Resumo TypeScript - Angular 4 5 6

Seção 2

Vídeo 2

Angular é um framework utilizado para construir aplicações web baseadas em HTML 5, CSS e JavaScript, organizando estas tecnologias e entregando uma aplicação web capaz de ser executada no browser sendo possível consumir um ou vários serviços disponibilizados por um servidor. Em geral este tipo de aplicação é chamada de Client-Side.

Seção 3

Vídeo 3

Angular pode ser usado com 4 linguagens: ECMAScript 5 (JavaScript atual), Ecmascript 2015, TypeScript e DART.

ES5 ==> não precisa de compilador, não terá módulos compatíveis com Ecmascript 2015 e não terá tipos.

Ecmascript 2015 ==> terá classes, terá módulos, precisará de um compilador como exemplo Babel e não terá tipos.

TypeScript ==> extends ECMAScript 2015, terá classes, terá módulos, decorators e um compilador que ajuda a detectar erros ainda em tempo de compilação.

DART ==> outra linguagem que pode ser utilizada para desenvolver em Angular 2.

O Angular é escrito em TypeScript.

==========

Seção 3

Vídeo 4

NodeJS ==> plataforma criada a partir do Google Chrome bastante utilizada em aplicações do lado do servidor (Server-Side). Com tudo, através desta plataforma gerenciaremos as dependências de uma aplicação em Angular com a utilização da ferramenta npm.

NodeJS ==> é um interpretador de código JavaScript com o código aberto, focado em migrar o JavaScript do lado do cliente para servidores.

npm (gerenciador do pacote do NodeJS) que pode ser usado no gerenciamento das dependências de uma aplicação onde uma delas poderá ser o gerenciamento dos pacotes do Angular.

npm ==> "node package manager"

Como instalar o NodeJS:

1) Baixe a plataforma em: http://nodejs.org/en/
2) Entre no terminal e verifique a versão do NodeJS: node -v

Como instalar o Angular-Cli:

1) O que é o Angular-Cli: ferramenta de linha de comando criada a partir do Angular 2 para ajudar a criar desde de um projeto inicial em Angular, até mesmo preparar nossa aplicação para produção, ou seja, fazer o build da nossa aplicação além de permitir criar nossos componentes, diretivas, pipes e módulos.

2) Como instalar o angular-cli. Entre no terminal do Node e faça: npm install -g @angular/cli
2.1) -g ==> significa que o acesso à ferramenta será global e não apenas na pasta onde está sendo feita a instalação.
2.2) cli ==> command line interface (interface de linha de comando) – ferramenta para criar aplicações angular
3) Verificar a versão da ferramenta Angular-Cli: ng -v
4) A versão do Angular dever ser verificada no arquivo package.json em “dependencies:{"@angular/common”...}”
5) Instalando uma versão específica do Angular-Cli: npm install -g @angular/cli@1.7.3 //neste caso versão 1.7.3
6) Instalando uma versão específica do Angular-Cli: npm install -g @angular/cli@versao_do_angular //[EXTRA]
7) Desinstalando Angular-Cli: npm uninstall -g @angular/cli //[EXTRA]
8) Limpando o cache após uma desinstalação: npm cache clean //[EXTRA]
9) Alterando a porta do servidor interno do webpack: ng serve - -port numero_da_porta_desejada //[EXTRA]
10) Carregando a aplicação e abrindo o browser: ng serve -o //[EXTRA]
==========

Seção 3

Vídeo 6

TypeScript é uma linguagem de programação que estende JavaScript.
O navegador não entende TypeScript, logo será necessário fazer uma espécie de compilação chamada de Transpile.

Trinspile:

TypeScript (*.ts) ==> **Transpile ("compilação")** ==> JavaScript(*.js)

Exibição de um arquivo TypeScript até o browser:

TypeScript (*.ts) ==> **Transpile ("compilação")** ==> JavaScript(*.js) ==> **import ("feito pelo navegador")** ==> html

Tipos em TypeScript: string - number - boolean - any (quer dizer que a variável aceitará qualquer tipo)

Exemplo:

```
let team: string = "Flamengo"

let nome: string
nome = "Caio"

let carro: any
carro = “Ferrari”  //a variável carro foi inicializada inicialmente como “string” a seguir a mesma variável recebe
carro = 1000       //um “number” e o compilador não detecta um erro, pois a variável foi tipada como “any”

let meuArray: number[] = [1,2,3]

let seuArray: Array<number> = [1,2,3]

let nossoArray = [1,2,3] //inferência de tipo
```

EXTRA:

```
let nome : string   //undefined

let idade : number  //undefined

let sobrenome : string  = ‘Ferreira’

let total : number = 200
```

==========

Seção 1

Vídeo 7

Como compilar um arquivo TypeScript (*.ts) através do terminal:

1) Instalar o compilador TypeScript através do terminal NodeJS: npm install -g typeScript
1.1) -g ==> significa que o acesso ao compilardor será global e não apenas na pasta onde está sendo feita a instalação
2) Aponte para a pasta onde esta o seu projeto.
3) Crie um arquivo de configuração: tsconfig.json
4) O arquivo tsconfig.json pode ser criado na raiz da aplicação com o comando: tsc - -init

tsconfig.json

Conteúdo do tsconfig.json configuração padrão.
1) sempre que for salvo um arquivo em *.ts será salvo um arquivo *.js no mesmo diretório do arquivo *.ts

```
{

}
```

Conteúdo do tsconfig.json configuração customizada.

```
{
        "compilerOnSave": true /** para compilação imediata quando usar a IDE Atom **/
}
```

4) verificar versão do compilador TypeScript: tsc -v
EXTRA: O comando "*tsc*" é um acrônimo para TypeScript Compiler.
5) compilando um arquivo: tsc nome_do_arquivo.ts
5.1) outra solução seria deixar o compilador rodando em background para a compilação automática então faça: tsc -w
Desta forma não será necessário usar o comando de compilação para cada arquivo: tsc nome_do_arquivo.ts

6) Configuração do arquivo tsconfig.json customizada

```
{
 "compilerOptions":{
 "noEmitOnError": true, /** emetir erro caso seja atribuído o tipo errado à variavel **/
 "outDir": "dist" /** nesta pasta estarão armazenados automaticamente os **/
                  /** arquivos *.js após a compilação dos arquivos *.ts **/
        }
}
```

Obs.: "compileOnSave": true, /** compila os arquivos de ts para js ao salvar o arquivo ts **/

Obs.: Quando o novo diretório for configurado para armazenar os arquivos *.js, como exemplo teríamos "outDir": "dist" configurado no arquivo tsconfig.json, será necessário reiniciar o Compilador TypeScript, o comando a seguir dever ser dado no diretório onde esta localizado o(s) arquivo(s) *.ts : **tsc -w**
A execução dos arquivos JavaScript deverá apontar para pasta onde está armanzenado arquivo *.js
Exemplo: **node dist/arquivo_javascript**

Como executar os arquivos na Linguagem TypeScript

Perceba que a execução ocorre sobre os arquivos *.js e não sobre os arquivos do TypeScript (*.ts)

1) no terminal faça: node nome_do_arquivo //não precisa da extensão .js

==========

Seção 1

Vídeo 8 e 9

Funções em TypeScript

```
function inc (speed: number, value: number = 3) : number {
	return speed + inc   /** se value não receber um valor, o default será 3 **/
}

console.log(`inc (5,7) = ${inc(5,7)}` ) //12 – para utilizarmos expressões devemos usar `xxx`
console.log(`inc (5) = ${inc(5)}` ) //8

let isEnoughToBeatMF = function (parsecs: number) : boolean {
	return parsecs < 12
}

let distance = 14 //passagem de valor por inferência de tipo neste caso number

/** `xxxx` ==> “Template String” permite usar expressões como este a seguir ==> ${distance} **/
console.log(`Is ${distance} parsecs enough to beat Millennium Falcon? ${isEnoughToBeatMF(distance) ? 'YES' : 'NO'}`)

/** Exemplo abaixo de Arrow Function **/
/** As funções Arrow Function não utilizam as palavras reservadas function e return **/

let tieFighters = ships.filter(function(ship)){
	return ship.type === ‘TieFighter’
}

let ties = ships.filter (ship => ship.type === ‘TieFighter’)

let call: (name: string) => void //declarando que a variável será uma função, os argumentos e o retorno
call = name => console.log(“Do you copy,” + name + “?”) //atribuindo o valor a variável que neste caso
                                                          //será a própria função.
call(“R2”) //realizando a chamada

/** lado direito declaração |   lado esquerdo implementação **/
let call = (name: string) => console.log(`Do you copy, ${name}?`)

/** função com parâmetro opcional ==> '?' **/
function soma (speed: number, sum?: number) : number {
	let i = sum || 9
	return speed + sum   /** se sum não receber um valor o default será 9 **/
}

console.log(`soma (5,9) = ${soma(5,9)}` ) //14
console.log(`soma (5,9) = ${soma(5)}` ) //14
console.log(`soma (7) = ${soma(7)}` ) //16
```

Função sem Parâmetros Rest

```
function add (jedis: number[]) : number {
	return jedis.reduce((a+b) => a+b,0)
}

console.log(`add([1,2,4]) = ${add([1,2,4])}`) //7
```

Função com Parâmetros Rest

```
function adicionar (...jedis: number[]) : number { //perceba os "..." no parâmetro da função
	return jedis.reduce((a+b) => a+b,0)
}

console.log(`add([1,2,4]) = ${add(1,2,4)}`) //aqui os parâmetros foram passados por vírgula o que
                                            //torna mais fácil a passagem de parâmetros
```

==========

Seção 1

Vídeo 11

As interfaces em TypeScript(.ts) não possuem um equivalente no arquivo JavaScript (.js) quando é feito o Transpile.

Exemplo de Interface em TypeScript

```
interface Containership {

cargoContainers : number

}
```

Veja que o atributo "cargoContainers" da interface dever ser implementado na classe que implementa a interface.

```
class Spacecraft() implements Containership{

	propulsor: string

	constructor (propulsor: string){
		this.propulsor = propulsor
	}

	constructor(public propulsor: string){
		propulsor = "Hyperdriver"
		this.cargoContainers = 4 //implementação do atributo  cargoContainers
	}

}
```

Exemplo de Interface em TypeScript com Atributo Opcional

```
interface Containership {

cargoContainers? : number

}
```

Veja que o atributo “cargoContainers” da interface não tem obrigação de ser implementado na classe que implementa a interface, pois o atributo possui o caracter especial “?”. Esta idéia também pode valer para métodos.

```
class Spacecraft() implements Containership{

	propulsor: string

	//constructor (public propulsor: string){} //forma customizada de implementar o construtor

	constructor(public propulsor: string){
		propulsor = “Hyperdriver”
		//this.cargoContainers = 4 //não temos obrigação de implementar o atributo cargoContainers
	}

}
```

==========

Seção 1

Vídeo 12

módulo: é uma unidade que pode conter classes, interfaces, funções e variáveis com a finalidade de isolar o conteúdo de um arquivo do restante da aplicação evitando assim que algo seja sobrescrito como exemplo: declarações de variáveis, métodos entre outros de mesmo nome em outros arquivos e isto pode ser feito com as palavras chaves **export** e **import**. É importante ressaltar que há diferença entre um Módulo em Angular e um Módulo em EcmaScript 2015.
O Módulo em EcmaScript 2015, TypeScript extends EcmaScript 2015, é uma forma de isolar funções, variáveis, classes e interfaces para exportação enquanto um Módulo em Angular é responsável por saber quais componentes, serviços, pipes e diretivas fazem parte da aplicação; como exemplo podemos citar o arquivo app.modulo.ts que contém o módulo AppModule. Toda aplicação em Angular terá pelo menos um módulo (módulo raiz AppModule contido no arquivo app.module.ts) e um componente ou função.

Como declarar um Módulo

Utilize o atributo: export

==========

```
interface Containership {

	cargoContainers: number

}

 export {Containership}
```

==========

```
class Spacecraft {

	//propulsor: string
	//constructor (propulsor: string){
```

```
        //          this.propulsor = propulsor
        //}

        //o conteúdo acima comentado equivale ao construtor abaixo.

        constructor (public propulsor: string){} //public ou private

        jumplntoHyperspace1(){
                console.log("====> Entering hyperspace with "+this.propulsor)
        }

        jumplntoHyperspace2(){
                console.log(">>>> Entering hyperspace with "+this.propulsor+" >>>>>>>>" )
        }

}

 export {Spacecraft}
```

==========

Seção 1

Vídeo 14

Definição de Tipo

É uma forma de definir os valores que um dado pode assumir e as operações que podem ser efetuadas sobre o dado. A definição de tipos atua sobre métodos, variáveis, funções entre outros.

Como Instalar a Definição de Tipos em TypeScript (1 a 3)

1)

```
> npm init -f                    // a instalação deve ser feita no mesmo diretório do arquivo tsconfig.json
                                 // desta forma será criado um arquivo de nome package.json.
                                 // -f ==> instalar sem fazer peguntas pegando a configuração padrão
```

Instalando uma biblioteca como exemplo "lodash"

2)

```
> npm install - -save  nome_do_pacote_da_biblioteca@numero_da_versão
> npm install - -save  lodash@4.14      // a instalação deve ser feita no mesmo diretório do arquivo tsconfig.json
                                        // - -save ==> significa que a biblioteca lodash será necessária tanto
                                        // em desenvolvimento(compilação) como em runtime(execução).
```

Instalando a Definição de Tipo para a biblioteca "lodash"

3)

```
> npm install - -save-dev @types/lodash@4.14     // a instalação deve ser feita no mesmo diretório do arquivo tsconfig.json
                                                 // - -save-dev ==> significa que a biblioteca lodash será necessária
                                                 // apenas em desenvolvimento, mas não em runtime.
```

* A instalação da definição de tipo permitirá fazer "auto-complete" de métodos além da verificação dos tipos dos parâmetros (string – number – boolean - any) neste caso da biblioteca "lodash".
* O resultado destes comandos será a criação do arquivo "packages.json".
* "package.json" ==> arquivo que conterá as configurações básicas da aplicação além de suas dependências.
* No final dos comandos acima serão criados os arquivos: package.json e package-lock.json além da pasta node_modules a qual conterá a biblioteca lodash.

Segue um exemplo de package.json.

package.json

```
{
  "name": "spaceproject",
  "version": "1.0.0",
  "description": "",
  "main": "Main.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },

  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "lodash": "^4.14.2"
  },
  "devDependencies": {
    "@types/lodash": "^4.14.116"
  }
}
```

<u>Como Importar uma Biblioteca</u>

Como exemplo de biblioteca: lodash

```
import {Containership} from "./craft/Containership"
import {Spacecraft} from "./craft/Spacecraft"

import * as _ from 'lodash'  //importando a biblioteca lodash
 // _ ==> é a variável padrão da biblioteca lodash assim como o $ ==>  é a variável padrão do JQuery

class MillenniumFalcon extends Spacecraft implements Containership {

        cargoContainers: number

        constructor (){
                super("Hyperdrive")
                this.cargoContainers = 4

        }

        jumplntoHyperspace(){

                        if(Math.random() >= 0.5){
                                super.jumplntoHyperspace1()
                        }else{
                                console.log("Failed")
                        }

        }


}

console.log(_.pad("Este foi um exemplo de importação de biblioteca.",40,"="))

let falcon = new MillenniumFalcon()
   falcon.jumplntoHyperspace()
```

==========

Seção 4

Vídeo 16

Gerando a Primeira Aplicação em Angular

O comando a baixo criará uma aplicação além de baixar todas as dependências necessárias.

> ng new NOME_DA_APLICAÇÃO - - prefix=NOME_DO_SEU_PREFIXO

//ng new jedi-academy - - prefix=jad
//jad ==> prefixo usado na chamada dos componentes
//Quando uma aplicação é gerada temos como resultado dentro desta aplicação alguns arquivos e pastas tais como: //package.json, angular.json(ou angular-cli.json), tsconfig.json, package-lock.json, tsconfig.json, node_modules, src e e2e.

Carregando a Aplicação

> cd NOME_DA_APLICAÇÃO
> ng serve //npm start
No navegador faça: localhost:4200 ==> sua aplicação deve responder a este endereço.

