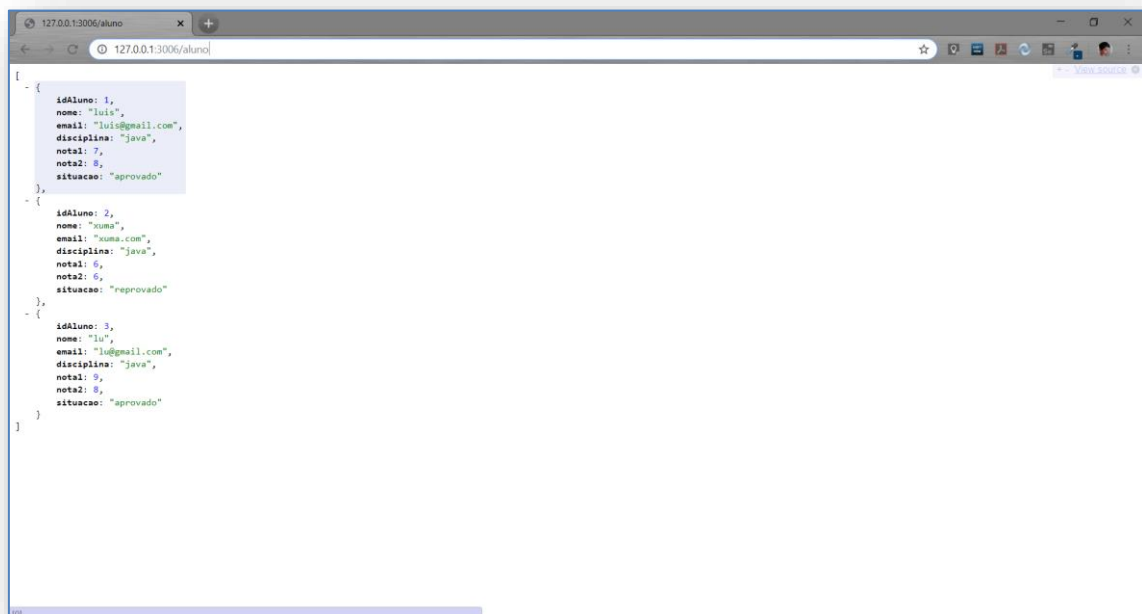
nodemon server.js



```
PS C:\Temp\documentar para aula\Aula06_server> nodemon server.js

[nodemon] 1.19.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
Listening at http://127.0.0.1:3006
```

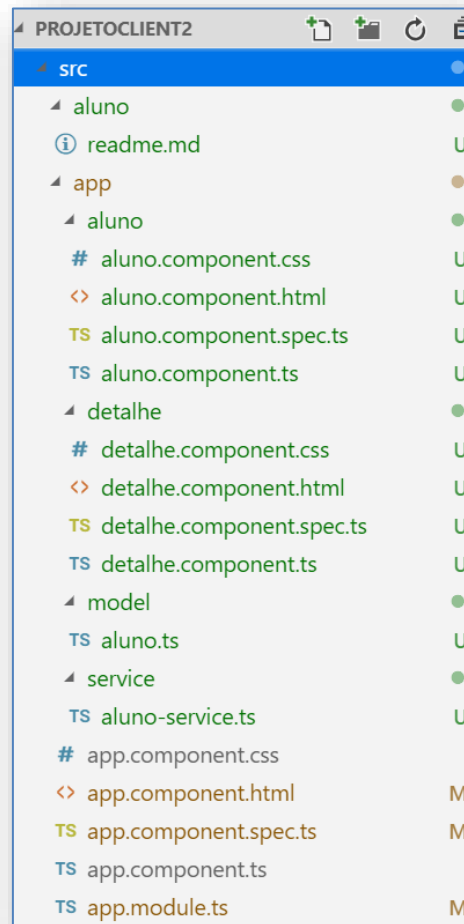
<http://127.0.0.1:3006/aluno>



```
[{"idAluno": 1, "nome": "luis", "email": "luis@gmail.com", "disciplina": "java", "nota1": 7, "nota2": 8, "situacao": "aprovado"}, {"idAluno": 2, "nome": "soma", "email": "soma.com", "disciplina": "java", "nota1": 6, "nota2": 6, "situacao": "reprovado"}, {"idAluno": 3, "nome": "lu", "email": "lu@gmail.com", "disciplina": "java", "nota1": 9, "nota2": 8, "situacao": "aprovado"}]
```

PROJETO CLIENT DETALHES

Estrutura do projeto depois de finalizado:



aluno.ts

```
export class Aluno{

  idAluno : number;
  nome : string;
  email : string;
  disciplina : string;
  nota1 : number;
  nota2 : number;
  situacao : string;

  constructor(idAluno ? : number, nome ? :string, email ? :string,
    disciplina ? : string, nota1 ? : number, nota2 ? :number,
```

```

    situacao ? :string ){
this.idAluno =idAluno;
this.nome =nome;
this.email = email;
this.disciplina = disciplina;
this.nota1 = nota1;
this.nota2 = nota2;
this.situacao = situacao
}

isNota1():boolean{
    if (this.nota1<0 || this.nota1>10){
        return false;
    }
    return true;
}

isNota2():boolean{
    if (this.nota2<0 || this.nota2>10){
        return false;
    }
    return true;
}

isSituacao():boolean{
    if (this.situacao=='aprovado' ||
this.situacao=='reprovado'){
        return true;
    }
    return false;
}
}

```

readme.md

```
ng g c aluno
```

```
ng g c detalhe
```

```
ng serve -o --port 2244
```

```
npm install primeng --save
```

```
npm i primeicons --save
```

```
npm i @angular/animations --save
```

```
npm i @fortawesome/fontawesome-free
```

aluno.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Aluno } from "../model/aluno";
```

//ATENÇÃO PARA O A URL DO PROJETO

```
const URL = 'http://localhost:3006/aluno';
```

```
@Injectable()
```

```
export class AlunoService {
```

```
    constructor(private http: HttpClient) {
    }
```

```
    public create(aluno: Aluno) {
        return this.http.post<Aluno>(`${URL}`, aluno);
    }
```

```
    public findAll() {
        return this.http.get<Aluno[]>(`${URL}`);
    }
```

```
    public delete(id: number) {
```

```

        return this.http.delete(`${URL}/${id}`);
    }
    public findBycode(id: number) {
        return this.http.get<Aluno>(`${URL}/${id}`);
    }
}

```

aluno.component.ts

```

import { AlunoService } from '../service/aluno-service';
import { Component, OnInit } from '@angular/core';
import { Aluno } from '../model/aluno';
import { Router } from '@angular/router';

@Component({
    selector: 'app-aluno',
    templateUrl: './aluno.component.html',
    styleUrls: ['./aluno.component.css']
})
export class AlunoComponent implements OnInit {

    alunoSelect: Aluno;
    aluno: Aluno;
    alunos: Aluno[] = [];
    msgerro: string;

    constructor(private service: AlunoService,
        private router: Router) {
        this.aluno = new Aluno();
        this.alunoSelect = new Aluno();
    }

    ngOnInit() {
        this.listar();
    }

    public listar() {
        this.service.findAll().subscribe(res => {
            this.alunos = res
        });
    }
}

```

```

    }

    public mostrar() {
        this.router.navigateByUrl(`detalhe/${this.alunoSelect.id
Aluno}`);
    }

    public gravar() {
        this.service.create(this.aluno).subscribe(res => {
            this.aluno = new Aluno();
            this.listar();
        })
    }

    public filtrarCod(id: number) {
        return this.alunos.filter(c => c.idAluno == id).
            map(c => {
                this.alunoSelect = c;
                alert(this.alunoSelect);
            })
    }

    public excluir(id: number) {
        this.service.delete(id).subscribe(res => {
            this.listar();
        }, err => {
            this.msgerro = 'Codigo nao encontrado ...';
            console.log('error :', err.error);
        })
    }