Principais Arquivos de uma Aplicação Angular

package.json : arquivo de configuração que contém todas as dependências da aplicação, seja dependências em runtime ou compilação.

```
{
  "name": "jedi-academy", //nome da aplicação gerada
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {  //dependências principais do Angular
    "@angular/animations": "^6.1.0",
    "@angular/common": "^6.1.0",
    "@angular/compiler": "^6.1.0",
    "@angular/core": "^6.1.0",
    "@angular/forms": "^6.1.0",
    "@angular/http": "^6.1.0",
    "@angular/platform-browser": "^6.1.0",
    "@angular/platform-browser-dynamic": "^6.1.0",
    "@angular/router": "^6.1.0",
    "core-js": "^2.5.4",
    "rxjs": "~6.2.0",
    "zone.js": "~0.8.26"
  },
  "devDependencies": { //dependências de desenvolvimento, ou seja,  serão necessárias apenas em compilação
    "@angular-devkit/build-angular": "~0.8.0",
    "@angular/cli": "~6.2.2",
    "@angular/compiler-cli": "^6.1.0",
    "@angular/language-service": "^6.1.0",
    "@types/jasmine": "~2.8.8",
    "@types/jasminewd2": "~2.0.3",
    "@types/node": "~8.9.4",
    "codelyzer": "~4.3.0",
    "jasmine-core": "~2.99.1",
    "jasmine-spec-reporter": "~4.2.1",
```

```
    "karma": "~3.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~2.0.1",
    "karma-jasmine": "~1.1.2",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.4.0",
    "ts-node": "~7.0.0",
    "tslint": "~5.11.0",
    "typescript": "~2.9.2"
  }
}
```

==========

angular-cli.json (ou angular.json) : arquivo de configuração do angular-cli onde se pode inserir styles e scripts

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "jedi-academy": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "jad",   //configuração do prefixo de nome “jad” que será utilizada na chamada dos nossos componentes
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/jedi-academy",
            "index": "src/index.html",
            "main": "src/main.ts",      //arquivo que “CARREGA” a aplicação a partir do módulo principal
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json", //arquivo com configuração do compilador do TypeScript
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [ //aqui podemos listar outros arquivos de estilo css.
              "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
                  "with": "src/environments/environment.prod.ts"
                }
              ],
              "optimization": true,
              "outputHashing": "all",
              "sourceMap": false,
              "extractCss": true,
              "namedChunks": false,
              "aot": true,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true
```

```
        }
      }
    },
    "serve": {
      "builder": "@angular-devkit/build-angular:dev-server",
      "options": {
        "browserTarget": "jedi-academy:build"
      },
      "configurations": {
        "production": {
          "browserTarget": "jedi-academy:build:production"
        }
      }
    },
    "extract-i18n": {
      "builder": "@angular-devkit/build-angular:extract-i18n",
      "options": {
        "browserTarget": "jedi-academy:build"
      }
    },
    "test": {
      "builder": "@angular-devkit/build-angular:karma",
      "options": {
        "main": "src/test.ts",
        "polyfills": "src/polyfills.ts",
        "tsConfig": "src/tsconfig.spec.json",
        "karmaConfig": "src/karma.conf.js",
        "styles": [
          "src/styles.css"
        ],
        "scripts": [],
        "assets": [
          "src/favicon.ico",
          "src/assets"
        ]
      }
    },
    "lint": {
      "builder": "@angular-devkit/build-angular:tslint",
      "options": {
        "tsConfig": [
          "src/tsconfig.app.json",
          "src/tsconfig.spec.json"
        ],
        "exclude": [
          "**/node_modules/**"
        ]
      }
    }
  }
},
"jedi-academy-e2e": {
  "root": "e2e/",
  "projectType": "application",
  "architect": {
    "e2e": {
      "builder": "@angular-devkit/build-angular:protractor",
      "options": {
        "protractorConfig": "e2e/protractor.conf.js",
        "devServerTarget": "jedi-academy:serve"
      },
      "configurations": {
        "production": {
```

```
            "devServerTarget": "jedi-academy:serve:production"
          }
        }
      },
      "lint": {
        "builder": "@angular-devkit/build-angular:tslint",
        "options": {
          "tsConfig": "e2e/tsconfig.e2e.json",
          "exclude": [
            "**/node_modules/**"
          ]
        }
      }
    }
  }
},
"defaultProject": "jedi-academy"
}
```

WebPack: biblioteca em JavaScript responsável por criar arquivos específicos na nossa aplicação os quais são chamados de bundles além de compilá-los.
Alguns exemplos de bundles:

polyfills.bundle.js => permite adicionar scripts que aumentam a compatibilidade com navegadores antigos.
main.bundle.js => arquivo que “CARREGA” a aplicação a partir do módulo principal (app.module.ts).
styles.bundle.js => arquivo que reunirá todos os arquivos css da sua aplicação.

Outros arquivos importantes:

styles.css : neste arquivo você poderá colocar todos os estilos globais da sua aplicação. Perceba que este arquivo esta listado no arquivo angular.js da sua aplicação.

polyfills.ts : arquivo que fornecerá suporte a funcionalidades novas aos navegadores antigos. Será neste arquivo que também implementaremos internacionalização de idiomas ou localização caso seja necessário.

==========

Seção 4

Vídeo 18

Explicação sobre a inicialização de uma Aplicação com Angular

Quando executamos “ng serve” na realidade estamos disparando um servidor interno do webpack o qual é utilizado apenas para desenvolvimento, a partir deste ponto o webpack verificará os arquivos da nossa aplicação começando pelo main.ts e todos os demais arquivos olhando também os imports, dependências e scripts fazendo assim um mapeamento da nossa aplicação separando estas informações em vários bundles. Uma vez estes bundles criados ao acessarmos o index.html através do navegador as referências aos bundles serão injetadas no final do arquivo index.html automaticamente e os scripts destes bundles também serão executados também automaticamente. É importante frisar que o main.bundle.js faz o bootstrap da aplicação.

main.ts : arquivo responsável por fazer bootstrap da aplicação

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
```

```
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}


platformBrowserDynamic().bootstrapModule(AppModule).catch(err => console.error(err)); //linha responsável por
INICIALIZAR nossa aplicação carregando o módulo principal  AppModule (app.module.ts)
```

==========

app.module.ts : arquivo gerado automaticamente pelo angular-cli. Este é um arquivo agrupador de componentes da sua aplicação.

app.module.ts

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

import { AppComponent } from './app.component';

@NgModule({  // @NgModule ==> função que serve para aplicar metadados em uma classe, atributo, método
             // ou em um argumento de um método. Este tipo de função também é chamado de “decorator”.
             // Aqui no caso estamos aplicando metadados em uma classe.

  declarations: [ // declara todos os componentes que temos dentro da aplicação
    AppComponent
  ],
  imports: [  // nossas dependências
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent] //o bootstrap indica qual o componente no array de declarations[] responsável
                            //por fazer a inicialização da aplicação, neste caso o componente será AppComponent.

})
export class AppModule { }
```

==========

app.component.ts : componente gerado automaticamente pelo angular-cli. Toda aplicação em Angular terá um módulo raiz tal qual app.modulo.ts e um componente de bootstrap como AppComponent.

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

@Component({ // @Component => decorator específico para componentes também e um exemplo de metadado
  selector: 'jad-root', //atributo que informa o componente raiz da nossa aplicação. => atributo obrigatório
  templateUrl: './app.component.html', //exibe a cara da página da aplicação. => atributo obrigatório
  styleUrls: ['./app.component.css'] //estilo de personalização => atributo não obrigatório

 //EXTRA: preserveWhitespaces: true //preserva os espaços em branco no HTML
})
```

```
export class AppComponent {
  title = 'jedi-academy';
}
```

==========

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JediAcademy</title>
  <base href="/"> <!- - tag responsável por fazer o sistema de rotas funcionar a partir das rotas - ->

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<jad-root>Loading...</jad-root> //este componente será substituído pelo valor do select contido no módulo principal
</body>
</html>
```

EXTRA:

Erro quando não se declara a tag base:

Angular reclama da ausência da tag base:

```
EXCEPTION: No base href set. Please provide a value for the
APP_BASE_HREF token or add a base element to the document.
```

EXTRA:

A tag base é importante quando usamos o sistema de rotas do Angular.

```
<!doctype html>
<html>
    <head>
        <base href="/"> <!-- novidade aqui -->
        <title>Alurapic</title>
        <meta charset="UTF-8">
       <!-- código posterior omitido -->
```

==========

app.component.html

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

@Component({
  selector: 'jad-root',  //selecionando a tag com o atributo selector
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'jedi-academy'; //
}
```

==========

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}! // {{}}=> expressão onde “title” será substituído por um valor
                              // contido em um componente em app.component.html.

                              //{{}} ==> este tipo de expressão é chamado de interpolação
  </h1>
 </div>
```

==========

Caminho no qual nossa Aplicação faz pelo Bootstrap

Bootstrap carrega o arquivo main.ts que por sua vez carrega o módulo principal (app.modulo.ts) o qual informará qual será o componente de Bootstrap em seguida o Angular pegará o template do componente e colocará na página.

O responsável por fazer o Bootstrap é um conjunto de bibliotecas em JavaScript chamado Webpack.

main.ts ==> página html <== app.component.ts <== app.component.html

---

Seção 4

Vídeo 19

Componentes em Angular

Componentes são pequenas partes independentes e reusáveis, além disso em Angular são “classes” que tem um determinado “ciclo de vida”, possuem um “template” para definir uma aparência e um “selector” que é a tag para ser usada em outras partes da aplicação.

Em Angular uma página pode ser composta por partes menores que podem funcionar de forma independente ou comunicando-se através de eventos. Neste framework os componentes são dispostos em forma de árvore sempre tendo na raiz um componente pai seguido de seus filhos.

Exemplo de component:

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

@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})

export class AppComponent {
	title = 'app';
}
```

Exemplo de component:

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

@Component({
selector: 'app-first',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})

export class MyfirstComponent {
	title = string = ‘My Start Wars Component!’;
	user = {name: ‘Luke Skywalker’}
}
```

==========

Para utilizarmos o decorator @Component devemos em primeiro lugar importar o modulo '@angular/core' .

selector: que define qual será a tag para o componente que foi criado.
templateUrl: é o caminho para o template do componente, podemos substituir esta propriedade por "template", neste caso definimos um template inline, incluindo por exemplo o nosso html do componente:

```
template:'<h1>Template do componente</h1>'
```

<u>Criando um Componente com a Ferramenta Angular/Cli</u>

cd ENTRE_NO_PROJETO_DESEJADO

ng generate component NOME_DO_SEU_COMPONENTE - -spec=false

EXTRA: Em alguns projetos quando houver mais de um arquivo module pode ser necessário especificar o modulo no qual o componente sera registrado para este fim utilize o comando a baixo.

EXTRA: Todo componente deve ser registrado em um módulo e somente em um único módulo.

> ng generate component novocomponente - -module=nome.do.arquivo.module.ts

Obs.: Para as versões mais recentes do Angular/Cli use o caminho completo até a raiz para criar o componente.

ng generate component CAMINHO_A_PARTIR_DA_RAIZ_ATE_A_PASTA_ONDE_SE_QUER_CRIAR_O_COMPONENTE/seucomponente

ng generate component header - -spec=false // - -spec=false ==> não criará arquivos de teste,

// header ==> nome do componente

//Após a geração do componente conforme os comandos acima três arquivos serão gerados:

src/app/header/header.component.html => template do componente

src/app/header/header.component.css => estilo do componente

header.component.ts => arquivo que conterá a classe do componente

Sempre que um componente for criado o mesmo deverá ser listado em um módulo. Veja abaixo:

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

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

    constructor() { }

    ngOnInit() {//método chamado na inicialização do componente pelo menos uma vez durante o ciclo de vida do mesmo.
    }

}
```

==========

Seção 4

Vídeo 21

Property Binding

É uma maneira de conectar uma propriedade de um elemento seja ele HTML ou de um componente customizado a uma expressão Angular. O property binding pode ser aplicado a qualquer elemento do DOM. A conexão sempre se dará do componente para o elemento html ficando desta forma conhecida como ONE-WAY BINDING.

Sintaxe de Property Binding ==> **[ ]**

EXTRA:

Obs.: O Property Binding permite conectar uma chamada de método, uma variável ou mesmo um atributo de um componente com uma propriedade de um template.

Veja: <img src= "{{ urlImage }}" /> //Interpolação – esta também pode ser considerada como uma forma de Property Binding

<img [src]="urlImage" /> //Property Binding

<img bind-src="urlImage" /> //Property Binding

<img [src]="acessarUrlImage()" /> //Property Binding

Obs.: O Framework Angular identifica um Property Binding através dos colchetes como visto acima.

//No componente

user = { name: ‘Luke Skywalker’}

//No template do componente

< input type= “text” [value] = “user.name” > // o atributo name de user terá o seu valor atribuído ao value de input.

// Sempre que o valor de name mudar o valor de input mudará.

Propriedades de DOM

=> Propriedades HTML

```
<form>
        <input type="text" name="username" value= "Joe"/>
</form>
```

=> Propriedades de DOM: estes objetos do DOM serão sempre acionados em tempo de execução

HtmlFormElement => HtmlInputElement ´=> type="text" name="username" value= "Joe"

Algumas propriedade dos elementos de um DOM: type, name, value, hidden, textContent …

Seção 4

Vídeo 23

Como passar dados para um Componente Angular

HeaderComponent.ts

```
import {Component, Input} from ´@angular/core´
@Component({
selector: ´mt-header´,
template: ´<h1>{{title}}</h1>´
})
export class HeaderComponent {
@Input() title: string
}
```

Passando valores...

1) <mt-header title=”Minha App”></mt-header>

<!-- template interpolation -->

2) <mt-header title=”{{isJedi ? ‘Jedi’ : ‘Sith’}}”></mt-header>

<!-- property binding -->

HeaderComponent.ts

import {Component, **Input**} from ´@angular/core´

@Component({

selector: ´mt-header´,

template: ´<h1>{{title}}</h1>´

})

export class HeaderComponent {

@Input(‘value’) title: string

}

<!-- usando o nome definido em @Input -->

3) <mt-header **value**=”Título”></mt-header>

Seção 4

Vídeo 24

Observação: sempre que você tiver que passar um valor que não seja uma string use a sintaxe de property binding.

1) <input type="text" [value]="user.name"> // **[ ] ==>** property binding

2) <jad-student name="Luke" **[**isJedi**]**="false"></jad-student> // **[ ] ==>** property binding

3) <jad-student [student]="**luke**"></jad-student> // **[ ]** ==> property binding

//**luke** = {name: ‘Caio’, isStudent: true, profile: ‘A good student.’}

Observação: quando utilizamos o atributo hidden temos que ficar cientes que o controle da visibildade de um elemento será feita via css o que não é uma boa idéia pois há a possibilidade do estilo css ser sobrescrito; uma forma de uso mais segura seria o emprego de diretivas como exemplo ***ngIF**.

app.component.html

<jad-student2 name="Leia" **[**isJedi**]**="false"></jad-student2> // **[ ] ==>** property binding

Seção 4

Vídeo 25

Diretivas

Diretivas são componentes sem templates, desta forma os componentes herdam tudo de Diretivas.

Tipos de Diretivas no Angular

Componentes: são diretivas com templates.

Estruturais: mudam o template e a estrutura do DOM.

Atributos: onde você associa um atributo ao elemento do DOM.

Exemplos de Diretivas Estruturais:

*ngIf

*ngFor

*ngSwitch

*ngSwitchWhen

*ngSwitchDefault

```
<!-- ng if -->
<input type="text" [value]="user.name">
<div *ngIf="user.isJedi">
  location of the jedi temple
</div>
```

A forma acima você dispensa o uso do elemento <template>

```
<!-- ng if -->
<input type="text" [value]="user.name">
<template [ngIf]="user.isJedi">
<div>
  location of the jedi temple
```

```
</div>
</template>

<!-- ng for -->
<ul>
<li *ngFor = "let user of users">{{user.name}}</li> //a variável user (let user) representa cada item da coleção de users
</ul>

<!-- ng for --><!-- usando um contador -->
<ul>
  <li *ngFor = "let user of users; let i=index">
    {{i+1}} - {{user.name}}
  </li>
</ul>
```

EXTRA:

```
//profile: string = "user"; //root – user - other
< div [ngSwitch]="profile"> // ngSwitch ==> avalia uma condição
        <p *ngSwitchCase="root">You can read & write</p> // ngSwitchCase => renderiza
        <p *ngSwitchCase="user">You can read</p>
        <p *ngSwitchDefault>Go back, please!</p>
</div>
```

EXTRA:

```
//permission: number = 2;  //1 - 2 - 3
< div [ngSwitch]="permission"> // ngSwitch ==> avalia uma condição
        <p *ngSwitchCase="'1'">You can read & write</p> // ngSwitchCase => renderiza
        <p *ngSwitchCase="'2'">You can read</p>
        <p *ngSwitchDefault>Go back, please!</p>
</div>
```

Seção 4

Vídeo 27

Extra:

Diferença semântica entre null e undefined

Há uma diferença semântica entre null e undefined: o valor null é utilizado para indicar a ausência de um objeto, enquanto undefined indica a ausência de um valor qualquer. A especificação explicita isso quando define os valores undefined e null.

Valor undefined: valor primitivo utilizado quando uma variável não teve o seu valor atribuído.
Valor null: valor primitivo que representa a ausência intencional de um valor de objeto.

Operador de Navegação Segura

O operador de navegação segura utiliza o caracter '**?**' para evitar que o compilador leia determinado atributo de um componente caso o atributo não seja passado na tag.

```
//student.component.ts

export interface Student{

        name: string

        isJedi: boolean

        temple?: string

}

//student.component.html

<div>  //<!-- <div *ngIf="student" -->

        Student: {{student?.name}} //significa que até o ponto '?' se a variável for undefined

                                   //a expressão não precisará ser avaliada, evitando assim erros.

        <div *ngIf="student?.isJedi">

                Jedi Temple >>> {{student?.temple}}

        </div>

</div>

<br/>
```

Seção 4

Vídeo 27

Eventos

Em eventos usamos parênteses ao redor de um evento para ligá-lo a um método de um componente,

sendo assim sempre que uma ação ocorrer o método será chamado.

1)

```
import { Component } from '@angular/core'

@Component ({

   selector: 'mt-clickable',
```

```
    template: ‘<button (click) = “clicked()”>Click!</button>’
})
export class ClickableComponent {
      clicked(): void {
                console.log(‘Button clicked!’)
        }
}
```

2)

```
<! no template do componente -->
<button (click)="clicked($event)">Click me!</button>
clicked(event): void{
 console.log(`Button clicked: ${event}`)
}
```

3) Nesta situação será disparada uma ação quando a “Tecla Espaço” for pressionada

```
<! no template do componente -->
<input (keydown.space)="keyDown($event)">
keyDown(event): void {
 console.log(`Key down: ${event}`)
}
```

<u>Emetindo Eventos de um Componente</u>

Recurso que permite emitir eventos personalizados.

1)

```
import { Component, Ouput, EventEmitter } from ‘@angular/core’
@Component ({
   selector: ‘mt-clickable’,
    template: ‘<button (click) = “clicked()”>Click!</button>’
})
export class ClickableComponent {
     @Output() myEvent = new EventEmitter()
      clicked(): void {
```

```
            this.myEvent.emit()
        }
}
```

2)

```
<! no template que usa o componente -->
<mt-clickable (myEvent)="willBeCalled()"></mt-clickable>

 willBeCalled(): void {
 console.log('Event from clickable')
}
```

Variável de Template

Uma variável de template permite pegar uma referência de um elemento do DOM ou a um componente do template.