    //select nome,email from aluno; MAP
    // select Upper(nome) from aluno; MAP
    //select * from aluno where id=1; filter
}

```

aluno.component.html

```
<h2>Designer Maravilhoso</h2>
```

```
<div id="aluno">
  <p-panel header="Componente Aluno" [toggleable]="true">
    Seleção o aluno:

    <select [(ngModel)]="alunoSelect" name="alunoSelect">
      <option [ngValue]="{}">Selecione</option>
      <option *ngFor="let item of alunos"
[ngValue]="item">
        {{item.idAluno}},{{item.nome}},{{item.disciplina
}}
      </option>
    </select>

    <p>Aluno selecionado: <b>{{alunoSelect.nome}}</b></p>
    Detalhe do aluno:
    <a routerLink="../detalhe/{{alunoSelect.idAluno}}"
title="Clique para ver Detalhes">
      {{alunoSelect.nome}}</a>
    </p-panel>
</div>
```

```
<div id="gravar" style="margin-top: 10px;">
  <p-panel header="Gravação de aluno" [toggleable]="true">

    <div class="ui-md-4">
      <div class="ui-inputgroup">
        <span class="ui-inputgroup-addon"><i class="fa
fa-user"></i></span>
        <input pInputText type="text" name="nome"
[(ngModel)]="aluno.nome" placeholder="nome">
      </div>
    </div>

    <div class="ui-md-4">
      <div class="ui-inputgroup">
```

```

        <span class="ui-inputgroup-addon"><i class="fa
fa-envelope"></i></span>
        <input pInputText type="email" name="email"
[(ngModel)]= "aluno.email" placeholder="email">
    </div>
</div>

<div class="ui-md-4">
    <div class="ui-inputgroup">
        <span class="ui-inputgroup-addon"><i class="fa
fa-book-open"></i></span>
        <input pInputText type="text" name="disciplina"
[(ngModel)]= "aluno.disciplina" placeholder="disciplina">
    </div>
</div>

<div class="ui-md-4">
    <div class="ui-inputgroup">
        <span class="ui-inputgroup-addon"><i class="fas
fa-keyboard"></i></span>
        <input pInputText type="number" name="nota1"
[(ngModel)]= "aluno.nota1" placeholder="nota1">
    </div>
</div>

<div class="ui-md-4">
    <div class="ui-inputgroup">
        <span class="ui-inputgroup-addon"><i class="fas
fa-keyboard"></i></span>
        <input pInputText type="number" name="nota2"
[(ngModel)]= "aluno.nota2" placeholder="nota2">
    </div>
</div>

<div class="ui-md-4">
    <div class="ui-inputgroup">
        <span class="ui-inputgroup-addon"><i class="fas
fa-thumbs-up"></i></span>
        <input pInputText type="text" name="situacao"
[(ngModel)]= "aluno.situacao" placeholder="situacao">
    </div>
</div>

```

```

        <button pButton (click)="gravar()" class="ui-button-
info" label="Gravar Aluno"></button>

    </p-panel>
</div>

<h2><a routerLink="../detalhe/{{alunoSelect.idAluno}}">

    {{alunoSelect.nome}}</a>
</h2>
<hr />

<select [(ngModel)]="alunoSelect" name="alunoSelect">
    <option [ngValue]="{}">Selecione</option>
    <option *ngFor="let item of alunos" [ngValue]="item">
        {{item.idAluno}},{{item.nome}},{{item.disciplina}}
    </option>
</select>

<br />
<h2>Gravar</h2>
<input type="text" name="nome" [(ngModel)]="aluno.nome"
placeholder="nome" />
<br />
<input type="text" name="email" [(ngModel)]="aluno.email"
placeholder="email" />
<br />
<input type="text" name="disciplina"
[(ngModel)]="aluno.disciplina" placeholder="Disciplina" />
<br />
<input type="number" name="nota1" [(ngModel)]="aluno.nota1"
placeholder="nota1" />
<br />
<input type="number" name="nota2" [(ngModel)]="aluno.nota2"
placeholder="nota2" />
<br />
<input type="text" name="situacao" [(ngModel)]="aluno.situacao"
placeholder="situacao" />
<br />
<button (click)="gravar()">Gravar</button>

```

```

<h2>Toque</h2>
<li *ngFor="let linha of alunos">
  {{linha | json}}
</li>
<br />
<button (click)="mostrar();">Detalhe</button>

```

detalhe.component.ts

```

import { AlunoService } from '../service/aluno-service';
import { Component, OnInit } from '@angular/core';
import { Aluno } from '../model/aluno';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-detalhe',
  templateUrl: './detalhe.component.html',
  styleUrls: ['./detalhe.component.css']
})
export class DetalheComponent implements OnInit {

  id: number;
  alunoSelecionado: Aluno;
  alunos: Aluno[] = [];

  constructor(private services: AlunoService,
    private route: ActivatedRoute) {
    this.alunoSelecionado = new Aluno();
    this.loadAlunos();
  }

  ngOnInit() {
    this.route.params.subscribe((param) => {
      let id = param.id;
      this.services.findBycode(+id).subscribe(res => {
        this.alunoSelecionado = res;
      })
      console.log(this.alunoSelecionado)
    })
  }
}

```

```

    loadAlunos() {
        this.services.findAll().subscribe(res => {
            this.alunos = res;
        })
    }
}

```

detalhe.component.html

```
{{alunoSelecionado | json}}
```

app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule, Component } from '@angular/core';
import { AppComponent } from './app.component';
import { AlunoComponent } from './aluno/aluno.component';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { DetalheComponent } from './detalhe/detalhe.component';
import { RouterModule } from '@angular/router';
import { AlunoService } from './service/aluno-service';

import { ToolbarModule } from 'primeng/toolbar';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { ButtonModule } from 'primeng/button';
import { PanelModule } from 'primeng/panel';
import { InputTextModule } from 'primeng/inputtext';
import { MessageModule } from 'primeng/message';

@NgModule({
  declarations: [
    AppComponent,
    AlunoComponent,
    DetalheComponent

```

```

    ],
    imports: [
        HttpClientModule,
        FormsModule,
        BrowserModule,
        RouterModule.forRoot([
            { path: '', redirectTo: 'aluno', pathMatch: 'full' },
            { path: 'aluno', component: AlunoComponent },
            { path: 'detalhe/:id', component: DetalheComponent }
        ]),
        ToolbarModule,
        BrowserAnimationsModule,
        ButtonModule,
        PanelModule,
        InputTextModule,
        MessageModule
    ],
    providers: [AlunoService],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

app.component.html

```
<router-outlet></router-outlet>
```

angular.json

```

{
  ...
  "styles": [
    "src/styles.css",
    "node_modules/primeng/resources/primeng.css",
    "node_modules/primeng/resources/themes/nova-colored/theme.css",
    "node_modules/primeicons/primeicons.css",

```

```
"node_modules/@fortawesome/fontawesome-free/css/all.css"
  ],
  "scripts": []
},...
```

Para rodar o projeto.

Digitar no terminal:

ng s -o

<http://localhost:4200/aluno>

Client

localhost:4200/aluno

Designer Maravilhoso

Componente Aluno

Selecione o aluno:

Aluno selecionado: 1.luis.java

Detalhe do aluno: 2.xuma.java

3.lu.java

Gravação de aluno

Gravar

Client

localhost:4200/aluno

nome

email

disciplina

nota1

nota2

situacao

Gravar Aluno

Gravar

nome

email

Disciplina

nota1

nota2

situacao

Gravar

Toque

- { "idAluno": 1, "nome": "luis", "email": "luis@gmail.com", "disciplina": "java", "nota1": 7, "nota2": 8, "situacao": "aprovado" }
- { "idAluno": 2, "nome": "xuma", "email": "xuma.com", "disciplina": "java", "nota1": 6, "nota2": 6, "situacao": "reprovado" }
- { "idAluno": 3, "nome": "lu", "email": "lu@gmail.com", "disciplina": "java", "nota1": 9, "nota2": 8, "situacao": "aprovado" }

Detalhe

Selecionar um aluno.

Client

localhost:4200/aluno

Designer Maravilhoso

Componente Aluno

Selecione o aluno: 1, luis.java

Aluno selecionado: luis

Detalhe do aluno: luis

Gravação de aluno

nome

email

disciplina

nota1

nota2

situacao

Gravar Aluno

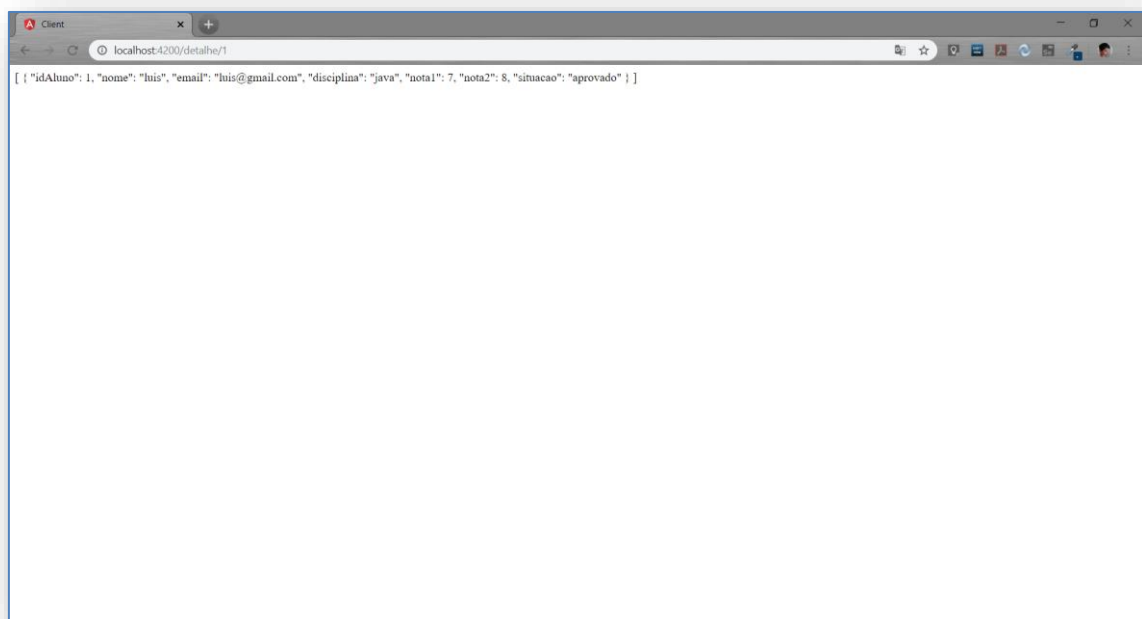
luis

1, luis.java

Gravar

nome

Clicar no link gerado para ir para detalhes



Digitando os dados para gravar:

A screenshot of a web application titled "Designer Maravilhoso". The application has two main sections: "Componente Aluno" and "Gravação de aluno".

Componente Aluno: This section contains a dropdown menu labeled "Selecione o aluno:". Below it, the text "Aluno selecionado:" and "Detalhe do aluno:" is displayed.

Gravação de aluno: This section contains a form with the following fields:

- Nome: "belem"
- E-mail: "belem@gmail.com"
- Disciplina: "java"
- Nota 1: "10"
- Nota 2: "9"
- Situação: "aprovada" (with a red border around the text)

Below the form is a blue button labeled "Gravar Aluno".

Gravar: This section contains a list of the saved data:

- belem
- belem@gmail.com
- java

Clicando em gravar. Mostra a lista atualizada

email

disciplina

nota1

nota2

situacao

Gravar Aluno

Selecionar

Gravar

nome

email

Disciplina

nota1

nota2

situacao

Gravar

Toque

- { "idAluno": 1, "nome": "luis", "email": "luis@gmail.com", "disciplina": "java", "nota1": 7, "nota2": 8, "situacao": "aprovado" }
- { "idAluno": 2, "nome": "xuma", "email": "xuma.com", "disciplina": "java", "nota1": 6, "nota2": 6, "situacao": "reprovado" }
- { "idAluno": 3, "nome": "lu", "email": "lu@gmail.com", "disciplina": "java", "nota1": 9, "nota2": 8, "situacao": "aprovado" }
- { "idAluno": 4, "nome": "belem", "email": "belem@gmail.com", "disciplina": "java", "nota1": 10, "nota2": 9, "situacao": "aprovado" }

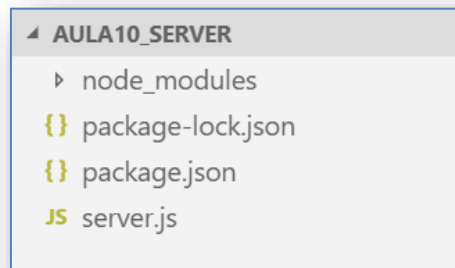
Detalhe

PROJETO SERVIDOR:

Passo a passo:

- Criar um diretório para armazenar o servidor
- Entrar nesse diretório com o Visual Code
- Criar os seguintes arquivos:
 - Server.js
 - Package.json

Estrutura do projeto depois de finalizado:



server.js

```
const express = require('express');
const body = require('body-parser');
const app = express();
const port = 3007;
const secret = "minhachave";
const jwt = require('jsonwebtoken');

app.use(body.json());
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Methods",
"POST,GET,DELETE,PUT,OPTIONS");
  res.header("Access-Control-Allow-Headers", "Origin, X-
Requested-With, Content-Type, Accept, authorization");
  res.header("Cache-Control", "no-cache");
  res.header("Pragma", "no-cache");
  next();
});
```

```

app.get('/criptografia/:login', (req, res) => {
  let login = req.params.login;
  if (login == 'camila' || login == 'lo' || login == 'lu') {
    const crypto = require('crypto');
    var resp2 =
crypto.createHash('md5').update(login).digest('hex');
    res.send(resp2);
  } else {
    res.status(403).send('error');
    return;
  }
});

app.get('/generate/:login/:senha', (req, res) => {
  let login = req.params.login;
  let senha = req.params.senha;
  let resp = null;
  if ((login == 'lu' && senha == '123') || (login == 'camila'
&& senha == '123') || (login == 'lo' && senha == '123')) {
    resp = jwt.sign({ login: login, senha: senha }, secret);
  } else {
    resp = 'fail';
  }
  res.send(resp);
});

app.post('/check', (req, res) => {
  let token = req.body.token;
  jwt.verify(token, secret, (err, dados) => {
    if (err) {
      res.status(403).send('error');
      return;
    } else {
      console.log(dados);
      res.send(dados);
    }
  });
});