#description => variável de template que permite passar uma referência de um elemento neste caso <textarea> dentro do template, realizar chamada de atributos ou mesmo uma chamada de um método. Veja abaixo:

discipulo.component.html

```
<div *ngIf="disc">
        Discipulo: <a href="#" (click)="clicked()">{{disc?.nome}}</a>
        Idade: {{disc?.idade}}
        <div *ngIf="disc?.isBrasileiro">
                Este discipulo é brasileiro.<button (click)="description.focus()">Focus!</button><br/>
                <textarea #description></textarea>
        </div>
        <div *ngIf="!disc?.isBrasileiro">
                Este discipulo é estrangeiro.
        </div>
</div>
//Ao clicar no botão o componente <textarea> receberá o focu.
```

Seção 5

Vídeo 31

Lidando com Erros

Durante o desenvolvimento de uma tela poderemos obter erros em tempo de compilação e execução. Para erros em tempo de compilação fique atento com as mensagens exibidas pelo compilador e para erros em tempo de execução verifique as mensagens exibidas na ferramente Developer tools do browser.

Seção 5

Vídeo 32

Instalando dependências de um projeto

> npm install //com este comando o projeto baixará as dependências criando a pasta “node_modules” caso a mesma não exista.

Onde ver a versão do Angular

Entre na aplicação e abra o arquivo package.json e veja a seção “dependencies: { ….}”.

Veja a seção abaixo neste caso estamos trabalhando com Angular 4.

package.json

```
{
  "name": "meat",
  "version": "1.0.0",
  "license": "MIT",
  "angular-cli": {},
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "lint": "tslint \"src/**/*.ts\" --project src/tsconfig.json --type-check && tslint \"e2e/**/*.ts\" --project e2e/tsconfig.json --type-check",
    "test": "ng test",
    "pree2e": "webdriver-manager update --standalone false --gecko false",
    "e2e": "protractor"
  },
  "private": true,

"dependencies": {
    "@angular/common": "4.0.0",
    "@angular/compiler": "4.0.0",
    "@angular/core": "4.0.0",
    "@angular/forms": "4.0.0",
    "@angular/http": "4.0.0",
```

```
    "@angular/platform-browser": "4.0.0",
    "@angular/platform-browser-dynamic": "4.0.0",
    "@angular/router": "4.0.0",
    "admin-lte": "2.3.11",
    "core-js": "2.4.1",
    "font-awesome": "4.7.0",
    "intl": "1.2.5",
    "jquery": "3.1.1",
    "reflect-metadata": "0.1.9",
    "rxjs": "5.1.0",
    "ts-helpers": "1.1.2",
    "zone.js": "0.8.4"
  },
  "devDependencies": {
    "@angular/cli": "1.2.7",  //versao da Ferramenta Angular Cli
    "@angular/compiler-cli": "4.0.0",
    "@types/jasmine": "2.5.38",
    "@types/node": "7.0.5",
    "codelyzer": "2.0.0",
    "jasmine-core": "2.5.2",
    "jasmine-spec-reporter": "3.2.0",
    "karma": "1.4.1",
    "karma-chrome-launcher": "2.0.0",
    "karma-cli": "1.0.1",
    "karma-jasmine": "1.1.0",
    "karma-remap-istanbul": "0.2.2",
    "protractor": "5.1.0",
    "ts-node": "2.1.0",
    "tslint": "4.5.0",
    "typescript": "2.2.0",
    "webdriver-manager": "10.2.5"
  }
```

Seção 5

Vídeo 33

Onde estão as configurações de styles e scripts de uma aplicação em Angular

Veja o arquivo angular-cli.json.

É importante ressaltar que o webpack incluirá estes arquivos de styles e scripts na aplicação na ordem definida abaixo.

angular-cli.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "project": {
    "version": "1.0.0",
    "name": "meat"
  },
  "apps": [
    {
      "root": "src",
      "outDir": "dist",
      "assets": [
        "assets",
        "favicon.ico"
      ],
      "index": "index.html",
      "main": "main.ts",
      "polyfills": "polyfills.ts",
      "test": "test.ts",
      "tsconfig": "tsconfig.json",
      "testTsconfig": "tsconfig.spec.json",
      "prefix": "mt",
      "mobile": false,
      "styles": [
        "../node_modules/font-awesome/css/font-awesome.min.css",
        "../node_modules/admin-lte/bootstrap/css/bootstrap.min.css",
        "../node_modules/admin-lte/dist/css/AdminLTE.min.css",
        "../node_modules/admin-lte/dist/css/skins/skin-red.min.css",
        "../node_modules/admin-lte/plugins/iCheck/flat/red.css",
        "styles.css"  //a ordem dos arquivos css define uma hierarquia de sobrescrita neste caso o arquivo styles.css prevalece por ser o último
      ],
      "scripts": [
        "../node_modules/jquery/dist/jquery.min.js",
        "../node_modules/admin-lte/bootstrap/js/bootstrap.min.js",
        "../node_modules/admin-lte/dist/js/app.min.js"
      ],
      "environmentSource": "environments/environment.ts",
```

```
    "environments": {
      "dev": "environments/environment.ts",
      "prod": "environments/environment.prod.ts"
    }
  }
],
"addons": [],
"packages": [],
"e2e": {
  "protractor": {
    "config": "./protractor.conf.js"
  }
},
  "lint": [
  {
    "project": "src/tsconfig.app.json"
  },
  {
    "project": "src/tsconfig.spec.json"
  },
  {
    "project": "e2e/tsconfig.e2e.json"
  }
],
"test": {
  "karma": {
    "config": "./karma.conf.js"
  }
},
"defaults": {
  "styleExt": "css",
  "prefixInterfaces": false,
  "inline": {
    "style": false,
    "template": false
  },
  "spec": {
    "class": false,
```

```
    "component": true,
    "directive": true,
    "module": false,
    "pipe": true,
    "service": true
  }
 }
}
```

Seção 5

Vídeo 35

O que são Rotas?

É um recurso em Angular 4 que permite utilizar uma navegação dinâmica entre páginas.

EXTRA:

Single Page Application (SPA)

Uma Single Page Application (SPA) ou "Aplicação de Página Única" no português, é aquela que **NÃO RECARREGA** durante o uso e geralmente já carrega todos os scripts e CSS's de que precisa no seu primeiro carregamento. O usuário pode realizar uma série de tarefas sem que a página recarregue. Se você é novo nessa abordagem, deve pensar assim: Como assim não recarrega? Quer dizer que colocamos o conteúdo de todas as páginas em uma só? Não ficará confuso?". Sim, realmente ficará confuso. E é por isso que em SPA não colocamos tudo em uma página apenas, criamos algo muito **semelhante** a páginas para realizar essa separação.

Em SPA, a página index.html é carregada com todos os scripts e CSS's, mas mediante as ações do usuário, outras páginas são inseridas dinamicamente no corpo de index.html, através de Ajax. Quando o usuário acessar a página B.html, por exemplo, realizamos uma requisição Ajax para essa página e manipulamos o DOM de index.html para inserir o conteúdo de B.html. Se acessarmos a página C.html, removemos o conteúdo de B.html e inserimos o conteúdo de C.html. Por que eu usei o termo "**semelhante**"? Porque essas páginas não possuem as tags head, nem body, e, para serem exibidas precisam ser incluídas como conteúdo da página index.html.

Exemplos de SPA são o Gmail e o Inbox: quando você apaga um e-mail ou inicia a escrita de um novo, sua página recarrega (fica em branco aguardando carregamento)? Claro que não.

Pode parecer complicado realizar todo esse trabalho, mas SPA's fornecem uma experiência para o usuário parecida com aplicativos nativos, como o Gmail. Dependendo da sua aplicação, esse comportamento pode ser interessante.

Vantagens de uma SPA:

* Carregamento do HTML, CSS, JavaScript e outros recursos (não os dados) apenas uma única vez assim que o usuário entra na aplicação;

* Após o carregamento dos recursos apenas os dados serão trafegados na aplicação, por exemplo acessar através de Requisições Ajax uma lista de funcionários.

* Separação total de back-end e front-end uma vez que uma “página” (componente) for carregada apenas os dados serão trafegados o que permite reutilizar o back-end para outras aplicações, seja uma aplicação web ou mobile, por isso usamos com frequência a Arquitetura Rest para acessar a base de dados.

Desvantagens de uma SPA:

* Pode haver lentidão no carregamento do HTML, CSS, JavaScript e outros recursos (não os dados) assim que o usuário entra na aplicação, por exemplo um site de vendas que exiba páginas com imagens de produtos;

*

*

EXTRA:

Configurando rotas

Angular possui um módulo criado para ajudar nesse processo de mudança de página, o RouterModule. O primeiro passo será criar o arquivo alurapic/client/app/app.routes.ts. É nesse arquivo que teremos centralizadas as configurações de rotas da nossa aplicação.

Seção 5

Vídeo 36

Veja abaixo:

app.component.html

```
<div class="wrapper">
  <mt-header></mt-header>
    <!-- Full Width Column -->
  <div class="content-wrapper">
    <div class="container">
            <router-outlet></router-outlet>  // esta tag permitirá a substituição dinâmica entre páginas
                                             // a seguir estas páginas que substituirão dinamicamente o
                                             // componente router-outlet deverão ser mapeadas no
                                             // "src/app/app.routes.ts" através da constante ROUTES
                                             // do tipo Routes. Veja a baixo:
                                             // import {Routes} from '@angular/router'
                                             // const ROUTES: Routes = [ ….. ]
    </div>
  </div>
  <!-- snakbar deve ser aqui -->
  <footer class="main-footer">
    <div class="container">
      <div class="pull-right hidden-xs">
        <b>Version</b> 1.0.0
      </div>
      <strong>Meat</strong> Angular Application
    </div>
  </footer>
</div>
```

==========

/src/app/**app.routes.ts** => arquivo criado para definir as rotas a partir de um array.

```
import {Routes} from '@angular/router';

import {HomeComponent} from './home/home.component'; //componente criado por nos

import {AboutComponent} from './about/about.component'; //componente criado por nos

import {RestaurantsComponent} from './restaurants/restaurants.component'; //componente criado por nos


export const ROUTES: Routes = [    //array com as rotas, ou seja, mapeamento que contém cada caminho de um componente.

{path: '', component: HomeComponent},  //quando não for especificado um caminho '/' o módulo de roteamento do Angular
                                        //exibirá o conteúdo do componente HomeComponent.

{path: 'about', component: AboutComponent},  //quando for especificado o caminho '/about' o módulo de roteamento do Angular
                                              //exibirá o conteúdo do componente AboutComponent.

{path: 'restaurants', component: RestaurantsComponent}  //quando for especificado o caminho '/restaurants' o módulo de roteamento
                                                         //do Angular exibirá o conteúdo do componente RestaurantsComponent.

]
```

==========

<u>Importando as rotas criadas para dentro do módulo principal</u>

app.modules.ts

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

import { NgModule } from '@angular/core';

import { HttpModule } from '@angular/http';

import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'; //importando a constante ROUTES

import { AppComponent } from './app.component';

import { HeaderComponent } from './header/header.component';

import { HomeComponent } from './home/home.component';

import { AboutComponent } from './about/about.component';

import { RestaurantsComponent } from './restaurants/restaurants.component';


@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
   RestaurantsComponent
  ],
  imports: [
    BrowserModule,
```

```
  HttpModule,
  RouterModule.forRoot(ROUTES) //Importando o array de rotas – o módulo principal aceita apenas o método forRoot(…) para a
                               //chamada das rotas outros módulos aceitam o método forChild(…)
                               //Outro ponto importante  "RouterModule.forRoot" já nos dá acesso a tag <router-outlet>
                               //além de outros recursos.
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

==========

header.component.html

```
<!-- Início do header -->
 <header class="main-header">
 <nav class="navbar navbar-static-top">
  <div class="container">
   <div class="navbar-header">
    <a [routerLink]="['/']"class="navbar-brand"><b>Meat</b></a> // [routerLink]="['/']" => roteamento para o  conteúdo do
                                                                 // componente HomeComponent
    <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar-collapse">
     <i class="fa fa-bars"></i>
    </button>
   </div>
   <!-- Collect the nav links, forms, and other content for toggling -->
   <div class="collapse navbar-collapse pull-left" id="navbar-collapse">
    <ul class="nav navbar-nav">
      <li routerLinkActive="active"><a [routerLink]="['/restaurants']">Restaurantes</a></li>// [routerLink]="['/restaurants']" =>
                                                                                 // roteamento para o conteúdo do
                                                                                 // componente RestaurantsComponent
                                                                                 // routerLinkActive="active" =>tratamento css
      <li routerLinkActive="active"><a [routerLink]="['/about']">Sobre</a></li> // [routerLink]="['/about']" => roteamento para o
                                                                                 // conteúdo do componente AboutComponent
    </ul>
   </div>
   <!-- /.navbar-collapse -->
  </div>
 </nav>
 </header>
<!-- Fim do header →
```

Seção 5

Vídeo 37

restaurant.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant.model';

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

  @Input() restaurant : Restaurant //o componente @Input foi usado pois RestaurantComponent será utilizado como um
                                   //componente filho em RestaurantsComponent, componente parent.Veja abaixo:

  constructor() { }

  ngOnInit() {
  }

}
```

restaurants.component.html

```
<!-- INíCIO do CONTEÚDO -->
    <section class="content-header">
      <h1>
        Todos os Restaurantes
      </h1>
    </section>
    <section class="content">
      <div class="row">
        <div *ngFor="let rest of restaurants" class="col-sm-6 col-xs-12">
          <mt-restaurant  [restaurant]="rest" ></mt-restaurant>
        </div>
      </div>
    </section>
<!-- FIM  do CONTEÚDO –>
```

==========

Seção 5

Vídeo 38

Injeção de Dependência

É um padrão de projeto onde a aplicação deixa de instanciar seus objetos manualmente e passa a instanciação para o framework criando assim os objetos que a aplicação precisa utilizar. Para deixar um serviço disponível para ser utilizado em um componente será necessário declará-lo na ***lista de providers de um componente ou módulo***. O framework além de gerenciar a instanciação dos objetos gerencia também suas dependências. Se a injeção for realizada em um componente a mesma fica disponível para o componente e seus filhos, porém se a injeção for feita em um módulo o recurso ficará disponível para todos os componentes do módulo.

```
export class MyFirstComponent {  //==> exemplo de instanciação sem injeção de dependência feita em componente
        constructor () {
           this.firstService: new MyFirstService() // instanciação
         }
}

@Component({providers: [MyFirstService]}) //==> exemplo de injeção de dependência feita em componente na lista de providers.
                                           //É importante observar que tanto MyFirstComponent e seus filhos terão acesso a
                                                 //mesma instância de  MyFirstService, desta forma este serviço atua como
um                                               //Singleton.
export class MyFirstComponent {
        constructor (private firstService: MyFirstService){
        }
}

@NgModule({
        declarations: […]
        providers: [ MyFirstService ], //==> exemplo de injeção de dependência feita em um componente na lista de providers
                                       //É importante ressaltar que neste caso todos os componentes registrados em "declarations"
                                       //terão acesso a mesma instância de MyFirstService, desta forma este serviço atua como um
                                       //Singleton.
})
export class AppModule {}
```

EXTRA:

```
@Component({
  selector: 'app-cursos',
  templateUrl: './cursos.component.html',
  styleUrls: ['./cursos.component.css'],
  providers: [CursosService], //UMA INSTANCIA DE UM SINGLETON APENAS PARA ESTE COMPONENTE E SEUS FILHOS.
})
export class CursosComponent implements OnInit {
```

Vídeo 39

Serviços

São classes comuns em Angular que podem ser injetadas em componentes ou mesmo em outros serviços. Os serviços são utilizados para encapsular o acesso às APIs de Back-End.

Escopos utilizados para declarar uma API de Back-End

* Módulo ……. providers:[] ==> todos os serviços declarados na lista de ***providers*** ficarão disponíveis para todos os componentes e serviços do módulo raiz.

* Componentes e componentes filhos …… providers:[] ==> quando declarado na lista de ***providers*** os serviços ficarão disponíveis apenas para o serviço onde fora feita a declaração e seus componentes filhos.

* Componentes …… viewProviders:[] ==> neste caso somente o serviço que possui a lista de viewProviders será beneficiado com a injeção de dependência nem mesmo os seus componentes filhos serão beneficiados com o(s) recurso(s).

Serviços também podem solicitar a injeção de outros serviços para isso será necessário utilizar o decorate @Injectable ou você deverá criar os objetos desta classe de serviço com o operador ***new***.

Assista também o vídeo abaixo onde fala-se de contexto de um serviço com relação a módulos:

79. Criando um Core Module Com Serviços

Basicamente: módulos importados no modulo raiz terão as mesmas instâncias dos seus serviços compartilhados para todos os componentes declarados no módulo raiz, porém se o módulo for do tipo lazy loading os serviços serão compartilhados apenas pelos componentes do modulo lazy loading, ou seja, estes serviços não estarão disponiveis para outros modulos e nem para o modulo raiz.

<========== EXEMPLO EXTRA - INÍCIO ==========>

EXTRA*: @Injectable ==> singnifica que a classe que possui esta anotação pode receber injeções de dependência do framework angular, seja para serviços do próprio framework (Title, Http, Router etc….) ou classes de serviços customizadas, ou seja, criadas por nós. Vale ressaltar que componentes não precisam da anotação @Injectable para injetar classes de serviços bastando apenas utilizar estas classes como parâmetro no construtor do próprio componente.*

Veja um exemplo abaixo:

```
export class CursosService {
  constructor() {
```

```
  }
  getCursos(): string[] {
    return ['Java', 'PHP', 'JavaScript', 'SQL']
  }
}

import { CursosService } from './cursos.service';
import { Injectable } from '@angular/core';

//@Injectable() //==> significa dizer que esta classe pode receber injecoes de dependencia
//{providedIn: 'root'} //==> escopo raiz - logo a mesma instancia pode ser usada em toda aplicação

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

  //private cursosService : CursosService = new CursosService();
  //constructor() { } //sem injecao de dependendicia, veja o objeto sendo criado logo a cima.

  //com injecao de dependencia via construtor, para isso eh necessario que seja utilizado @Injectable() nesta classe (CursoMasterService).
  constructor(private cursosService : CursosService) {
  }

  getCursosMaster(): string[] {
    return this.cursosService.getCursos();
  }
}

import { CursoMasterService } from './curso-master.service';
import { CursosService } from './cursos.service';
import { Component, OnInit } from '@angular/core';

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

  linkGoogle : string
```

```
  cursos : string[] = []

  constructor(private cursoMasterService: CursoMasterService) {
  }

  ngOnInit() {
   this.cursos = this.cursoMasterService.getCursosMaster();
   this.linkGoogle = 'Google'
  }

}
```

<========== EXEMPLO EXTRA - FIM ==========>

```
import { Injectable } from '@angular/core'
import  { Http } from '@angular/http'
@Injectable() //==> injetando o serviço Http do framework
export class MyService {

        constructor (private http: Http){ //quando usamos private ou public na frente de um argumento do construtor
        }                                 //o TypeScript criará automaticamente uma propriedade, seja no serviço ou em
                                          //um componente.
                                          //Neste caso: private http; (no corpo deste serviço de forma invisível)
                                                      this.http = http; (dentro do construtor de forma invisível)

        list(){
            return this.http.get('/url')
        }
}
```

<u>Alguns Serviços do Framework</u>

Title: serviço utilizado para alterar e obter o título de uma página.