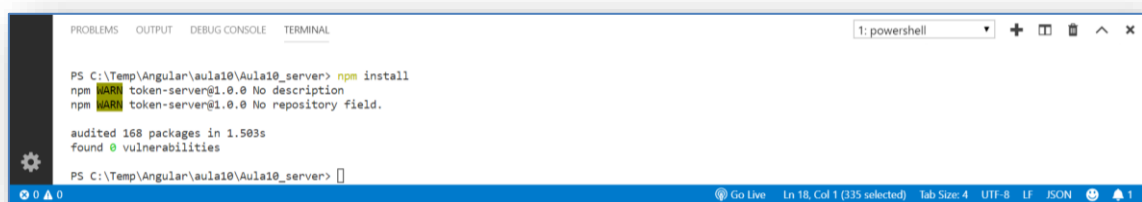
app.listen(port, () =>
  console.log(`Executando a Porta ${port}`));

```

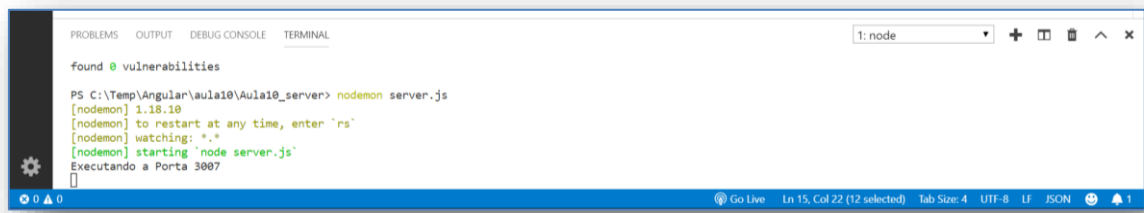
Package.json

```
{
  "name": "token-server",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "voces",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.18.3",
    "express": "^4.16.4",
    "jsonwebtoken": "^8.5.0"
  }
}
```

Instalar as dependencias:
No terminal, digitar: “npm install”



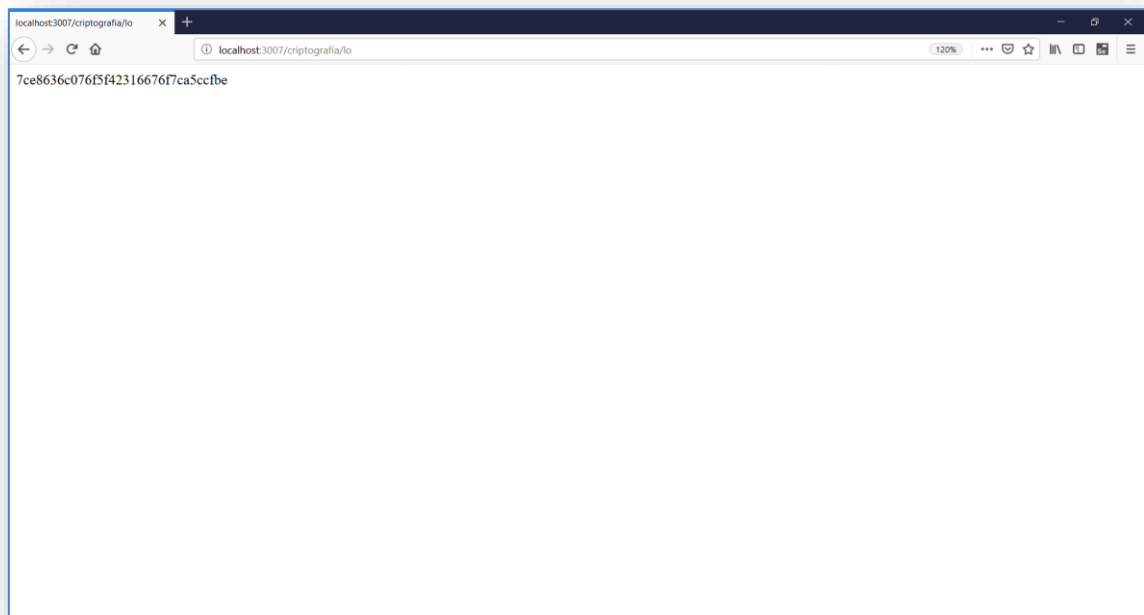
Rodando o servidor:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
found 0 vulnerabilities

PS C:\Temp\Angular\aula10\Aula10_server> nodemon server.js
[nodemon] 1.18.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
Executando a Porta 3007
```

<http://localhost:3007/criptografia/lo>



PROJETO ANGULAR:

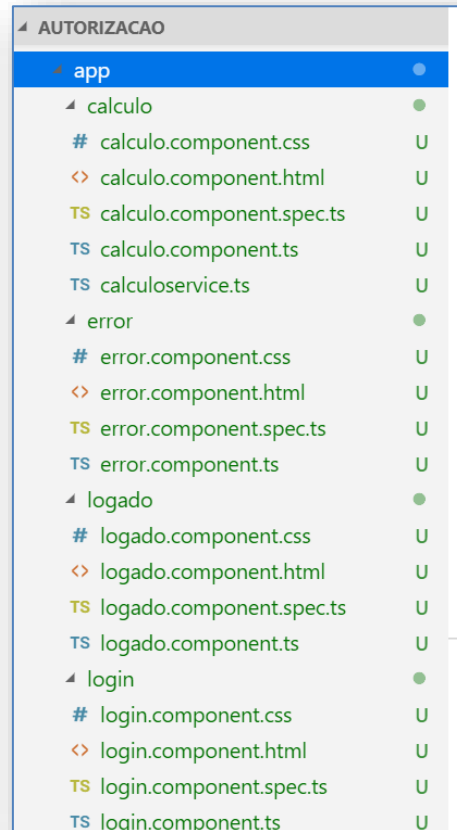
Para criar o projeto:

Criar uma pasta para trabalhar com o projeto.

Pelo prompt(terminal), entrar nessa pasta.

Digitar o comando para criação do projeto:

“ng new nome_do_projeto”



Criar os seguintes componentes:

- Calculo
- Error
- Logado
- Login

Para criar os componentes automaticamente, digitar no terminal:

“ng g c nome_do_componente” ou “ng generate component nome_do_componente”

Login.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

const url = 'http://localhost:3007';

@Injectable()
export class LoginService {

    constructor(private http: HttpClient) {
    }

    criptografia(login: string): Observable<string> {
        return
this.http.get<string>(`${url}/criptografia/${login}`,
        { responseType: 'text' as 'json' });
    }

    generateToken(login: string, senha: string):
    Observable<string> {
        return
this.http.get<string>(`${url}/generate/${login}/${senha}`,
        { responseType: 'text' as 'json' });
    }

    check(token: any): any {
        return this.http.post<string>(`${url}/check`, { token },
        { responseType: 'text' as 'json' });
    }
}
```

Login.ts

```
export class login {  
    login: string;  
    senha: string;  
}
```

Login.component.ts

```
import { LoginService } from '../service/login.service';  
import { Component, OnInit } from '@angular/core';  
import { Router } from '@angular/router';  
  
@Component({  
    selector: 'app-login',  
    templateUrl: './login.component.html',  
    styleUrls: ['./login.component.css']  
})  
export class LoginComponent implements OnInit {  
  
    logincriptografado: string;  
    login: string;  
    senha: string;  
    token: string;  
    checkString: string;  
  
    constructor(private service: LoginService,  
                private router: Router) {  
    }  
  
    ngOnInit() {  
    }  
  
    crypto() {  
        this.service.criptografia(this.login).subscribe(res => {  
            this.logincriptografado = res;  
        });  
    }  
}
```