Http: serviço que encapsula o serviço Http.

Router: serviço que realiza navegação de forma programática.

Exemplo:

```
import { Title } from '@angular/platform-browser'
```

```
@Component ({
        viewProviders: [Title]
})
export class MyPageComponent {
    constructor(title: Title){
        title.setTitle (':: My Fancy Title ::')
    }
}
```

Seção 5

Vídeo 40

Utilizando uma classe de Serviço para retornar um array

/restaurants/restaurant/…

…/restaurant.model.ts

```
export interface Restaurant {
        id: string
        name: string
        category: string
        deliveryEstimate: string
        rating: number
        imagePath: string
}
```

…/restaurant.component.ts

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

import { Restaurant } from './restaurant.model';

@Component({
  selector: 'mt-restaurant',
  templateUrl: './restaurant.component.html'
})
export class RestaurantComponent implements OnInit {
 @Input() restaurant: Restaurant
  constructor() {
```

```
}
  ngOnInit() {
  }
}
```

…/restaurant.component.html

```
 <a href="#">
  <div class="place-info-box">
            <span class="place-info-box-icon"><img [src]="restaurant.imagePath" /></span>
            <div class="place-info-box-content">
              <span class="place-info-box-text">{{restaurant.name}}</span>
              <span class="place-info-box-star"><i class="fa fa-star"></i>{{restaurant.rating}}</span>
              <span class="place-info-box-detail">{{restaurant.category}}</span>
              <span class="place-info-box-detail">{{restaurant.deliveryEstimate}}</span>
            </div>
            <!-- /.info-box-content -->
  </div>
</a>
<!-- /.info-box -->
```

…/restaurants/…

.../restaurants.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';

@Component({
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html'
})
export class RestaurantsComponent implements OnInit {

restaurants: Restaurant[]

/*** TRANSFERIDO PARA A CLASSE DE SERVICO – TRECHO COMENTADO
  restaurants: Restaurant[] = [

                {
```

```
            id: "bread-bakery",
            name: "Bread & Bakery",
            category: "Bakery",
            deliveryEstimate: "25m",
            rating: 4.9,
            imagePath: "assets/img/restaurants/breadbakery.png"
          },
          {
            id: "burger-house",
            name: "Burger House",
            category: "Hamburgers",
            deliveryEstimate: "100m",
            rating: 3.5,
            imagePath: "assets/img/restaurants/burgerhouse.png"
          }
  ]
 ***/
  constructor(private restaurantsService: RestaurantsService) { // this. restaurantsService =  restaurantsService
}                                                                 // com o uso de public ou private teremos
                                                                  // automaticamente uma propriedade do tipo
                                                                  // restaurantsService no nosso componente


  ngOnInit() { // este é um método de ciclo de vida cujo o qual será executado uma única vez quando o nosso
               // componente for chamado na tela.
        this.restaurants = this.restaurantsService.restaurants()
  }


}
```

…/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model';
export class RestaurantsService {
        constructor(){ }
        /*** CRIACAO DO ARRAY ***/
        rests: Restaurant[] = [
                {
                 id: "bread-bakery",
                name: "Bread & Bakery",
                category: "Bakery",
                deliveryEstimate: "25m",
                rating: 4.9,
                imagePath: "assets/img/restaurants/breadbakery.png"
            },
            {
                id: "burger-house",
```

```
                name: "Burger House",
                category: "Hamburgers",
                deliveryEstimate: "100m",
                rating: 3.5,
                imagePath: "assets/img/restaurants/burgerhouse.png"
            }
    ]
    /*** METODO QUE RETORNARA O ARRAY ***/
    restaurants(): Restaurant[]{
        return this.rests;
    }
}
```

…/restaurants.component.html

```
<!-- INíCIO do CONTEÚDO -->
    <section class="content-header">
      <h1>
        All Restaurantes
      </h1>
    </section>

     <section class="content">
      <div class="col-sm-6 col-xs-12" *ngFor="let rest of restaurants">
      <mt-restaurant [restaurant]="rest"></mt-restaurant> <!-- o componente mt-restaurant possui uma propriedade de
                                                                nome restaurant -->
      </div>
    </section>
```

…/app/app.module.ts

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

import {ROUTES} from './app.routes';

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
```

```
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component';
import { RestaurantsService } from './restaurants/restaurants.service'; //importando a classe de serviço

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
    RestaurantsComponent,
    RestaurantComponent

  ],
  imports: [
    BrowserModule,
    HttpModule,
    RouterModule.forRoot(ROUTES)
  ],
  providers: [RestaurantsService], // declarando RestaurantsService na lista de providers para obter a injeção
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Seção 5

Vídeo 41

Programação reativa é programar com fluxos(streams) de dados assíncronos. Assíncrono significa que não se realiza ao mesmo tempo ou no mesmo ritmo de desenvolvimento (em relação a outra coisa).

Isso é importante devido o crescimento da Internet e a demanda por tempo real, a programação precisa ser dinâmica, ou seja; diferente da forma tradicional de desenvolvimento.
A forma tradicional de programar/desenvolver, por exemplo, cria várias tarefas e elas se comunicam em tempos determinados, com respostas determinadas sem escalabilidade caso ocorra alguma falha, alguma demanda maior, são "rígidas", seguem regras diretas. Isto funcionava e continua sendo utilizada até hoje, entretanto esta "lógica" não é mais compatível com a necessidade atual.
No método tradicional é como preparar a mistura do cimento, depois que este enviou os dados dizendo que está preparado, é chamada a função de construir a parede, depois o reboco, esperar um tempo determinado, passado o período é chamada a função de pintar e se faltar o pintor? Ferrou tudo, deu erro no sistema! Ou se a comunicação falhar? Erro também, sistema trava, geralmente é preciso refazer tudo!

Na programação reativa isso tudo ocorre, mas de forma inteligente, interligada em paralelo, sem seguir aquela ordem cronológica e linear, como por exemplo: a parede esta construída, falta o reboco, tempo pra secar, depois pintar. Não tem pintor, falha nesta tarefa, o sistema não irá travar, reconhece que é possível continuar outras partes da construção em paralelo como o chão, botar o piso, enquanto fica em espera a tarefa de pintar, sem ter que recomeçar tudo outra vez, concluir depois a pintura, sem ter sido interrompido o sistema pelo "erro".
É por aí a lógica, parece meio óbvio se for pensar, mas os sistemas ainda são pouco inteligentes, precisam ser devidamente programados para estarem aptos a gerenciar falhas, rotinas diferentes, seguir caminhos alternativos, muitos acessos, funcionar em tempo real e sempre estar online/ativo!

A biblioteca que Angular usa para Reactive Programming é o RxJS.

RxJS é uma biblioteca para uso em projetos que utilizam Javascript (incluindo JSX e TypeScript), que faz parte do projeto de Programação Reativa, daí que vem o Rx, de Reactive Extension. Além dela, existem também outros projetos voltados para a linguagem Java, Python, C++, etc.

Reactive Programming é a combinação de dois padrões Itarator e Observer.

Iterator: permite a navegação sequencial ao longo de coleções de objetos.

Observer: este padrão de projeto torna possível que objetos relacionados a uma determinada estrutura sejam notificados quando há mudanças de estado ocorridas nesta última.

Promises ==> um único evento

Observable ==> múltiplos eventos

Em RxJS o objeto principal é Observable.

Métodos da API Http: get – post - put – delete – head – options retornam um objeto do tipo Observable<Response>

Exemplo:

```
this.http.get('/url')
                .map(response => response.json()) // o método map restornará um Observable
                 .subscribe(data=>this.mydata=data) // a requisição é feita sempre depois que a aplicação
                                                    // se inscreve com o método subscribe.
```

Funções da classe Observable:

```
Observable.from([1, 2, 3, 4, 5])
        .map(x => x + 3)
        .filter(x => x %2 = =0)
        .subscribe(x => console.log(x)) //4, 6, 8
```

Refazendo um chamada Http:

Exemplo:

```
this.http.get('/url')
               .retry(2) // refaz uma chamada Http
                .map(response => response.json()) // o método map restornará um Observable
                 .subscribe(data=>this.mydata=data) // a requisição é feita sempre depois que a aplicação
                                                    // se inscreve com o método subscribe no componente
                                                    // enquanto no servico configuramos o Observable.
```

Realizando múltiplos mapeamentos Http até encontrar a chamada desejada:

```
this.http.get('/url')
                .map(response => response.json()) // o método map restornará um Observable
                .map(result => result.id) // o método map restornará um Observable
                .subscribe(data=>this.mydata=data) // a requisição é feita sempre depois que a aplicação
                                                   // se inscreve com o método subscribe.
```

Quando um objeto se inscreve em um Observable será necessário remover a inscrição posterior para evitar Memory Leaks, porém um Observable da API Http, Router e Async não precisam de um cancelamento de inscrição, ou seja, **unsubscribe**.

Memory Leaks: ou vazamento de memória, é um fenômeno que ocorre em sistemas computacionais quando uma porção de memória, alocada para uma determinada operação, não é liberada quando não é mais necessária.

Lidando com erros – operador catch

```
save(myData){
        return this.http.post('/url', JSON.stringify(myData))
                                .map(response => reponse.json())
                                 .map(result => result.id)
                                catch(error => {
                                        console.log(error)
                                        Observable.throw(`Error posting ${myData}`)
                                })
}
```

Seção 5

Vídeo 42

Configurando a API de Back-End (Json - server)

Como precisaremos utilizar Http será necessário utilizar uma API de back-end neste caso será utilizado o Json-server, logo precisaremos baixar a biblioteca e para isso entre **na raiz da sua aplicação** e execute os comandos abaixo:

```
>npm install -g json-server   //instalará a biblioteca json-server
>json-server db.json //inicializando
//json-server -p 4000 db.json //rodando o json-server em uma porta específica
```

Observação:

O arquivo db.json foi um arquivo criado manualmente por nós para ser manipulado pela biblioteca do Json-server.

Após a inicialização será exibida uma série de endereços que pode ser acessados pelo navegador constatando assim o sucesso da instalação com a exibição dos dados contidos no arquivo db.json. Veja alguns exemplos de endereços abaixo obtidos após a instalação e inicialização da biblioteca json-server :

Resources
http://localhost:3000/restaurants
http://localhost:3000/menu
http://localhost:3000/reviews
http://localhost:3000/orders

Home
http://localhost:3000

Seção 5

Vídeo 43

<u>Incializando Json-Server</u>

>npm install -g json-server //instalará a biblioteca json-server

>json-server db.json //inicializando o json-server

//json-server -p ***4000*** db.json //rodando o json-server em uma ***porta específica***

Observação: db.json => nossa massa de dados no formato "json" previamente cridada

<u>Acessando o recurso pelo browser</u>

http://localhost:3000/restaurants

Saída:


```
[

  {
    "id": "bread-bakery",
    "name": "Bread & Bakery",
    "category": "Bakery",
    "deliveryEstimate": "25m",
    "rating": 4.9,
    "imagePath": "assets/img/restaurants/breadbakery.png",
    "about": "A Bread & Brakery tem 40 anos de mercado. Fazemos os melhores doces e pães. Compre e confira.",
    "hours": "Funciona de segunda à sexta, de 8h às 23h"
  },
  {
    "id": "burger-house",
    "name": "Burger House",
    "category": "Hamburgers",
    "deliveryEstimate": "100m",
    "rating": 3.5,
    "imagePath": "assets/img/restaurants/burgerhouse.png",
    "about": "40 anos se especializando em trash food.",
    "hours": "Funciona todos os dias, de 10h às 22h"
  },
  {
    "id": "coffee-corner",
    "name": "Coffee Corner",
    "category": "Coffee Shop",
    "deliveryEstimate": "30-40m",
    "rating": 4.8,
    "imagePath": "assets/img/restaurants/coffeecorner.png",
    "about": "A Coffe Corner foi eleita a melhor cafeteria da cidade.",
    "hours": "Funciona de segunda à sábado, de 6h às 22h"
  },
  {
    "id": "green-food",
    "name": "Green Food",
    "category": "Saudável",
```

```
    "deliveryEstimate": "75m",
    "rating": 4.1,
    "imagePath": "assets/img/restaurants/greenfood.png",
    "about": "Comida saudável é no Green Food. Compramos barato, vendemos caro. ;)",
    "hours": "Somente em horário de almoço, das 11h às 15h"
  },
  {
    "id": "ice-cream",
    "name": "Ice Cream",
    "category": "Ice Creams",
    "deliveryEstimate": "40-65m",
    "rating": 4.5,
    "imagePath": "assets/img/restaurants/icy.png",
    "about": "A Ice Cream é uma sorveteria inovadora.",
    "hours": "Funciona todos os dias, de 10h às 1h"
  },
  {
    "id": "tasty-treats",
    "name": "Tasty Treats",
    "category": "Doces",
    "deliveryEstimate": "20m",
    "rating": 4.4,
    "imagePath": "assets/img/restaurants/tasty.png",
    "about": "A doceria com mais tradição da cidade",
    "hours": "Funciona de segunda à sábado, de 8h às 23h"
  }
]
```


Criando um arquivo para obter a massa de dados carregada no Json-server

/app/app.api.ts

export const MEAT_API = 'http://localhost:3000' //exportando uma constante (const) de nome MEAT_API

Modificando nossa Classe de Serviço para chamar a API do Json-server

/app/restaurants/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'

import {MEAT_API} from '../app.api'

import {Injectable} from '@angular/core'

import {Http} from '@angular/http'

import {Observable} from 'rxjs/Observable'

import 'rxjs/add/operator/map'


@Injectable() //para uma classe de serviço,  RestaurantsService, poder receber uma injeção de outro serviço, Http neste caso, precisamos da anotação
              //@Injectable()

export class RestaurantsService {

    constructor(private http: Http){ }


    restaurants(): Observable<Restaurant[]> { // Observable<Response> => pertence a API RxJS (Reactive Extensions for JavaScript)

     return this.http.get(`${MEAT_API}/restaurants`).map(response => response.json()) //Todos os métodos da API Http retornam um
                                                                                        //Observable<Response> que é uma resposta crua.

     }                                                                                  //A função "map(...)" retorna um json
                                                                                        //"response => response.json()" mapeando a resposta,

                                                                                        //ou seja, o objeto response para o formato "json"

}//Um objeto do tipo Observable<Response> possui muitos dados além dos dados no formato "json", por isso chamamos de resposta crua, logo
//precisamos preparar este objeto para retornar apenas os dados "json" devido a isso usamos a função map.
```

Observação:

Métodos do API Http: request - get - post - put – delete – patch – head - options

RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code (RxJS Docs).

/restaurants /restaurants.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';

@Component({
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html'
})
export class RestaurantsComponent implements OnInit {
  restaurants: Restaurant[]
  constructor(private restaurantsService: RestaurantsService) { }

  ngOnInit() {
        this.restaurantsService.restaurants().subscribe(restaurants => this.restaurants = restaurants)
  }//A função subscribe(….) é responsável por capturar cada requisição Http já formatada em json.

    //A função subscribe(….) deve ser feita no componente.

}
```

Seção 5

Vídeo 44

Operador catch

/restaurant/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
```

```
@Injectable()
export class RestaurantsService {
   constructor(private http: Http){ }
    restaurants(): Observable<Restaurant[]>{
          return this.http.get(`${MEAT_API}/restaurants`) //o método get(...) apenas configura o Observable
                                                          //para acessar uma url quando alguém chamar o
                                                          //o método subscribe(…) também do  Observable.
                        .map(response => response.json())//aqui troca-se o objeto Response pelo corpo da
                                                           //resposta no formato em "json"
   }
}
```

<u>Existem duas formas de tratar erros no Angular:</u>

1) Implementando a interface ErrorHandler: trata erros de forma global para isso deve ser configurado como um providers: […] no módulo app.modulo.ts

2) Utilizando o operador catch() [import 'rxjs/add/operator/catch']:

/restaurant/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
import 'rxjs/add/operator/catch' //para tratar erros
import {ErrorHandler} from '../app.error-handler' //classe criada para logar os erros na classe  RestaurantsService
@Injectable()
export class RestaurantsService {
   constructor(private http: Http){ }
    restaurants(): Observable<Restaurant[]>{
          return this.http.get(`${MEAT_API}/restaurants`) //configura o Observable  para acessar uma url
                      .map(response => response.json())// troca-se o Response pelo formato em "json"
                      .catch(ErrorHandler.handleError){ //trata o erro de forma básica – necessário criar um arquivo
                  }
   }
```

```
}
```

/app/ ErrorHandler

```
import {Response} from '@angular/http'
import 'rxjs/add/observable/throw'
import {Observable} from 'rxjs/Observable'
export class ErrorHandler  {
	static handleError(error: Response  | any){//Com Response será possível tratar o tipo do erro, ver o tipo do erro,
	//o nível do status, identificar a url etc…
	//Com any será possível tratar qualquer outro tipo de erro
	//que não seja Response.
	let errorMessage: string
	if(error instanceof Response){
		errorMessage = `Erro ${error.status} ao acessar a URL ${error.url} - ${error.statusText}`
	}else{
		errorMessage = error.toString()
	}
		console.log(errorMessage)
		return Observable.throw(errorMessage);
	}
}
```

Forçando um Erro para ser capturado pelo catch(…) com o uso da nossa classe ErrorHandler

Na classe de serviço faça: this.http.get(`${MEAT_API}/restaurantsX`) //esta URL não existe

/restaurants/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
import 'rxjs/add/operator/catch'
import {ErrorHandler} from '../app.error-handler' //classe criada por que logara os erros

@Injectable()
export class RestaurantsService {

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

```
 restaurants(): Observable<Restaurant[]>{
            return this.http.get(`${MEAT_API}/restaurantsX`) //esta URL não existe
                        .map(response => response.json())
                        .catch(ErrorHandler.handleError)//captura o erro e será tratado com o método handleError (…)
                                                //da classe  ErrorHandler
   }
}
```

Observação:

Como a url `${MEAT_API}/restaurantsX`) não existe no mapeamento; o tipo de erro abaixo será visível na ferramenta Developer Tools o que acarretará a não exibição dos restaurantes.

*** Failed to load resource: the server responded with a status of 404 (Not Found)**
app.error-handler.ts:20 Erro 404 ao acessar a URL http://localhost:3000/restaurantsXS - Not Found

Seção 5

Vídeo 45

Parametrizando Rotas

A parametrização pode ser utilizada para identificar um objeto que está armazenado no back-end.

/app/app.routes.ts

```
import {Routes} from '@angular/router';
import {HomeComponent} from './home/home.component';
import {AboutComponent} from './about/about.component';
import {RestaurantsComponent} from './restaurants/restaurants.component';

export const ROUTES: Routes = [
        {path: '', component: HomeComponent},
        {path: 'about', component: AboutComponent},
        {path: 'restaurants', component: RestaurantsComponent},
        {path: 'restaurant/:id', component: RestaurantsComponent}
]

<!-- passando um parametro →
<a [routerLink]="['/restaurant', restaurant.id]">Bakery</a>
```

2 Formas de obter Parâmetros

*** A partir de um Snapshot:** muito útil quando não estamos preocupados se a rota mudará e o nosso componente será destruído e voltará novamente.