```

enviar() {
    this.service.generateToken(this.login, this.senha).
        subscribe(res => {
            this.token = res
        });
}

check() {
    this.service.check(this.token).subscribe(res => {
        this.checkString = res;
    })
}

entran() {
    if (this.token === 'fail') {
        this.router.navigateByUrl('/error');
    } else if (!this.checkString) {
        this.router.navigateByUrl('/error');
    } else {
        sessionStorage.setItem("msg", "Seja Bem Vindo ao Login");
        sessionStorage.setItem("login", this.login);
        sessionStorage.setItem("senha", this.senha);
        this.router.navigateByUrl('/logado');
    }
}
}
}

```

Login.component.html

```

<form>
  Login <br />
  <input type="text" [(ngModel)]="login" name="login" />
  <br /> Senha <br />
  <input type="text" [(ngModel)]="senha" name="senha" />
  <br /><br />

  <button (click)="crypto()" type="button">
    Criptografar</button>
  <button (click)="enviar()" type="button">Enviar</button>

```

```

        <button (click)="check()" type="button">Checar
Criptografia</button>
        <button (click)="entrar()" type="button">Entrar</button>

<br /><br />
LoginCriptografado : {{logincriptografado}}
<br />
Token Recebido :{{token}}
<br />
Token Decodificado :{{checkString}}

</form>

```

Logado.component.ts

```

import { Component, OnInit } from '@angular/core';
import { LoginService } from '../service/login.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-logado',
  templateUrl: './logado.component.html',
  styleUrls: ['./logado.component.css']
})
export class LogadoComponent implements OnInit {

  checkString: string;
  login: string;
  senha: string;
  msg: String;

  ngOnInit() {
  }

  constructor(private service: LoginService,
    private router: Router
  ) {
    if (sessionStorage.getItem("login") == "") {
      this.router.navigateByUrl('/error');
    }
  }

```

```

    }
    this.msg = sessionStorage.getItem("msg");
    this.senha = sessionStorage.getItem("senha");
    this.login = sessionStorage.getItem("login");
  }

  logout() {
    sessionStorage.setItem("msg", "");
    sessionStorage.setItem("senha", "");
    sessionStorage.setItem("login", "");
    sessionStorage.setItem("checkString", "");
    this.router.navigateByUrl('/login');
  }
}

```

Logado.component.html

```

<div>
  <h2>Bem vindo ao Sistema de Logado</h2>
  <h4>{{msg}}</h4>
  <br />

  {{login}},{{senha}}

  <br />
  <button (click)="logout()" type="button">Logout</button>

</div>

```

Error.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-error',
  templateUrl: './error.component.html',
  styleUrls: ['./error.component.css']
})
export class ErrorComponent implements OnInit {

  texto: string = "Dados Invalidos, Entre em contato com Adm";

  constructor() { }

  ngOnInit() {
  }

}
```

Error.component.html

```
<h2> Error !!!! </h2>
<hr />
<br />
{{texto}}
<br />
<a routerLink="/login">Voltar Login</a>
```

App.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import { LogadoComponent } from './logado/logado.component';
import { ErrorComponent } from './error/error.component';
import { HttpClientModule } from '@angular/common/http';
import { FormsModule } from '@angular/forms';
import { RouterModule } from '@angular/router';
import { LoginService } from './service/login.service';
import { CalculoComponent } from './calculo/calculo.component';

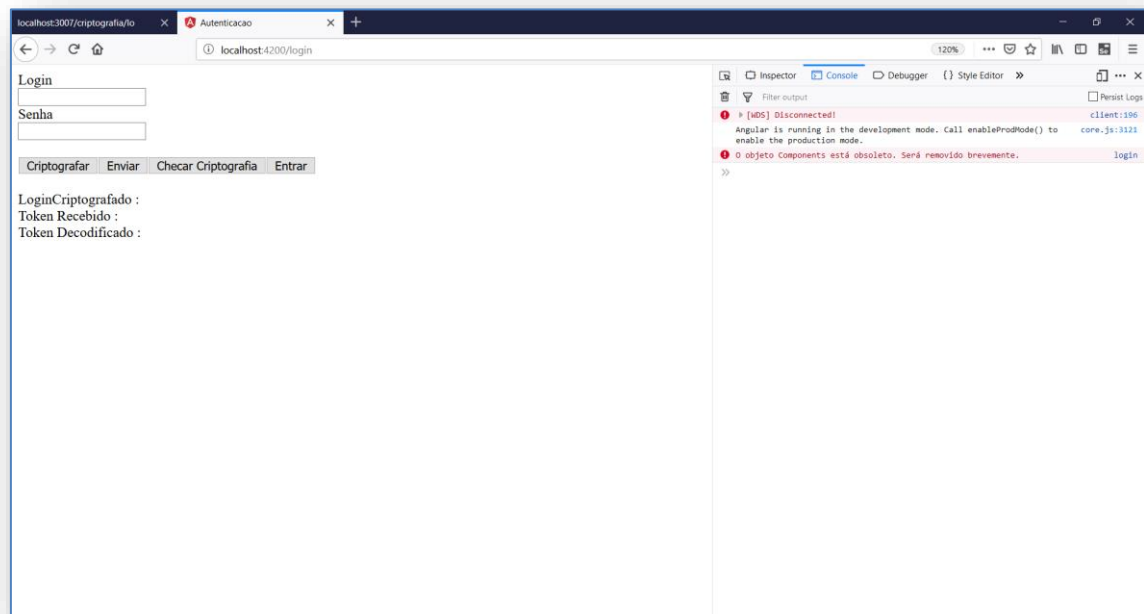
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    LogadoComponent,
    ErrorComponent,
    CalculoComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule,
    RouterModule.forRoot([
      { path: '', redirectTo: '/login', pathMatch: 'full' },
      { path: 'login', component: LoginComponent },
      { path: 'logado', component: LogadoComponent },
      { path: 'error', component: ErrorComponent },
      { path: '**', component: ErrorComponent }
    ])
  ],
  providers: [LoginService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

App.component.html

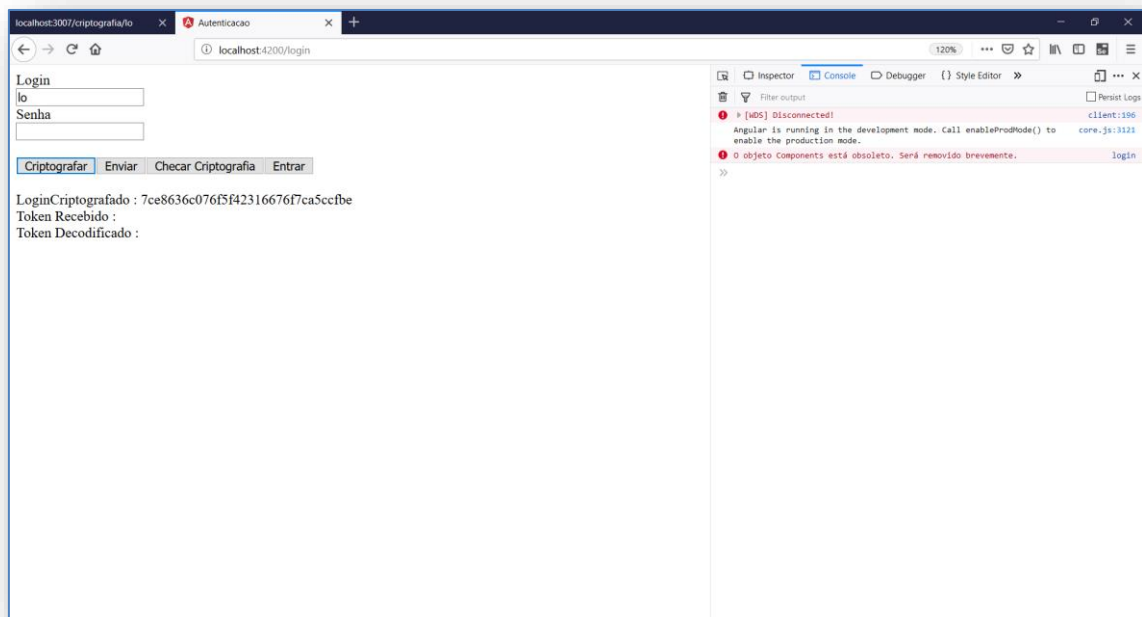
```
<router-outlet></router-outlet>
```

Rodando o projeto

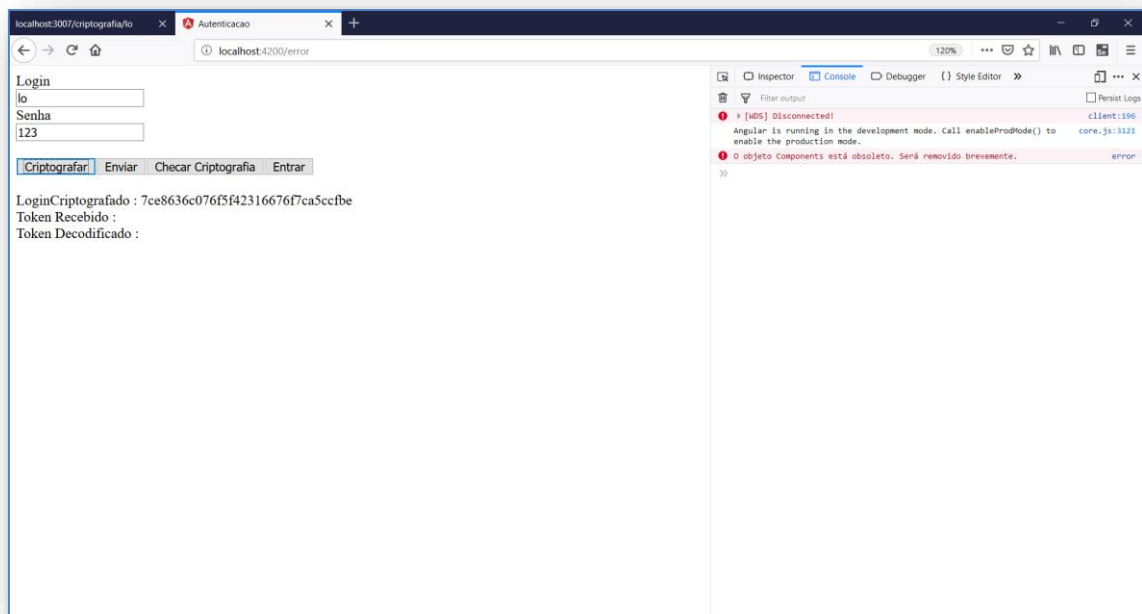
<http://localhost:4200/login>



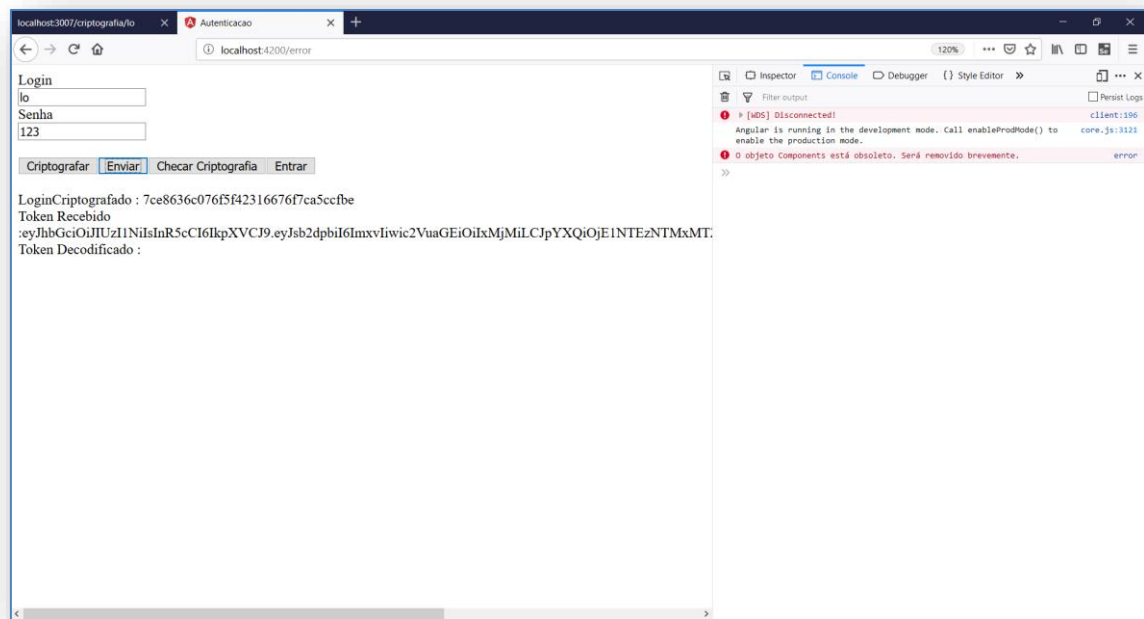
Digitar um nome de usuário e clicar em criptografar