```
export class MyComponent implements OnInit {


myObj: any
constructor(private route: ActivatedRoute){}// ActivatedRoute ==> objeto que respresenta a rota ativa no momento;
                                            //a partir de um objeto deste tipo pegamos o valor do parâmetro que fora
                                             //passado pela url.
ngOnInit(){
        const id = this.route.snapshot.params['id'] //obtendo o valor do id
        this.myObj = //... obter os dados baseado no id
  }
}
```

*** Subscribe:** utilizado sempre que a rota mudar, por exemplo: /restaurante/**id**/details

```
export class MyComponent implements OnInit {


myObj: any
constructor(private route: ActivatedRoute){}// ActivatedRoute ==> objeto que respresenta a rota ativa no momento
                                            // ActivatedRoute ==> a partir de um objeto deste tipo pegamos o valor
                                             // do parâmetro.


ngOnInit(){
        this.route.subscribe(params => {
        const id = params['id']
        this.myObj = //... obter os dados baseado no id
      })
    }
}
```

Seção 5

Vídeo 47

Retornando um Objeto Parametrizado por Id

restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
import 'rxjs/add/operator/catch'
import     {ErrorHandler} from '../app.error-handler' //classe criada para logar os erros




@Injectable()
export class RestaurantsService {

constructor(private http: Http){ }

 restaurants(): Observable<Restaurant[]>{ //retando um Observable com uma lista de restaurantes
            return this.http.get(`${MEAT_API}/restaurants`)
                                                .map(response => response.json())
                                                .catch(ErrorHandler.handleError)
    }

restaurantById(id: String): Observable<Restaurant>{ //retando um Observable com um restaurante
            return this.http.get(`${MEAT_API}/restaurants/${id}`)
                                                .map(response => response.json())
                                                .catch(ErrorHandler.handleError)
    }
}
```

restaurant-detail.component.ts

```
import {Component, OnInit} from '@angular/core';
import {RestaurantsService} from '../restaurants/restaurants.service';
import {Restaurant} from '../restaurants/restaurant/restaurant.model';
import {ActivatedRoute} from '@angular/router';

@Component({
  selector: 'mt-restaurant-detail',
```

```
  templateUrl: './restaurant-detail.component.html'
})
export class RestaurantDetailComponent implements OnInit {


  restaurant : Restaurant //como o valor passado para a propriedade "restaurant" será atribuido pelo
                                //componente RestaurantDetailComponent não utilizaremos @Input na propriedade restaurant
  constructor(private restaurantsService: RestaurantsService, private route: ActivatedRoute) { }
// ActivatedRoute : um objeto da classe  ActivatedRoute permite identificar a url acionada bem como os parâmetros passados nesta url.
//O acesso aos parâmetros pode ser feito com snapshot ou subscribe. Veja: this.route.snapshot.params['id'] / this.route.subscribe.params['id']
  ngOnInit() {
                tthis.restaurantsService.restaurantById(this.route.snapshot.params['id']).subscribe(restaurant => this.restaurant = restaurant)
  }
  //O objeto ActivatedRoute fornece qual foi a url acionada e
  //qual foi(ou foram) o(s) parâmetro(s) passado(s) na url
  //this.route.snapshot ==> faz o acesso ao objeto uma única vez como se fosse uma foto
  //this.route.snapshot.params[‘xxxxx’] ==> acessa o valor do parâmetro
  //this.route.snapshot.params['id'] ==> acessa o valor do parâmetro id

}
```

restaurant-detail.component.html

```
 <!-- INÍCIO DO CONTEÚDO -->
     <section class="content-header">
       <h1>
       </h1>
     </section>


  <section class="content">
       <div class="row">
          <div class="col-xs-12">
            <div class="box box-solid">
              <div class="box-header with-border">
                <i class="fa fa-home"></i>


                <h3 class="box-title">{{restaurant?.name}}</h3><span class="pull-right"><i class="fa fa-star"></i>{{restaurant?.rating}}</span>
              </div>
              <!-- /.box-header -->
              <div class="box-body">
                <div class="col-sm-3 col-xs-12" *ngIf="restaurant">
                  <img class="box-img-detail" [src]="restaurant?.imagePath" />
                </div>
```

```
            <dl class="col-sm-9 col-xs-12">
              <dt>Categoria</dt>
              <dd>{{restaurant?.category}}</dd>
              <dt>Quem somos</dt>
              <dd>{{restaurant?.about}}</dd>
              <dt>Horários</dt>
              <dd>{{restaurant?.hours}}</dd>
            </dl>
          </div>
          <!-- /.box-body -->
          <div class="box-footer detail-footer">
          <a class="pull-left detail-active" href="detail-restaurant.html">
            Menu
          </a>
          <a class="pull-right" href="detail-restaurant-reviews.html">
            Avaliações
          </a>
        </div>
        </div>
      </div>
      <!-- /.col-xs-12 -->

    </div>
    <div class="row">
      <mt-menu></mt-menu>
    </div>
  </section>
  <!-- FIM DO CONTEÚDO -->
```

Seção 5

Vídeo 48

Rotas (Navegação) entre dois Componentes Filhos

app.route.ts

```
import {Routes} from '@angular/router';

import {HomeComponent} from './home/home.component';
import {AboutComponent} from './about/about.component';
import {RestaurantsComponent} from './restaurants/restaurants.component';
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component';
```

```
import {MenuComponent} from './restaurant-detail/menu/menu.component';
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component';

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        {path: 'about', component: AboutComponent},
        {path: 'restaurants/:id', component: RestaurantDetailComponent ,
                children: [
                        {path: 'menu', component: MenuComponent},
                        {path: 'reviews', component: ReviewsComponent}
        ]},
        {path: 'restaurants', component: RestaurantsComponent}
]
```

restaurant-detail.component.html

```
<!-- INÍCIO DO CONTEÚDO -->
  <section class="content-header">
    <h1>
    </h1>
  </section>
  <section class="content">
    <div class="row">
      <div class="col-xs-12">
        <div class="box box-solid">
          <div class="box-header with-border">
            <i class="fa fa-home"></i>
            <h3 class="box-title">{{restaurant?.name}}</h3><span class="pull-right"><i class="fa fa-star"></i>{{restaurant?.rating}}</span>
          </div>
          <!-- /.box-header -->
          <div class="box-body">
            <div class="col-sm-3 col-xs-12" *ngIf="restaurant">
              <img class="box-img-detail" [src]="restaurant?.imagePath" />
            </div>
            <dl class="col-sm-9 col-xs-12">
              <dt>Categoria</dt>
              <dd>{{restaurant?.category}}</dd>
              <dt>Quem somos</dt>
              <dd>{{restaurant?.about}}</dd>
              <dt>Horários</dt>
```

```
            <dd>{{restaurant?.hours}}</dd>
          </dl>
        </div>
        <!-- /.box-body -->
        <div class="box-footer detail-footer">
        <a class="pull-left" routerLinkActive="detail-active" [routerLink]="['menu']" >
          Menu
        </a>
        <a class="pull-right" routerLinkActive="detail-active" [routerLink]="['reviews']" >
          Avaliações
        </a>
      </div>
      </div>
    </div>
    <!-- /.col-xs-12 -->
  </div>
  <div class="row">
    <!-- <mt-menu></mt-menu> -->
    <router-outlet></router-outlet>
  </div>
</section>
<!-- FIM DO CONTEÚDO -->
```

Seção 5

Vídeo 49

O que são Pipes

Um pipe é responsável por transformar um valor de entrada em outro valor desejado. Por exemplo, ele pode receber uma data e formatá-la para o padrão brasileiro. Veja os exemplos abaixo:

```
//in component
user: = {name: 'Luke Skywalker', isJedi: true}

<!-- in template -->
<div>{{user | json}}</div>  ==> {name: 'Luke Skywalker', isJedi: true} //util para debugar o valor de um objeto

<!-- in template -->
<div>{{user.nome | uppercase}}</div>  ==> LUKE SKYWALKER

<!-- in template -->
<div>{{user.nome | lowercase}}</div>  ==> luke sywalker

<!-- in template -->
<div>{{0.7 | lowercase}}</div>  ==> 70%
```

```
<!-- in template -->
<div>{{user.birthday | date:  'dd/MM/yyyy'}}</div>  ==>  12/12/1999

<!-- in template -->
<div>{{user.salary | currency}}</div>  ==>  USD45.00  //o pipe currency possui até três parâmetros opcionais

<!-- in template -->
<div>{{user.salary | currency: 'BRL' true}}</div>  ==>  R$45.00  //BRL: moeda //true: exibe o símbolo de moeda

<!-- in template -->
<div>{{user.name | slice 0:4}}</div>  ==>  Luke   //limita a quantidade de caracteres da string

<!-- in template -->
<div>{{['one', 'two'] | slice 0:1 | json | uppdercase}}</div>  ==>  ["ONE"]   // veja que é possível combiná-los
```

Extra: Comandos Angular

> ng generate component <nome>

Exemplo de uso:

> ng generate component favorito

O resultado da execução desse comando é:

```
ng generate component favorito
installing component
create src/app/favorito/favorito.component.css
create src/app/favorito/favorito.component.html
create src/app/favorito/favorito.component.spec.ts
create src/app/favorito/favorito.component.ts
update src/app/app.module.ts
```

> ng generate class <nome>

Exemplo de uso:

> ng generate class produto

A execução desse comando criará o arquivo /src/app/*produto.ts* contendo uma classe chamada Produto.

> ng generate enum <nome>

Exemplo de uso:

> ng generate enum situacaoProduto

O comando do exemplo cria o arquivo /src/app/status-produto.ts contendo um enum chamado SituacaoProduto.

> ng generate interface <nome>

Exemplo de uso:

> ng generate interface IProdutoService

A execução desse comando criará um arquivo /src/app/iproduto-service.ts contendo a declaração de uma interface com nome IProdutoService.

> ng generate module <nome>

> ng g m <nome> - -routing //criará módulo com arquivo de roteamento

Exemplo de uso:

> ng generate module admin

O resultado dessa operação é a criação de um diretório chamado admin e um arquivo, dentro dessa pasta, chamado admin.module.ts.

> ng generate service <nome>

Exemplo de uso:

ng generate service api

O resultado disso será a criação do arquivo /src/app/api.service.ts contendo a classe que representa o serviço e o arquivo /src/app/api.service.spec.ts, que contém os testes dessa classe.

> ng generate pipe <nome>

Muito utilizado em aplicações Angular, um pipe é responsável por transformar um valor de entrada em outro valor desejado. Por exemplo, ele pode receber uma data e formatá-la para o padrão brasileiro. A criação de pipes também é bem simples usando o CLI, conforme pode ser visto no exemplo abaixo.

Exemplo de uso:

> ng generate pipe date

O resultado desse comando será a criação de dois arquivos: /src/app/date.pipe.ts e /src/app/date.pipe.spec.ts. O primeiro contém a declaração da classe que representa o pipe, enquanto o segundo contém os testes.

Além disso, o arquivo /src/app/app.module.ts é atualizado para informar ao Angular a existência desse novo artefato.

Seção 5

Vídeo 50

Criando o Componente Reviews de Restaurant

1) Crie o método na classe de serviço neste caso RestaurantService, pois será o restaurante que conterá os reviews.

2) Crie a chamada do método da classe de serviço, RestaurantService, no componente **reviews.component.ts**, neste

exercício usaremos **"| async"** ao invés do subscribe(...).

3) Implementaremos **"| async"** na página **reviews.component.html.** Perceba que neste caso não criamos um componente para reviews apenas por uma questão de escolha. Vale ressaltar que o **"| async"** faz o subscribe do Observable automaticamente.

restaurant.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
import 'rxjs/add/operator/catch'
import {ErrorHandler} from '../app.error-handler' //classe criada para logar os erros
import {MenuItem} from '../restaurant-detail/menu-item/menu-item.model'

@Injectable()
export class RestaurantsService {
 constructor(private http: Http){ }
 restaurants(): Observable<Restaurant[]>{ //retornando um Observable com uma array de restaurantes
            return this.http.get(`${MEAT_API}/restaurants`)
                        .map(response => response.json())
                        .catch(ErrorHandler.handleError)
   }

   restaurantById(id: string): Observable<Restaurant>{ //retornando um Observable com um restaurante
            return this.http.get(`${MEAT_API}/restaurants/${id}`)
                        .map(response => response.json())
                        .catch(ErrorHandler.handleError)
   }

   reviewsOfRestaurant(id: string): Observable<any>{  //retornando um Observable<any> neste caso com reviews
      return this.http.get(`${MEAT_API}/restaurants/${id}/reviews`)// Veja "/reviews" no arquivo app.routes.ts
                  .map(response => response.json())
                  .catch(ErrorHandler.handleError)
   }

//     menuOfRestaurant(id: string): Observable<MenuItem[]>{  //retornando um Observable<MenuItem[]>
//       return this.http.get(`${MEAT_API}/restaurants/${id}/menu`)// Veja "/menu" no arquivo app.routes.ts
```

```
//                          .map(response => response.json())
//                         .catch(ErrorHandler.handleError)
//    }

}
```

reviews.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RestaurantsService } from '../../restaurants/restaurants.service';
import {Observable} from 'rxjs/Observable';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'mt-reviews',
  templateUrl: './reviews.component.html'
})
export class ReviewsComponent implements OnInit {

  reviews: Observable<any> //variavel "reviews" sendo usada como referencia no retorno do metodo
                    //do nosso servico. Em geral usariamos o metodo "subscribe(...)"
                    //Veja o html com pipe + async deste componente no caso "mt-reviews"

  constructor(private restaurantsService: RestaurantsService,
                                    private route: ActivatedRoute) { }

  ngOnInit() {
            this.reviews = this.restaurantsService.reviewsOfRestaurant(this.route.parent.snapshot.params['id'])
            //this.route.parent.snapshot.params => como o parametro "id" passado pertence ao componente pai (restaurant)
            //precisaremos do comando "parent" para chegar nesta rota alternativa neste caso   "/restaurants/${'id'}/reviews"
            }

}
```

reviews.component.html

```
  <!-- INICIO DO conteúdo -->
  <div class="col-xs-12 col-sm-6" *ngFor="let rvw of reviews | async">

  <div class="box box-solid">
    <div class="box-header with-border">
      <span class="review-score pull-right"><i class="fa fa-star"></i>{{rvw.rating}}</span>
```

```
    <div class="user-block">
      <img *ngIf="rvw.rating < 3" alt="User Image" class="img-circle" src="assets/img/reactions/pissed.png">
      <img *ngIf="rvw.rating >= 3 && rvw.rating < 4" alt="User Image" class="img-circle" src="assets/img/reactions/soso.png">
      <img *ngIf="rvw.rating >= 4" alt="User Image" class="img-circle" src="assets/img/reactions/soso.png">
      <span class="username">{{rvw.name}}</span>
      <span class="description">{{rvw.date | date: 'dd/MM/yyyy'}}</span>
    </div>
  </div>

    <div class="box-body">
      <p>{{rvw.comments}}</p>
    </div>

 </div>
 <!--FIM DO conteúdo -->
```

Seção 5

Vídeo 51

Criando os Itens de Menu – Pipe currency e Eventos de Componentes

/menu-item/menu-item-component.ts

//Teremos um array de menu-item que será retornado por um método do service, logo teremos um ngFor que ira iterar sobre este array e precisaremos de um objeto menu-item para receber estes valores e exibi-los através de um componente em html.

```
export interface MenuItem {
        id: string
        name: string
        description: string
        price: number
        imagePath: string
}
```

menu-item-component.ts

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

```
import { MenuItem } from './menu-item.model';

@Component({
  selector: 'mt-menu-item',
  templateUrl: './menu-item.component.html'
})
export class MenuItemComponent implements OnInit {

        //Aqui, estamos criando uma propriedade que represente um "item do menu" para que o componente parent
        //que realmente tenha os itens passe os mesmos para que popule o template, ou seja, que venham da
        //API JSON.

 @Input() menuItem: MenuItem //Sempre que tivermos uma propriedade onde o componente parent, MenuComponent,
                             //irá informar um valor para o componente filho, MenuItemComponent, precisaremos
                             //marcar esta propriedade com o "decorate input" neste caso a propriedade "menuItem".
                             //Agora nosso componente menu-item.component está pronto, a seguir
                            //trabalharemos no nosso template neste caso o "menu-item.componente.html".

  //EventEmitter --> classe usada para criar eventos

  @Output() add = new EventEmitter() //todos os eventos são marcados com o decorate @Output()

 constructor() { }

  ngOnInit() {
  }

  //esta funcao esta sendo chamada no componente "menu-item.componente.html"
  emitAddEvent(){
        this.add.emit(this.menuItem)
        //this.add.emit('>>>>>>>>>'+ this.menuItem)
  }

  //Apos criamos o metodo acima devemos criar um metodo no restaurante.service, componente parent,
  //para fazer a busca e a seguir criar a iteracao no componente "menu-item.componente.html" o qual
  //será feito por último.

  //Agora no componente menu.component.ts devemos criar a referencia que será carregada com
  //o resultado do metodo menuOfRestaurant usando subscribe(...) ou async
}
```