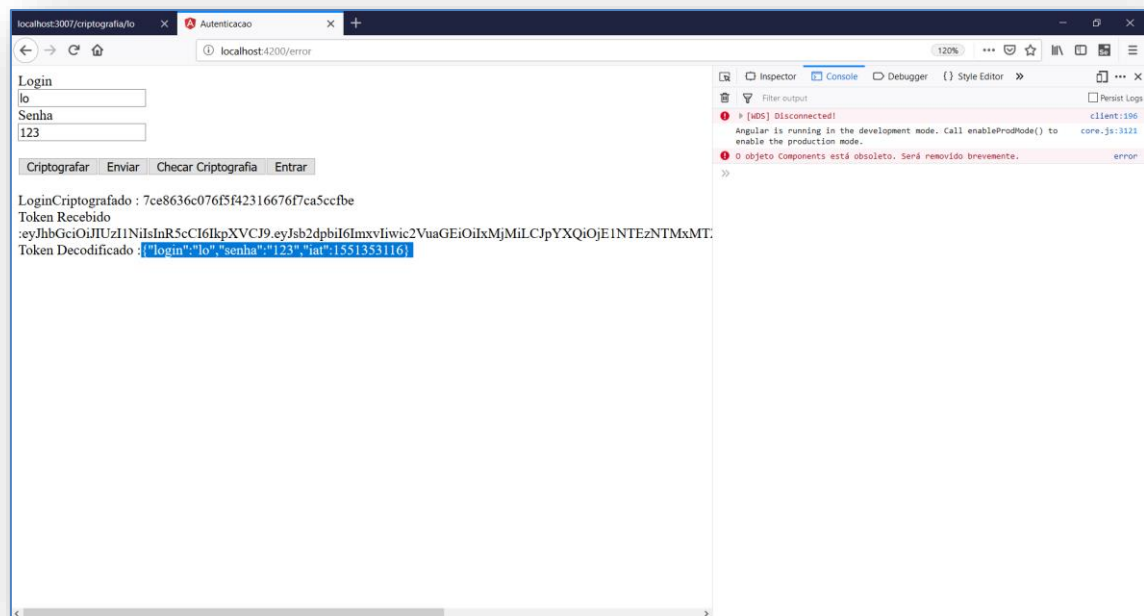
Digitar a senha



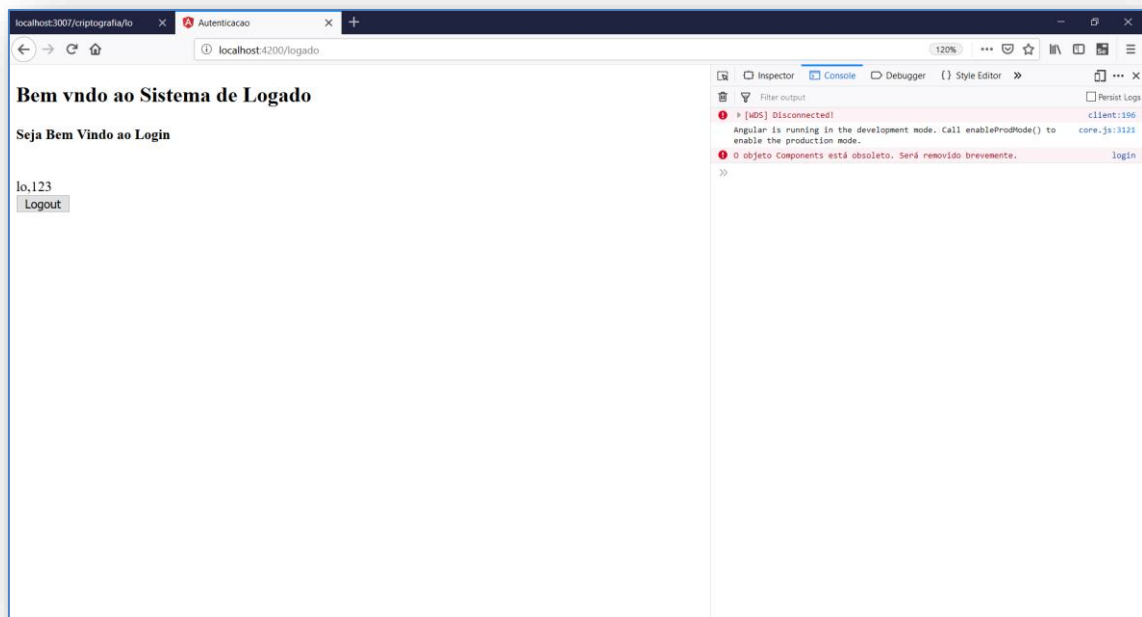
Clicar em enviar



Clicar em checar criptografia

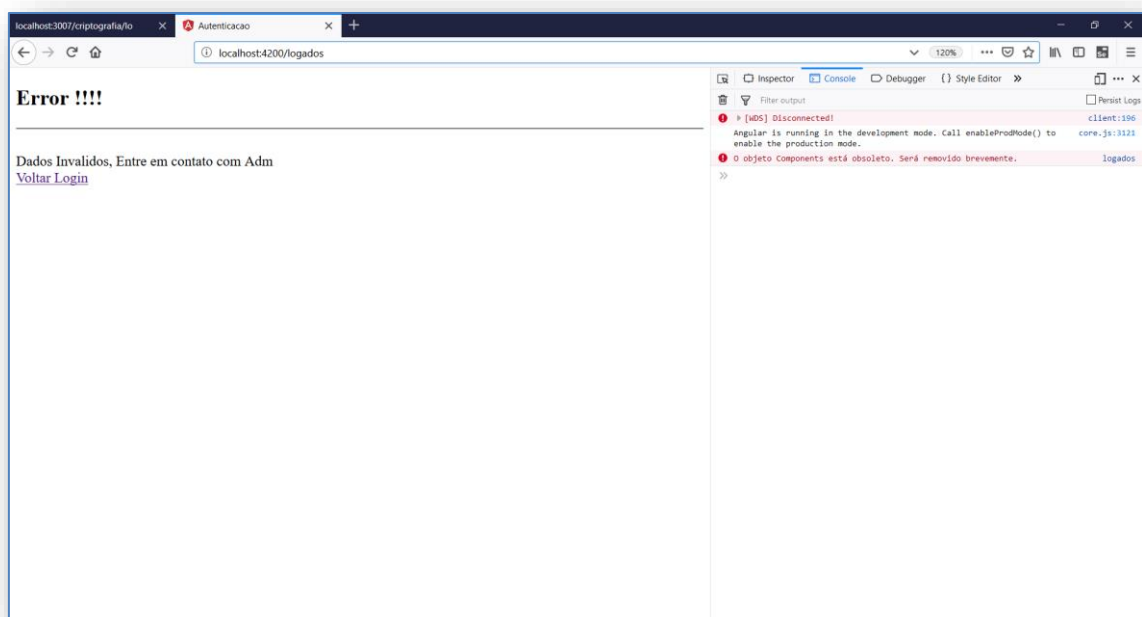


Clicar em entrar



Se tentar entrar pela url ou uma URL que não existe cai no erro

<http://localhost:4200/logados>



Calculoservice.ts

```
import { Injectable } from "@angular/core";

@Injectable({
  providedIn: 'root'
})
export class CalculoService {

  constructor() { }

  soma(a: number, b: number) {
    return a + b;
  }

  multiply(a: number, b: number) {
    return a * b;
  }

  //cursos que seguem o mesmo sentido do javascript
  //de graca (ecma6) _ javascript
  //(angular6) refazer
  //visualAngular
  //teste-Front
  //Mean
  //ionic 3
  //mongodb

  subtrair(a: number, b: number): Promise<number> {
    return new Promise((resolve, reject) => {
      if (a == null || b == null) {
        reject(-1);
        return;
      }
      resolve(a - b);
    });
  }
}
```

Calculo.component.spec.ts

```
import { TestBed } from '@angular/core/testing';
import { CalculoService } from '../calculoservice';
```

```
describe('CalculoService', () => {
  let service: CalculoService;
  beforeEach(() => {
    TestBed.configureTestingModule({})
    service = TestBed.get(CalculoService)
  });
```

```
  it('should be created', () => {
    expect(service).toBeTruthy();
  });
```

```
  it('Teste de Servico de soma :', () => {
    expect(10).toBe(service.soma(5, 5));
  });
```

//NESSE TESTE EXISTE UM ERRO. O RESULTADO CORRETO DA MULTIPLICAÇÃO É 50 E COLOCAMOS 100

```
  it('Teste de Servico de Multiplicacao :', () => {
    expect(100).toBe(service.multiply(5, 10));
  });
```

```
  it('Teste de Servico de Subtracao Asincrono :', async () => {
    expect(10).toBe(await service.subtrair(20, 10));
  });
```

```
  it('Teste de Servico de Subtracao sincrono:',
    (done: DoneFn) => {
      service.subtrair(100, 50).then((result) => {
        expect(50).toBe(result);
        done();
      })
    });
```

```
});
```