/menu-item/menu-item.component.html

```
<div class="menu-item-info-box">
        <span class="menu-item-info-box-icon"><img [src]="menuItem.imagePath"></span>
        <div class="menu-item-info-box-content">
          <span class="menu-item-info-box-text">{{menuItem.name}}</span>
          <span class="menu-item-info-box-detail">{{menuItem.description}}</span>
          <span class="menu-item-info-box-price">{{menuItem.price | currency: 'BRL' : true}}</span>
          <a (click)="emitAddEvent()"><i class="fa fa-plus-circle"></i> Adicionar</a>
        </div>
        <!-- /.info-box-content -->
</div>
<!-- /.info-box -->
```

restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Injectable} from '@angular/core'
import {Http} from '@angular/http'

import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map'
import 'rxjs/add/operator/catch'
import {ErrorHandler} from '../app.error-handler' //classe criada para logar os erros
import {MenuItem} from '../restaurant-detail/menu-item/menu-item.model'

@Injectable()
export class RestaurantsService {

constructor(private http: Http){ }



  restaurants(): Observable<Restaurant[]>{ //retornando um Observable com uma lista de restaurantes
        return this.http.get(`${MEAT_API}/restaurants`)
                .map(response => response.json())
                .catch(ErrorHandler.handleError)
  }
```

```
  restaurantById(id: string): Observable<Restaurant>{ //retornando um Observable com um restaurante
          return this.http.get(`${MEAT_API}/restaurants/${id}`)
                        .map(response => response.json())
                        .catch(ErrorHandler.handleError)
  }

  reviewsOfRestaurant(id: string): Observable<any>{  //retornando um Observable<any> neste caso com reviews
    return this.http.get(`${MEAT_API}/restaurants/${id}/reviews`)// Veja "/reviews" no arquivo app.routes.ts
                  .map(response => response.json())
                  .catch(ErrorHandler.handleError)
  }

   menuOfRestaurant(id: string): Observable<MenuItem[]>{  //retornando um Observable<MenuItem[]>
    return this.http.get(`${MEAT_API}/restaurants/${id}/menu`)// Veja "/menu" no arquivo app.routes.ts
                  .map(response => response.json())
                  .catch(ErrorHandler.handleError)
  }
}
```

menu.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RestaurantsService } from '../../restaurants/restaurants.service';
import {Observable} from 'rxjs/Observable';
import { ActivatedRoute } from '@angular/router';
import {MenuItem} from '../menu-item/menu-item.model'

@Component({
  selector: 'mt-menu',
  templateUrl: './menu.component.html'
})
export class MenuComponent implements OnInit {

menu: Observable<MenuItem[]>
```

```
constructor(private restaurantsService: RestaurantsService, private route: ActivatedRoute) { }

  ngOnInit() {
              this.menu = this.restaurantsService.menuOfRestaurant(this.route.parent.snapshot.params['id'])
            //this.route.parent.snapshot.params => como o parametro "id" passado pertence ao componente pai (restaurante)
            //precisaremos do comando "parent"
  } //Agora vá no template menu-item.component.html e torne dinamico com um *ngFor

  addMenuItem(item: MenuItem){
            console.log(item)
            //console.log('====>'+ item)
  }
}
```

menu.component.html

```
<!--INÍCIO DO MENU -->
  <div class="col-md-9 col-xs-12">
     <mt-menu-item *ngFor="let item of menu | async"
     [menuItem]="item" (add)="addMenuItem($event)"></mt-menu-item>
  </div>
            <!-- $event => parametro que captura o evento e o objeto passado neste caso
             o menuItem ao clicar no link do componente menu-item.componente.html -->
  <div class="col-md-3 col-xs-12">
             <mt-shopping-cart></mt-shopping-cart>
  </div>
<!-- FIM DO MENU -->
```

Seção 5

Vídeo 54

Configurando a formatação de dados para o padrão brasileiro

1) Verificar se a dependência '**intl**' foi instalada no projeto. Veja o arquivo package.json.

1.1) package.json

```
{
 "name": "meat",
 "version": "1.0.0",
 "license": "MIT",
 "angular-cli": {},
 "scripts": {
   "ng": "ng",
   "start": "ng serve",
   "lint": "tslint \"src/**/*.ts\" --project src/tsconfig.json --type-check && tslint \"e2e/**/*.ts\" --project e2e/tsconfig.json --type-check",
   "test": "ng test",
   "pree2e": "webdriver-manager update --standalone false --gecko false",
   "e2e": "protractor"
 },
 "private": true,
 "dependencies": {
   "@angular/common": "4.0.0",
   "@angular/compiler": "4.0.0",
   "@angular/core": "4.0.0",
   "@angular/forms": "4.0.0",
   "@angular/http": "4.0.0",
   "@angular/platform-browser": "4.0.0",
   "@angular/platform-browser-dynamic": "4.0.0",
   "@angular/router": "4.0.0",
   "admin-lte": "2.3.11",
   "core-js": "2.4.1",
   "font-awesome": "4.7.0",
```

"intl": "1.2.5", //dependência para formatação de dados no padrão brasileiro

```
   "jquery": "3.1.1",
   "reflect-metadata": "0.1.9",
   "rxjs": "5.1.0",
   "ts-helpers": "1.1.2",
   "zone.js": "0.8.4"
 },
 "devDependencies": {
   "@angular/cli": "1.2.7",
   "@angular/compiler-cli": "4.0.0",
   "@types/jasmine": "2.5.38",
   "@types/node": "7.0.5",
   "codelyzer": "2.0.0",
   "jasmine-core": "2.5.2",
   "jasmine-spec-reporter": "3.2.0",
   "karma": "1.4.1",
   "karma-chrome-launcher": "2.0.0",
   "karma-cli": "1.0.1",
   "karma-jasmine": "1.1.0",
   "karma-remap-istanbul": "0.2.2",
   "protractor": "5.1.0",
   "ts-node": "2.1.0",
   "tslint": "4.5.0",
   "typescript": "2.2.0",
```

```
  "webdriver-manager": "10.2.5"
 }
}
```

2) polyfills.ts

```
// This file includes polyfills needed by Angular and is loaded before
// the app. You can add your own extra polyfills to this file.
import 'core-js/es6/symbol';
import 'core-js/es6/object';
import 'core-js/es6/function';
import 'core-js/es6/parse-int';
import 'core-js/es6/parse-float';
import 'core-js/es6/number';
import 'core-js/es6/math';
import 'core-js/es6/string';
import 'core-js/es6/date';
import 'core-js/es6/array';
import 'core-js/es6/regexp';
import 'core-js/es6/map';
import 'core-js/es6/set';
import 'core-js/es6/reflect';

import 'core-js/es7/reflect';
import 'zone.js/dist/zone';
```

import 'intl'; //aqui estamos incluindo a dependência do pacote 'intl' e na linha abaixo o script - feito pelo marques

import 'intl/locale-data/jsonp/pt-BR.js'; //script necessário para fazer a conversão para moeda brasileira - feito pelo marques

3) app.module.ts

```
……
…...
import { NgModule, LOCALE_ID } from '@angular/core'; //configurando a formatação para o padrão brasileiro
…..
@NgModule({
declarations: [
….
….
  ],
imports: [
….
….
  ],
  providers: [… , …. , { provide: LOCALE_ID, useValue: 'pt-BR' }], //declarando um padrão de locale na lista de providers
                                                                    //no módulo principal.
  bootstrap: [AppComponent]
})
```

4) Talvez seja necessário parar o **Webpack** e reinicializá-lo para que as configurações de formatação passem a fazer efeito.

Obs.: **Webpack**: é um empacotador que permite dividir seu código em múltiplos módulos para serem lidos sob demanda.

Seção 6

Vídeo 55

Template Forms

Template Forms é uma forma declarativa de configurar nossos formulários no template de um componente e para isso devemos usar a diretiva ngModel nos inputs os quais serão controlados pelo Framework Angular.

Sempre que declaramos um <form> em um template o Angular associa a diretiva ngForm de forma implícita. Com essa diretiva poderemos determinar a partir do form: **validade (valid/invalid), valor (true/false), dirty, pristine, touched e untouched.**

Obs.: Para utilizarmos a diretiva ngModel precisamos importar o pacote FormsModule.

//import { FormsModule } from '@angular/forms'; //utilizado na validacao de formulario

valid/invalid : informa se o valor do campo esta de acordo com as regras de validação.

pristine : informa o estado inicial do campo ou do form.

dirty : uma vez que o usuário entra no campo e digita algo o mesmo deixa de ser pristine e passa ser dirty.

untouched : quando o usuário apenas entra no campo, mas não digita nenhuma informação.

touched : quando o usuário apenas entra no campo e digita alguma informação.

EXTRA: * **NgModel** pode ser visto como uma representação de uma entidade que pode ser um atributo simples de um componente como uma String ou mesmo como um objeto mais complexo.

<form> ==========> < form ngForm >

< input type="text" name="nome" ngModel /> //o atributo name é obrigatório.

</form>

Uma vez utilizada a diretiva **ngModel** o formulário passa a ficar ciente do valor do campo, desta forma se o campo for valido ou invalido o formulário será valido ou invalido.

Uso do ngModel com One-Way-Binding

```
<form>      ==================================>      < form ngForm >
    < input type="text" name="nome" [ngModel]="username" /> //o atributo name é obrigatório.
</form>


@Component({ . . . })
export class UserComponent{
            username : string = "Nome do usuário"
}
```

//Perceba que apenas quando o valor do componente mudar o valor do campo será atualizado; o oposto não é verdadeiro.

Uso do ngModel com Two-Way-Binding

<form> ==========> < form ngForm >

< input type="text" name="nome" **[(ngModel)]**="username" /> //o atributo name é obrigatório.

</form>

@Component({ . . . })

export class UserComponent{

username : string = "Nome do usuário"

}

//Perceba se o valor do campo mudar a valor da propriedade no componente também mudará.

Obs.: Para utilizarmos a diretiva ngModel precisamos importar o pacote FormsModule no arquivo app.module.ts

//import { FormsModule } from '@angular/forms'; //utilizado na validacao de formulario

Como obter uma referência a diretiva ngForm através de uma Template Variable

<form #myForm="ngForm">

< input type="text" name="nome" /> //o atributo name é obrigatório.

< input type="text" name="ultimoNome" /> //o atributo name é obrigatório.

< input type="text" name="endereco" /> //o atributo name é obrigatório.

< button [disabled]="myForm.invalid /> //myForm => é uma Template Variable que referencia ngForm

</form> //ngForm => permite acessar propriedades do formulário como

//por exemplo habilitar ou não um botão baseado na validade

//dos campos, ou seja, os inputs do formulário.

Seção 6

Vídeo 57

Inserindo Formulário em um Componente

1) Importar o módulo **FormsModule** no arquivo app.module.ts

app.module.ts

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

import { NgModule, LOCALE_ID } from '@angular/core'; // NgModule ==> pacote necessário para utilizarmos a diretiva @NgModule

import { HttpModule } from '@angular/http';

import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

```
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderComponent } from './order/order.component';

import { FormsModule } from '@angular/forms'; //utilizado na validacao de formulario e na utilização da diretiva "ngModule"

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    OrderComponent
  ],
  imports: [
    BrowserModule,
    HttpModule,
    FormsModule, //utilizado na validacao de formulario
    RouterModule.forRoot(ROUTES)
  ],
  providers: [RestaurantsService,ShoppingCartService,{provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

2) Inserindo ngModel no formulário no template de um componente

order.component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>
```

```
<section class="content">
<section class="invoice">

                {{form.valid}}              //true / false ==> validade do formulário
                {{form.value | json}}       //{"address" : "Av. Geremario Dantas", "number": "2010", "optional" : ""} ==> debug dos inputs do formulario

 <form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
  <div class="row">
   <div class="col-xs-12">
    <h2 class="page-header">
     <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
   </div>
  </div>

  <div class="row invoice-info">
   <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
   </div>
   <div class="col-sm-6 col-xs-12">
    <div class="form-group">
     <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Endereço</label>
     <input type="text" class="form-control" name="address" ngModel placeholder="Endereço" >
     <span class="help-block"></span>
    </div>
   </div>
   <div class="col-sm-2 col-xs-6">
    <div class="form-group has-success">
     <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Número</label>
     <input type="text" class="form-control" name="number" ngModel placeholder="Número">
     <span class="help-block"><i class="fa fa-check"></i> Ok</span>
    </div>
   </div>
   <div class="col-sm-4 col-xs-6">
    <div class="form-group has-error">
     <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Complemento</label>
     <input type="text" class="form-control" name="optional" ngModel placeholder="Complemento">
     <span class="help-block"><i class="fa fa-remove"></i> Help block with error</span>
    </div>

   </div>
   <!-- /.col -->
  </div>
  <!-- /.row -->

  <!-- Table row -->
  <div class="row">
   <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
   </div>
   <div class="col-xs-12 table-responsive">
```

```
    <table class="table table-striped">
      <thead>
      <tr>
        <th class="text-center">Quantidade</th>
        <th>Item</th>
        <th>Descrição</th>
        <th class="text-right">Subtotal</th>
        <th class="text-right"></th>
      </tr>
      </thead>
      <tbody>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Cup Cake</td>

        <td>Recheado com doce de leite</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
        </td>
      </tr>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Croissant</td>

        <td>Recheado com presunto e queijo</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger "><i class="fa fa-remove"></i></a>
        </td>
      </tr>

      </tbody>
    </table>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->

<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>
```

```
        <div class="form-group">
          <div>
            <label>
              <div class="iradio_flat-red checked" aria-checked="false" aria-disabled="false"
                  style="position: relative;">
                  <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
                   <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
              </div>
              Dinheiro
            </label>
          </div>
         <div>
           <label>
             <div class="iradio_flat-red" aria-checked="false" aria-disabled="false"
                 style="position: relative;">
                 <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
                   <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
             </div>
             Cartão de Débito
           </label>
         </div>


        </div>


      </div>
      <!-- /.col -->
      <div class="col-sm-6 col-xs-12">
        <p class="lead">Frete e Total:</p>


        <div class="table-responsive">
          <table class="table">
            <tbody><tr>
              <th style="width:50%">Itens:</th>
              <td class="text-right">R$ 20,00</td>
            </tr>
            <tr>
              <th>Frete:</th>
              <td class="text-right">R$ 8,00</td>
            </tr>
            <tr>
              <th>Total:</th>
              <td class="text-right">R$ 28,00</td>
            </tr>
          </tbody></table>
        </div>
      </div>
      <!-- /.col -->
    </div>


  </form>
```

```
 <div class="row">
   <div class="col-xs-12">
     <a href="order-summary.html" class="btn btn-success pull-right"><i class="fa fa-credit-card"></i> Concluir Pedido
     </a>
   </div>
 </div>

</section>
</section>
<!-- FIM DO conteúdo-->
```

Seção 6

Vídeo 58

Verificando o estado de um campo

Para saber em qual estado um campo se encontra utilizamos a diretiva ngModel e relacionamos com uma template variable.

```
<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
    <input type="text"  name="nome" [ngModel]="username" #ipt="ngModel" >
     <span *ngIf="ipt.invalid">Nome inválido!</span>
</form>
```

Validators

required

pattern-regex

minlength

maxlength

Validação com Template Forms

```
<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
    <input type="text"  name="nome" required minlength="5" pattern="^[a-zA-Z]*$" #iptNome="ngModel" ngModel placeholder="Insira o seu nome..." autocomplete="off" >
     <span *ngIf="!iptNome.valid && (iptNome.dirty || iptNome.touched)" />
</form>
```

Seção 6

Vídeo 59

Aplicando Validação no Formulário de Compra

```
<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
    <input type="text"  name="nome" required minlength="5" pattern="^[a-zA-Z]*$" #iptNome="ngModel" ngModel placeholder="Insira o seu nome..." autocomplete="off" >
     <span *ngIf="!iptNome.valid && (iptNome.dirty || iptNome.touched)" />
    <button [disable]="!form.valid" />
</form>
```

Seção 6

Vídeo 60

Aplicando Feedback Visual no Formulario de Compra

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>
<section class="content">
<section class="invoice">
                {{form.valid}}
                {{form.value | json}}
<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
  </div>
</div>


<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
  <div class="form-group" [class.has-success]="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"  //classe css que deixa o campo verde
                          [class.has-error]="!iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"> //classe css que deixa o campo verde
     <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Endereço</label>
     <input type="text" class="form-control" name="address" required minlength="5" #iptAddress="ngModel" ngModel placeholder="Endereço" autocomplete="off" >
     <span class="help-block" *ngIf="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"> //classe css que emite uma mensagem de campo valido
      <i class="fa fa-check"></i> Ok!!!</span>
       <span class="help-block" *ngIf="!iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"> //classe css que deixa uma mensagem de campo invalido
      <i class="fa fa-remove"></i> Campo obrigatório!!!</span>
    </div>
  </div>
  <div class="col-sm-2 col-xs-6">
   <div class="form-group"  [class.has-success]="iptNumber.valid && (iptNumber.dirty || iptNumber.touched)"
```

```
                    [class.has-error]="!iptNumber.valid && (iptNumber.dirty || iptNumber.touched)">
    <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Número</label>
    <input type="text" class="form-control" name="number" required minlength="5" #iptNumber="ngModel" ngModel placeholder="Número">
     <span class="help-block" *ngIf="iptNumber.valid && (iptNumber.dirty || iptNumber.touched)">
     <i class="fa fa-check"></i> Ok!</span>
      <span class="help-block" *ngIf="!iptNumber.valid && (iptNumber.dirty || iptNumber.touched)">
     <i class="fa fa-remove"></i> Campo obrigatório!</span>
    </div>
  </div>
  <div class="col-sm-4 col-xs-6">
    <div class="form-group">
      <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Complemento</label>
      <input type="text" class="form-control" name="optional" ngModel placeholder="Complemento">
      <!--
      <span class="help-block"><i class="fa fa-remove"></i> Help block with error</span>
      -->
    </div>


  </div>
  <!-- /.col -->
</div>
<!-- /.row -->


<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>
  <div class="col-xs-12 table-responsive">


    <table class="table table-striped">
      <thead>
      <tr>
        <th class="text-center">Quantidade</th>
        <th>Item</th>
        <th>Descrição</th>
        <th class="text-right">Subtotal</th>
        <th class="text-right"></th>
      </tr>
      </thead>
      <tbody>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Cup Cake</td>


        <td>Recheado com doce de leite</td>
```

```
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
        </td>
      </tr>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Croissant</td>


        <td>Recheado com presunto e queijo</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger "><i class="fa fa-remove"></i></a>
        </td>
      </tr>


      </tbody>
    </table>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->


<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>


      <div class="form-group">
        <div>
          <label>
            <div class="iradio_flat-red checked" aria-checked="false" aria-disabled="false"
                style="position: relative;">
                <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
                  <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
            </div>
            Dinheiro
          </label>
        </div>
        <div>
          <label>
            <div class="iradio_flat-red" aria-checked="false" aria-disabled="false"
                style="position: relative;">
                <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
                <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
```

```
          </div>
          Cartão de Débito
        </label>
      </div>


    </div>


  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>


    <div class="table-responsive">
      <table class="table">
        <tbody><tr>
          <th style="width:50%">Itens:</th>
          <td class="text-right">R$ 20,00</td>
        </tr>
        <tr>
          <th>Frete:</th>
          <td class="text-right">R$ 8,00</td>
        </tr>
        <tr>
          <th>Total:</th>
          <td class="text-right">R$ 28,00</td>
        </tr>
      </tbody></table>
    </div>
  </div>
  <!-- /.col -->
</div>


</form>


<div class="row">
<div class="col-xs-12">
  <a href="order-summary.html" class="btn btn-success pull-right"><i class="fa fa-credit-card"></i> Concluir Pedido
  </a>
</div>
</div>
</section>
</section>
<!-- FIM DO conteúdo-→
```

Seção 6

Vídeo 61

Criando um Componente de Input (Content Projection)

Quando criamos um “componente customizado de input” o nosso formulário perde a rastreabilidade do componente não podendo ser rastreado através do “{{form.value | json}}”, isto ocorre por que o “input text” esta encapsulado dentro de outro componente, logo uma saída para este problema seria transformar nosso componente customizado em um componente container (Content Projection) onde apenas aplicaríamos um estilo visual ao componente e o componente parent passaria o “input text” ou então em um segundo cenário utilizaríamos a interface ControlValueAccessor que permitirá conectar o componente customizado com as diretivas especiais do formulário como NgModel ou as diretivas de React Form. Na técnica onde utilizamos Content Projection usaremos a tag “<ng-content></ng-content>”, esta tag será substituída automaticamente pelo input text passado pelo componente parent.

1) Crie em “src/app” uma pasta de nome shared

2) Navegue até a pasta src/app e crie um componente

3) > ng generate component input-customizado --spec=false

4) input-customizado.component.html

```
import { Component, OnInit, Input, ContentChild, AfterContentInit } from '@angular/core';
import {NgModel} from '@angular/forms';


@Component({
  selector: 'mt-input-customizado-container', //selector: 'mt-input-customizado',
  templateUrl: './input-customizado.component.html'
})
export class InputCustomizadoComponent implements OnInit, AfterContentInit {


 meuInputGenerico : any // esta variável está fazendo o papel de uma “template variable” que pega uma referência para o ngModel de um componente de input
 @Input() label : string
 @Input() errorMessage : string
 @ContentChild(NgModel) model: NgModel  //aqui estamos dizendo ao Angular qual diretiva queremos obter como referência, neste caso “NgModel”
  constructor() { }
  ngOnInit() {
  }
  ngAfterContentInit(){ // este método será chamado quando o conteúdo do componente “<ng-content></ng-content>” for definido com algum componente input
        this.meuInputGenerico = this.model
        if(this.meuInputGenerico === undefined){ //verifica se this.model que referência ngModel em um componente input foi declarado no componente input
                                                 //que fora passado pelo component parent
                    throw new Error('Esse componente precisa ser usado com uma diretiva ngModel')
```

```
            }
  }

  hasSuccess() : boolean {
            return this.meuInputGenerico.valid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }

  hasError() : boolean {
            return this.meuInputGenerico.invalid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }
}
```

5) **input-customizado.component.html**

```
<div class="form-group" [class.has-success]="hasSuccess()"
                        [class.has-error]="!meuInputGenerico.valid && (meuInputGenerico.dirty || meuInputGenerico.touched)">
<label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> {{label}}</label>

    <ng-content></ng-content  <!-- SIMILAR  A   <router-outlet></router-outlet> AQUI ENTRARA QUALQUER  COMPONENTE INPUT DINAMICAMENTE -->

<span class="help-block" *ngIf="hasSuccess()">
<i class="fa fa-check"></i> Ok!!!
</span>
<span class="help-block" *ngIf="hasError()">
<i class="fa fa-remove"></i> {{errorMessage}}
</span>
</div>
<!--
<div class="form-group" [class.has-success]="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"
          [class.has-error]="!meuInputGenerico.valid && (meuInputGenerico.dirty || iptAddress.touched)">
  <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Endereço</label>
  <input type="text" class="form-control" name="address" required minlength="5" #iptAddress="ngModel" ngModel placeholder="Endereço" autocomplete="off" >
  <span class="help-block" *ngIf="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)">
  <i class="fa fa-check"></i> Ok!!!
  </span>
   <span class="help-block" *ngIf="!iptAddress.valid && (iptAddress.dirty || iptAddress.touched)">
  <i class="fa fa-remove"></i> Campo obrigatório!!!
  </span>
</div>
→
```

6) Utilizando o componente generico em: order.componente.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>
```

```
<section class="content">
<section class="invoice">


                {{form.valid}}
                {{form.value | json}}


<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
  </div>
</div>


<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
                <input type="text" class="form-control" name="address" required minlength="5" ngModel placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
    <input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off" pattern="^[0-9]*$">
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" name="optional" ngModel placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->


<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>
  <div class="col-xs-12 table-responsive">


    <table class="table table-striped">
```

```
      <thead>
      <tr>
        <th class="text-center">Quantidade</th>
        <th>Item</th>
        <th>Descrição</th>
        <th class="text-right">Subtotal</th>
        <th class="text-right"></th>
      </tr>
      </thead>
      <tbody>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Cup Cake</td>


        <td>Recheado com doce de leite</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
        </td>
      </tr>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Croissant</td>


        <td>Recheado com presunto e queijo</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger "><i class="fa fa-remove"></i></a>
        </td>
      </tr>


      </tbody>
    </table>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->


<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>
```

```
    <div class="form-group">
      <div>
        <label>
          <div class="iradio_flat-red checked" aria-checked="false" aria-disabled="false" style="position: relative;">
            <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
                <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
          </div>
          Dinheiro
        </label>
      </div>
      <div>
        <label>
          <div class="iradio_flat-red" aria-checked="false" aria-disabled="false"
            style="position: relative;">
            <input type="radio" name="r3" class="flat-red" checked="" style="position: absolute; opacity: 0;"/>
            <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;"></ins>
          </div>
          Cartão de Débito
        </label>
      </div>


    </div>


  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>


    <div class="table-responsive">
      <table class="table">
        <tbody><tr>
          <th style="width:50%">Itens:</th>
          <td class="text-right">R$ 20,00</td>
        </tr>
        <tr>
          <th>Frete:</th>
          <td class="text-right">R$ 8,00</td>
        </tr>
        <tr>
          <th>Total:</th>
          <td class="text-right">R$ 28,00</td>
        </tr>
      </tbody></table>
    </div>
  </div>
  <!-- /.col -->
</div>
```

```
</form>


<div class="row">
<div class="col-xs-12">
  <a href="order-summary.html" class="btn btn-success pull-right"><i class="fa fa-credit-card"></i> Concluir Pedido
  </a>
</div>
</div>
</section>
</section>
<!-- FIM DO conteúdo-→
```

Seção 6

Vídeo 62

Validação com Expressões Regulares

<input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off" pattern="^[0-9]*$">

Expressão Regular que aceita apenas números: ^[0-9]*$

```
<div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
  <div class="form-group">
   <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
   <input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off" pattern="^[0-9]*$">
   </mt-input-customizado-container>
   </div>
 </div>
```

Seção 6

Vídeo 63 e 64

Criando Componente Usando ControlValueAccessor (Parte 1 e 2)

radio-option,model.ts

```
export class RadioOption {
        constructor(public label: string, public value: any){
        }
}
```

radio.component.html

```
<div *ngFor="let option of options">
<label>
```

```
  <div (click)="setValue(option.value)" class="iradio_flat-red" [class.checked]="option.value === value" aria-checked="false" aria-disabled="false" style="position: relative;">

 <ins class="iCheck-helper" style="position: absolute; top: 0%; left: 0%; display: block; width: 100%; height: 100%; margin: 0px; padding: 0px; background: rgb(255, 255, 255); border: 0px; opacity: 0;">

     </ins>

</div>

  {{option.label}}

</label>

</div>

<!-- class="iradio_flat-red checked" ==> checked eh uma classe css assim como iradio_flat-red →
```

radio.component.ts

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

import { RadioOption } from './radio-option.model';


import {NG_VALUE_ACCESSOR,ControlValueAccessor} from '@angular/forms';


@Component({

  selector: 'mt-radio',

  templateUrl: './radio.component.html',

  providers: [  // Aqui, precisamos registrar o nosso componente na lista providers junto com um provide chamado

     {            // NG_VALUE_ACCESSOR para ficar disponivel com as diretivas ngModel e reacting form (esta ultima será vista mais adiante)

                    provide : NG_VALUE_ACCESSOR,

                    useExisting: forwardRef(()=> RadioComponent), // forwardRef ( ) => funcao onde passamos uma referencia ao nosso componente  RadioComponent

                    multi: true

                  }

   ]  // Na realidade estamos registrando o nosso componente  RadioComponent como um  NG_VALUE_ACCESSOR

      // para ficar disponivel com as diretivas ngModel e reacting form

})

export class RadioComponent implements OnInit, ControlValueAccessor{


                @Input() options : RadioOption[]


                value : any


                onChange : any


                constructor() { }


                ngOnInit() {

                }


                // setValue(value : any){ //metodo do inicio do exericio parte 1

                //                this.value = value

                //                console.log(`Valor setado eh:${value}`)
```

```
// }

// Apenas com o evento no componente radio onde se utiliza o metodo setValue ja eh possivel
// capturar o valor que fora clicado mas ainda não se pode linkar o componente com o formulario
// a não ser que se utilize para este componente a interface ControlValueAcess.

setValue(value : any){ //metodo do exericio parte 2 – agora com a funcao onChange
        this.value = value
        //console.log(`Valor setado eh:${value}`)
        this.onChange(this.value) //chamando a funcao onChange(...) dentro do metodo setValue onde
                        //ocorre a mudanca de valor avisando assim as diretivas que o valor
                        //do componente mudou
        console.log(`Valor setado eh:${value}`)
}

// A Interface ControlValueAcess quando implementada em nosso componente vai dar poderes ao mesmo
// no sentido de interagir com as diretivas especias de formulario como ngModel ou
// as diretivas de reacting form que seráo vistas em exericios futuros.


// Implementando os metodos da Interface ControlValueAccess - sao quatro metodos mas na realidade
// precisamos apenas de dois para ter o que queremos funcionando no caso a linkagem com as diretivas
// do formulario

 /**
 * Write a new value to the element.
 */
writeValue(obj: any): void { // metodo que será chamado pela(s) diretiva(s) quando a(s) mesma(s)
                    // desejar(em) passar um valor para o seu componente
  this.value = obj
  console.log(`Utilizando o metodo writeValue para capturar o bojeto passado como parametro : ${this.value}`)
}

/**
 * Set the function to be called when the control receives a change event.
 */
registerOnChange(fn: any): void { //metodo onde se passa uma funcao neste caso "this.onChange"
                    //onde esta funcao será chamada sempre que o nosso valor
                    //interno do componente mudar, ou seja, teremos um novo valor
                    // e o nosso valor mudara em setValue
  this.onChange = fn
}

/**
 * Set the function to be called when the control receives a touch event.
 */
registerOnTouched(fn: any): void { // metodo que registra que o usuario entrou no seu componente
}
```

```
  /**
   * This function is called when the control status changes to or from "DISABLED".
   * Depending on the value, it will enable or disable the appropriate DOM element.
   *
   * @param isDisabled
   */
  setDisabledState?(isDisabled: boolean): void {
  }
}
```

order.component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>

<section class="content">
<section class="invoice">

                {{form.valid}}
                {{form.value | json}}

<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
  </div>
</div>

<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
    <input type="text" class="form-control" name="address" required minlength="5" ngModel placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
    <input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off" pattern="^[0-9]*$">
```

```
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" name="optional" ngModel placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->

<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>
  <div class="col-xs-12 table-responsive">

    <table class="table table-striped">
      <thead>
      <tr>
        <th class="text-center">Quantidade</th>
        <th>Item</th>
        <th>Descrição</th>
        <th class="text-right">Subtotal</th>
        <th class="text-right"></th>
      </tr>
      </thead>
      <tbody>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
        <td>Cup Cake</td>

        <td>Recheado com doce de leite</td>
        <td class="text-right">R$ 10,00</td>
        <td class="text-right">
          <a class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
        </td>
      </tr>
      <tr>
        <td class="text-center">
          <a class="btn btn-sm"><i class="fa fa-minus"></i></a>
          1
          <a class="btn btn-sm"><i class="fa fa-plus"></i></a>
        </td>
```

```
      <td>Croissant</td>

      <td>Recheado com presunto e queijo</td>
      <td class="text-right">R$ 10,00</td>
      <td class="text-right">
        <a class="btn btn-sm danger "><i class="fa fa-remove"></i></a>
      </td>
    </tr>

    </tbody>
  </table>
 </div>
 <!-- /.col -->
</div>
<!-- /.row -->

<div class="row">
 <!-- accepted payments column -->
 <div class="col-sm-6 col-xs-12">
  <p class="lead">Formas de Pagamento:</p>

    <div class="form-group">  <!-- componente radio button customizado -->
      <mt-radio [options]="paymentOptions" name="paymentOption" ngModel></mt-radio>
    </div>

 </div>
 <!-- /.col -->
 <div class="col-sm-6 col-xs-12">
  <p class="lead">Frete e Total:</p>

  <div class="table-responsive">
    <table class="table">
      <tbody><tr>
        <th style="width:50%">Itens:</th>
        <td class="text-right">R$ 20,00</td>
      </tr>
      <tr>
        <th>Frete:</th>
        <td class="text-right">R$ 8,00</td>
      </tr>
      <tr>
        <th>Total:</th>
        <td class="text-right">R$ 28,00</td>
      </tr>
    </tbody></table>
  </div>
 </div>
 <!-- /.col -->
</div>

</form>
```

```
<div class="row">
<div class="col-xs-12">
  <a href="order-summary.html" class="btn btn-success pull-right"><i class="fa fa-credit-card"></i> Concluir Pedido
  </a>
</div>
</div>

</section>
</section>
<!-- FIM DO conteúdo-->
```

order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  paymentOptions : RadioOption[] = [
      {label: 'Dinheiro', value: 'MON' },
      {label: 'Cartão de Dinheiro', value: 'DEB' },
      {label: 'Cartão Refeição', value: 'REF' }
  ]

  constructor() { }

  ngOnInit() {
  }

}
```

Seção 6

Vídeo 65 e 66

Implementando o Componente com os Itens da Compra

/order-items/order-items.component.ts

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

```
import {CartItem} from '../../restaurant-detail/shopping-cart/cart-item.model';

@Component({
  selector: 'mt-order-items',
  templateUrl: './order-items.component.html'
})
export class OrderItemsComponent implements OnInit {


 @Input() items : CartItem[]
 @Output() increaseQty = new EventEmitter<CartItem>(); //EventEmitter => suporta tipo generico
 @Output() decreaseQty = new EventEmitter<CartItem>();
 @Output() remove = new EventEmitter<CartItem>();

 constructor() { }

 ngOnInit() {
 }

 emitIncreaseQty(item: CartItem){//Este metodo deve ser inserido como um evento "click" no proprio
                                 //componente order-items.componente.html o que permitirá relancar
                                 //o objeto do tipo CartItem para o componente parent neste caso o
                                 //componente order.component.html para ser capturado atraves do $event
        this.increaseQty.emit(item)
 }

 emitDecreaseQty(item: CartItem){//Este metodo deve ser inserido como um evento "click" no proprio
                                 //componente order-items.componente.html o que permitirá relancar
                                 //o objeto do tipo CartItem para o componente parent neste caso o
                                 //componente order.component.html para ser capturado atraves do $event
        this.decreaseQty.emit(item)
 }

 emitRemove(item: CartItem){//Este metodo deve ser inserido como um evento "click" no proprio
                            //componente order-items.componente.html o que permitirá relancar
                            //o objeto do tipo CartItem para o componente parent neste caso o
                            //componente order.component.html para ser capturado atraves do $event
        this.remove.emit(item)
 }

  //Veja agora o template do seu componente order-items.componente.html pois os tres metodos:
  //emitIncreaseQty(item: CartItem), emitDecreaseQty(item: CartItem) e emitRemove(item: CartItem)
  //devem ser associados como um evento "click" no template deste componente.