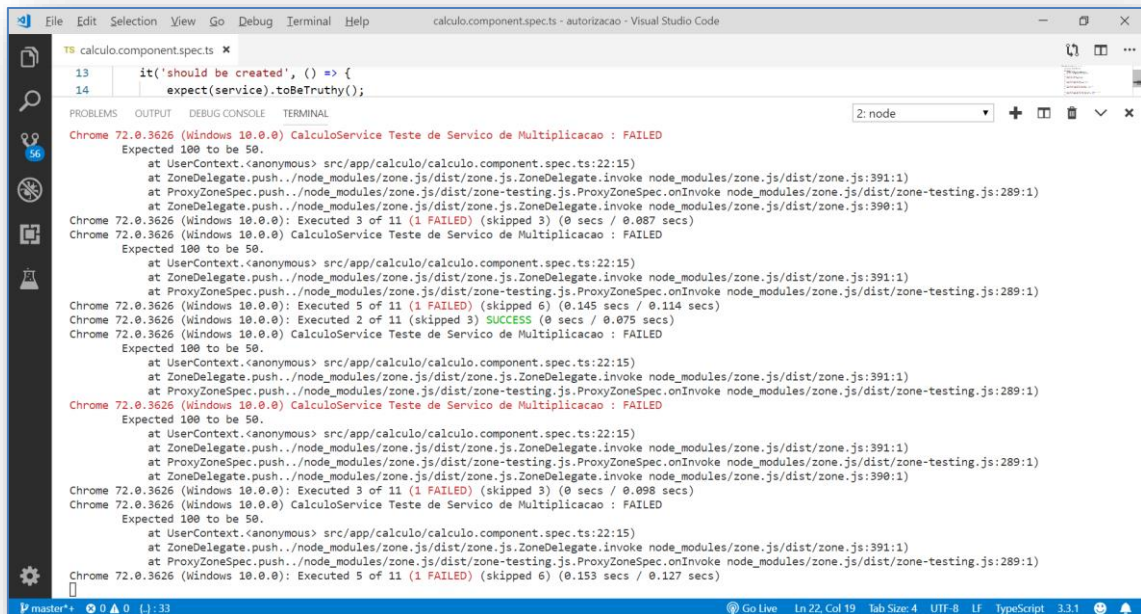
Para executar o teste:

- Em “app.component.html”, comentar a tag <router-outlet>
- Em todos os componentes que não serão testados, desabilitar os testes colocando a letra “x” na frente da palavra “describe”



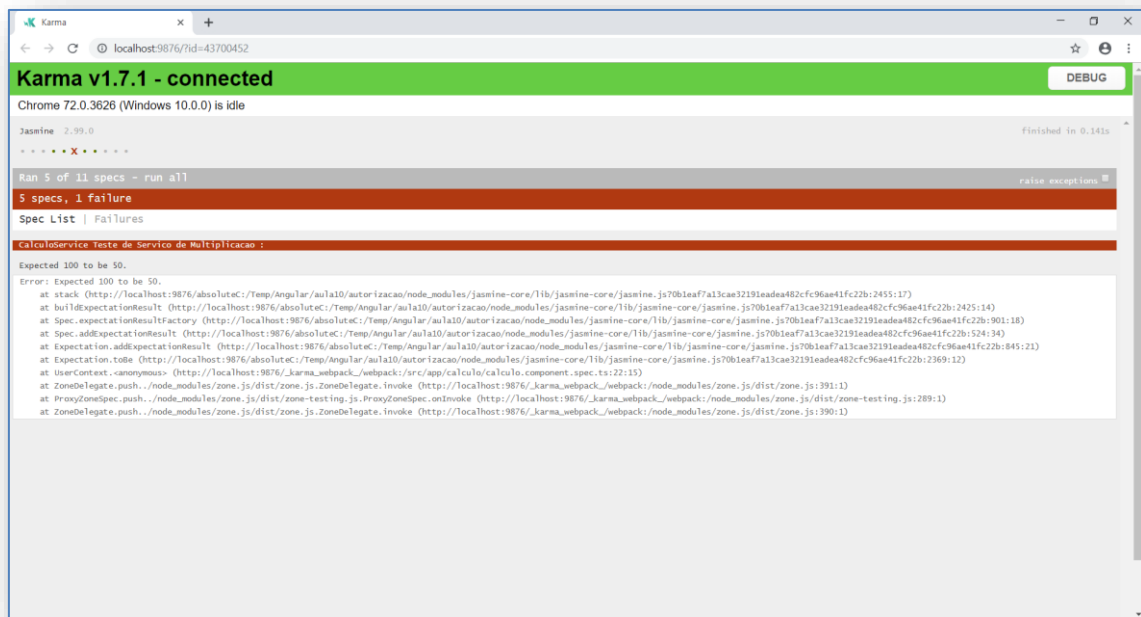
```
1 import { async, ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { ErrorComponent } from './error.component';
4
5 xdescribe('ErrorComponent', () => {
6   let component: ErrorComponent;
7   let fixture: ComponentFixture<ErrorComponent>;
8
9   beforeEach(async(() => {
10     TestBed.configureTestingModule({
11       declarations: [ ErrorComponent ]
12     })
13     .compileComponents();
14   }));
15
16   beforeEach(() => {
17     fixture = TestBed.createComponent(ErrorComponent);
18     component = fixture.componentInstance;
19     fixture.detectChanges();
20   });
21 })
```

No terminal, digitar: “ng test”



```
13 it('should be created', () => {
14   expect(service).toBeTruthy();
15 })
16
17 Chrome 72.0.3626 (Windows 10.0.0) CalculoService Teste de Servico de Multiplicacao : FAILED
18   Expected 100 to be 50.
19   at UserContext.<anonymous> src/app/calculo/calculo.component.spec.ts:22:15
20   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:391:1
21   at ProxyZoneSpec.push.../node_modules/zone.js/dist/zone-testing.js.ProxyZoneSpec.onInvoke node_modules/zone.js/dist/zone-testing.js:289:1
22   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:390:1
23 Chrome 72.0.3626 (Windows 10.0.0): Executed 3 of 11 (1 FAILED) (skipped 3) (0 secs / 0.087 secs)
24 Chrome 72.0.3626 (Windows 10.0.0) CalculoService Teste de Servico de Multiplicacao : FAILED
25   Expected 100 to be 50.
26   at UserContext.<anonymous> src/app/calculo/calculo.component.spec.ts:22:15
27   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:391:1
28   at ProxyZoneSpec.push.../node_modules/zone.js/dist/zone-testing.js.ProxyZoneSpec.onInvoke node_modules/zone.js/dist/zone-testing.js:289:1
29   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:390:1
30 Chrome 72.0.3626 (Windows 10.0.0): Executed 5 of 11 (1 FAILED) (skipped 6) (0.145 secs / 0.114 secs)
31 Chrome 72.0.3626 (Windows 10.0.0): Executed 2 of 11 (skipped 3) SUCCESS (0 secs / 0.075 secs)
32 Chrome 72.0.3626 (Windows 10.0.0) CalculoService Teste de Servico de Multiplicacao : FAILED
33   Expected 100 to be 50.
34   at UserContext.<anonymous> src/app/calculo/calculo.component.spec.ts:22:15
35   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:391:1
36   at ProxyZoneSpec.push.../node_modules/zone.js/dist/zone-testing.js.ProxyZoneSpec.onInvoke node_modules/zone.js/dist/zone-testing.js:289:1
37   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:390:1
38 Chrome 72.0.3626 (Windows 10.0.0): Executed 3 of 11 (1 FAILED) (skipped 3) (0 secs / 0.098 secs)
39 Chrome 72.0.3626 (Windows 10.0.0) CalculoService Teste de Servico de Multiplicacao : FAILED
40   Expected 100 to be 50.
41   at UserContext.<anonymous> src/app/calculo/calculo.component.spec.ts:22:15
42   at ZoneDelegate.push.../node_modules/zone.js/dist/zone.js.ZoneDelegate.invoke node_modules/zone.js/dist/zone.js:391:1
43   at ProxyZoneSpec.push.../node_modules/zone.js/dist/zone-testing.js.ProxyZoneSpec.onInvoke node_modules/zone.js/dist/zone-testing.js:289:1
44 Chrome 72.0.3626 (Windows 10.0.0): Executed 5 of 11 (1 FAILED) (skipped 6) (0.153 secs / 0.127 secs)
```

Abriu o navegador mostrando o erro encontrado



Corrigindo o valor para “50”:

```
it('Teste de Servico de Multiplicacao :', () => {  
  expect(50).toBe(service.multiply(5, 10));  
});
```