}
```

/order-items/order-items.component.html

```
<div class="col-xs-12" *ngIf="!items">
            <p>
              Não há itens no seu carrinho. Que tal começar por <a [routerLink]="['/restaurants']">aqui</a>?
            </p>
</div>

<div class="col-xs-12 table-responsive" *ngIf="items">

  <table class="table table-striped">
    <thead>
    <tr>
      <th class="text-center">Quantidade</th>
      <th>Item</th>
      <th>Descrição</th>
      <th class="text-right">Subtotal</th>
      <th class="text-right"></th>
    </tr>
    </thead>
    <tbody>
    <tr *ngFor="let item of items"> <!-- iteracao sobre o array de CartItem[]
                        declarado em order-items.component.ts, perceba
                        que este array será alimentado pelo componente parent-->
      <td class="text-center">
        <a (click)="emitDecreaseQty(item)" class="btn btn-sm"><i class="fa fa-minus"></i></a>
        {{item.quantity}}
        <a (click)="emitIncreaseQty(item)" class="btn btn-sm"><i class="fa fa-plus"></i></a>
      </td>
      <td>{{item.menuItem.name}}</td>

      <td>{{item.menuItem.description}}</td>
      <td class="text-right">{{item.value() | currency: 'BRL': true}}</td>
      <td class="text-right">
        <a (click)="emitRemove(item)" class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
      </td>
    </tr>
    <tr>
    </tbody>
  </table>
 </div>
```

Fornecendo os Itens da Compra

/order/order.service.ts

```
import {Injectable} from  '@angular/core';
import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';
```

```
import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';

@Injectable() //injetando o servico ShoppingCartService
export class OrderService {
	constructor(private cartService :ShoppingCartService ){
}

	cartItems() : CartItem[]{
		return this.cartService.items
	}

	increaseQty(item : CartItem){
		return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
						//para ShoppingCartService, ou seja, de tras para frente,
						//pois quem já esta trabalhando com CartItem eh o servico
						//ShoppingCartService
	}

	decreaseQty(item : CartItem){
		return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
						//para ShoppingCartService, ou seja, de tras para frente,
						//pois quem já esta trabalhando com CartItem eh o servico
						//ShoppingCartService
	}

	remove(item : CartItem){
	this.cartService.removeItem(item)
  }
}
```

/shopping-cart/shopping-cart.service.ts

```
import {CartItem} from './cart-item.model'
import {MenuItem} from '../menu-item/menu-item.model'

export class ShoppingCartService {

	//items: CartItem[] //==>aqui nos so temos uma declaracao do array
	items: CartItem[] = [] //==> aqui podemos dizer que estamos inicializando com um array vazio

	clear(){
		this.items = [] //recebe um array vazio
	}

	addItem(item: MenuItem){
		let foudItem = this.items.find((mItem)=>mItem.menuItem.id === item.id)
		//(mItem)=>mItem.menuItem.id === item.id //verificando se ja existe um objeto no array atrave do id
```

```
		if(foudItem){
			foudItem.qtd = foudItem.qtd + 1  //incrementando a quantidade em 1 //foudItem.qtd = foudItem.qtd + 1
				//ou agora poderiamos fazer this.increaseQty(foudItem) no lugar de
				//foudItem.qtd = foudItem.qtd + 1
		}else{
			this.items.push(new CartItem(item))
		}

	}

	removeItem(item: CartItem){
		this.items.splice(this.items.indexOf(item),1)//removera o objeto CartItem a partir do indice 1
	}

	total(): number {
		return this.items.map(item => item.value()).reduce((prev, value) => prev+value,0)
		//map --> trabalha o parametro refatorando um item pegando apenas o valor neste caso price
		//reduce --> faz um somatorio incializando com 0
	}


	//================================= trabalhando com o componente order ================================//

	increaseQty(item : CartItem){
	   item.qtd = item.qtd + 1
	}

	decreaseQty(item : CartItem){
		item.qtd = item.qtd - 1

		if(item.qtd === 0){  // se a quantidade de itens chegar a zero o carItem devera
			 // ser removido da lista em order.component.html
			this.removeItem(item)
		}
	}
	//Os metodos increaseQty e decreaseQty criados em shopping-cart.service.ts devem ser expostos no OrderSevice
	//e depois seguem para que sejam encapsulados no componente order.componente.html uma vez que não podemos
	//chamar o servico direto no template de order.componente.html

}
```

/order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';
import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';
```

```
@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  paymentOptions : RadioOption[] = [
      {label: 'Dinheiro', value: 'MON' },
      {label: 'Cartão de Dinheiro', value: 'DEB' },
      {label: 'Cartão Refeição', value: 'REF' }
  ]

  constructor(private orderService : OrderService) { }

  ngOnInit() {
 }

  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQty(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQty(item : CartItem){
    this.orderService.decreaseQty(item)
  }

  remove(item : CartItem){
    this.orderService.remove(item)
  }
}
```

/order/order-component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>

<section class="content">
<section class="invoice">

                {{form.valid}}
                {{form.value | json}}
```

```
<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
  </div>
</div>

<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
    <input type="text" class="form-control" name="address" required minlength="5" ngModel placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
    <input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off" pattern="^[0-9]*$">
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" name="optional" ngModel placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->

<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>
```

<mt-order-items [items]="cartItems()"
(increaseQty)="increaseQtyOrder($event)"
(decreaseQty)="decreaseQtyOrder($event)"
(remove)="removeOrder($event)"></mt-order-items> //Observação: o evento customizado ficará
//no order-items.component.html

```
  <!-- /.col -->
</div>
<!-- /.row -->

<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>

      <div class="form-group">  <!-- componente radio button customizado -->
        <mt-radio [options]="paymentOptions" name="paymentOption" ngModel></mt-radio>
      </div>

  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>

    <div class="table-responsive">
      <table class="table">
        <tbody><tr>
          <th style="width:50%">Itens:</th>
          <td class="text-right">R$ 20,00</td>
        </tr>
        <tr>
          <th>Frete:</th>
          <td class="text-right">R$ 8,00</td>
        </tr>
        <tr>
          <th>Total:</th>
          <td class="text-right">R$ 28,00</td>
        </tr>
      </tbody></table>
    </div>
  </div>
  <!-- /.col -->
</div>

</form>

<div class="row">
<div class="col-xs-12">
  <a href="order-summary.html" class="btn btn-success pull-right"><i class="fa fa-credit-card"></i> Concluir Pedido
  </a>
</div>
</div>
```

```
</section>
</section>
<!-- FIM DO conteúdo -->
```

/order-items.component.html

```
<div class="col-xs-12" *ngIf="items.length === 0">
        <p>
          Não há itens no seu carrinho. Que tal começar por <a [routerLink]="['/restaurants']">aqui</a>?
        </p>
</div>

<div class="col-xs-12 table-responsive" *ngIf="items.length > 0">

  <table class="table table-striped">
   <thead>
   <tr>
    <th class="text-center">Quantidade</th>
    <th>Item</th>
    <th>Descrição</th>
    <th class="text-right">Subtotal</th>
    <th class="text-right"></th>
   </tr>
   </thead>
   <tbody>
   <tr *ngFor="let item of items"> <!-- iteracao sobre o array de CartItem[]
                    declarado em order-items.component.ts, perceba
                    que este array será alimentado pelo componente parent-->
    <td class="text-center">
     <a (click)="emitDecreaseQty(item)" class="btn btn-sm"><i class="fa fa-minus"></i></a>
     {{item.qtd}}
       <a (click)="emitIncreaseQty(item)" class="btn btn-sm"><i class="fa fa-plus"></i></a>
    </td>
    <td>{{item.menuItem.name}}</td>

    <td>{{item.menuItem.description}}</td>
    <td class="text-right">{{item.value() | currency: 'BRL': true}}</td>
    <td class="text-right">
     <a (click)="emitRemove(item)" class="btn btn-sm danger"><i class="fa fa-remove"></i></a>
    </td>
   </tr>
   <tr>
   </tbody>
  </table>
 </div>
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderComponent } from './order/order.component';

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formulario
import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';
import { RadioComponent } from './shared/radio/radio.component';
import { OrderItemsComponent } from './order/order-items/order-items.component';
```

import { OrderService } from './order/order.service';

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    OrderComponent,
    InputCustomizadoComponent,
    RadioComponent,
    OrderItemsComponent
  ],
  imports: [
    BrowserModule,
    HttpModule,
```

```
    FormsModule, //utilizado na validacao de formulario
    RouterModule.forRoot(ROUTES)
  ],
  providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Seção 6

Vídeo 68

68. Finalizando a Compra (HTTP POST)

Nesta seção implementaremos o método que envia os dados da compra para o nosso servidor back-end.

/order/order.service.ts

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

import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';

import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';


import { Http } from  '@angular/http';

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/map';

import {MEAT_API} from '../app.api';

import { Headers, RequestOptions } from  '@angular/http';
```

```
@Injectable() //injetando o servico ShoppingCartService
export class OrderService {

        constructor(private cartService: ShoppingCartService, private http: Http){

        }

        cartItems() : CartItem[]{
                return this.cartService.items
        }

        increaseQty(item : CartItem){
                return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
                                //para ShoppingCartService, ou seja, de tras para frente,
                                //pois quem já esta trabalhando com CartItem eh o servico
                                //ShoppingCartService
        }

        decreaseQty(item : CartItem){
                return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
                                //para ShoppingCartService, ou seja, de tras para frente,
                                //pois quem já esta trabalhando com CartItem eh o servico
                                //ShoppingCartService
        }

        remove(item : CartItem){
        this.cartService.removeItem(item)
}

itemsValue(): number {  //valor total da compra dos items
                return this.cartService.total()
}

clear(){
    this.cartService.clear()
}

//No metodo  checkOrder(...) estamos configurando o nosso Observable, perceba que a configuracao do Observable ocorre no servico
//enquanto que no componente, OrderComponent, utilizamos o subscribe para no inscrevermos no Observable
checkOrder(order: Order): Observable<string>{//quando o nosso objeto Order for enviando para o servidor
                                //o mesmo produzira um id do tipo string para o objeto Order,
                                //o qual representa a nossa compra.
                                //Neste metodo utilizaremso POST uma vez que estamos inserindo
                                //dados na nossa base de dado json.
```

```
                                        //`${MEAT_API}/orders` => local para onde nossos dados seráo salvos neste
                                        //caso em http://localhost:3000/orders ou veja o arquivo db.json
                                         //JSON.stringify(order) =>cria a representacao textual do objeto Order que
                                        //esta em texto devido ao Protocolo HTTP para uma respresentacao JSON.
                                        //headers.append("Content-Type","application/json") => indica o formato do dado
                                        //que será enviado.
                                        //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
                                        //que neste caso devido ao metodo POST será apenas o objeto Order que fora
                                        //salvo no nosso arquivo json.db
                                        //Para resumir quando usamos um objeto Http no servico estamos de fato
                                        //configurando o nosso objeto Observable
    const headers = new Headers()
    headers.append("Content-Type","application/json")

    return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
            .map(response=> response.json());
  }

}
```

/order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';

@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
     {label: 'Dinheiro', value: 'MON'},
     {label: 'Cartão de Débito', value: 'DEB'},
     {label: 'Cartão Refeição', value: 'REF'}
  ]

  constructor(private orderService : OrderService) { }
```

```
  ngOnInit() {

  }

  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQtyOrder(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQtyOrder(item : CartItem){
    this.orderService.decreaseQty(item)
  }

  removeOrder(item : CartItem){
    this.orderService.remove(item)
  }

  itemsValueOrder(): number { //valor total da compra dos items
    return this.orderService.itemsValue()
  }

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                              //A classe Order possui alem dos dados do endereco contera tambem um array
//                              //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                              //e o "id >> atributo contido em MenuItem".

                            //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                            //apenas a "quantidade >> atributo contido em CartItem" e o
                            //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                            //em MenuItem ja se encontram em nossa base de dados json

//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//    this.orderService.checkOrder(order) //==> neste metodo foi configurado um Observable. Veja a classe de servico deste metodo.
//      console.log(order)
// }
```

checkOrder(order: Order): void{ //metodo que finaliza a compra

//A classe Order possui alem dos dados do endereco contera tambem um array

//de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"

//e o "id >> atributo contido em MenuItem".

//Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira

//apenas a "quantidade >> atributo contido em CartItem" e o

//"id >> atributo contido em MenuItem" uma vez que os demais dados contidos

```
                    //em MenuItem ja se encontram em nossa base de dados json

  order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

  this.orderService.checkOrder(order) //==> neste metodo foi configurado um Observable. Veja a classe de servico deste metodo.
            .subscribe((orderId:string) => {
            console.log(`Compra concluída: ${orderId}`)
            })


  this.orderService.clear() //para limpar a compra depois de concluir
  console.log(order)
}

}
```

/order/order.component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>

<section class="content">
<section class="invoice">

            {{form.valid}}
            {{form.value | json}}

<form #form="ngForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i> Finalize o seu pedido
    </h2>
  </div>
</div>

<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
    <input type="text" class="form-control" name="address" required minlength="5" ngModel placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
  </div>
```

```
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
        <input type="text" class="form-control" name="number" required minlength="5" ngModel placeholder="Número" autocomplete="off"
pattern="^[0-9]*$">
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" name="optionalAddress" ngModel placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->

<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>
    <mt-order-items [items]="cartItems()"
                (increaseQty)="increaseQtyOrder($event)"
                (decreaseQty)="decreaseQtyOrder($event)"
                (remove)="removeOrder($event)"></mt-order-items>

  <!-- /.col -->
</div>
<!-- /.row -->
<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>

      <div class="form-group">  <!-- componente radio button customizado -->
        <mt-radio [options]="paymentOptions" name="paymentOption" ngModel required></mt-radio>
      </div>

  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>
```

```
  <mt-delivery-costs [delivery]="deliveryTax" [itemsValue]="itemsValueOrder()"></mt-delivery-costs>

  </div>
  <!-- /.col -->
</div>

</form>

<div class="row">
<div class="col-xs-12">
  <button (click)="checkOrder(form.value)" class="btn btn-success pull-right"
        [disabled]="!form.valid || cartItems().length === 0">
    <i class="fa fa-credit-card"> Concluir Pedido</i>
  </button>
</div>
</div>

</section>
</section>
<!-- FIM DO conteúdo-->
```

Console do browser:

{address: "R. Guilherme Veloso, 408", number: "4567890", optional: "fundos", paymentOption: "DEB"}

1. address: "R. Guilherme Veloso, 408"
2. number: "4567890"
3. optional: "fundos"
4. paymentOption: "DEB"
5. __proto__:
    1. constructor: *ƒ Object()*
    2. hasOwnProperty: *ƒ hasOwnProperty()*
    3. isPrototypeOf: *ƒ isPrototypeOf()*
    4. propertyIsEnumerable: *ƒ propertyIsEnumerable()*
    5. toLocaleString: *ƒ toLocaleString()*
    6. toString: *ƒ toString()*
    7. valueOf: *ƒ valueOf()*
    8. __defineGetter__: *ƒ __defineGetter__()*
    9. __defineSetter__: *ƒ _defineSetter__()*
    10. __lookupGetter__: *ƒ __lookupGetter__()*
    11. __lookupSetter__: *ƒ __lookupSetter__()*
    12. get __proto__: *ƒ __proto__()*
    13. set __proto__: *ƒ __proto__()*

/order/order.model.ts

```
class Order{
        constructor(  //perceba que este construtor tem os mesmos atributos contidos no formulário (veja o campo “name” além do “array” )
                      //Os dados que serão enviados podem ser visto no debugg da console do browser ao se clicar no
                      //botao "Concluir Pedido" localizado no template order.component.html
        public address: string,
        public number: number,
        public optionalAddress: string,
        public paymentOption: string,
        public orderItems: OrderItem[] = [] // possui a quantidade de CartItem e o id do MenuItem o qual na realidade é o nome do MenuItem,
        ){}
}

class OrderItem{
    constructor(public quantity: number, public menuId: string){} // possui a quantidade de CartItem e o id do MenuItem o qual
                                                                  // na realidade é o nome do MenuItem,
}
export {Order, OrderItem}
```

Seção 6

Vídeo 69

69. Navegacao Programatica via Router

/order/order.service.ts

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

import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';

import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';
```

```
import { Http } from  '@angular/http';

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/map';

import {MEAT_API} from '../app.api';

import { Headers, RequestOptions } from  '@angular/http';

@Injectable() //injetando o servico ShoppingCartService
export class OrderService {

  constructor(private cartService: ShoppingCartService, private http: Http){

  }

  cartItems() : CartItem[]{
    return this.cartService.items
  }

  increaseQty(item : CartItem){
    return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
                                              //para ShoppingCartService, ou seja, de tras para frente,
                                              //pois quem já esta trabalhando com CartItem eh o servico
                                              //ShoppingCartService
  }

  decreaseQty(item : CartItem){
    return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
                                              //para ShoppingCartService, ou seja, de tras para frente,
                                              //pois quem já esta trabalhando com CartItem eh o servico
                                              //ShoppingCartService
  }

  remove(item : CartItem){
     this.cartService.removeItem(item)
   }

   itemsValue(): number {  //valor total da compra dos items
      return this.cartService.total()
```

```
  }

  clear(){
    this.cartService.clear()
  }

  checkOrder(order: Order): Observable<string>{
                //quando o nosso objeto Order for enviando para o servidor
                 //o mesmo produzira um id do tipo string para o objeto Order,
                 //o qual representa a nossa compra.
                //Neste metodo utilizaremso POST uma vez que estamos inserindo
                //dados na nossa base de dado json.
                //`${MEAT_API}/orders` => local para onde nossos dados seráo salvos
                //JSON.stringify(order) =>cria a representacao textual do objeto Order que
                //esta em texto devido ao Protocolo HTTP para uma respresentacao JSON.
                //headers.append("Content-Type","application/json") => indica o formato do dado
                //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
                //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
                //que neste caso devido ao metodo POST será apenas o id do objeto Order
                //que fora salvo no nosso arquivo db.json
                //Para resumir quando usamos um objeto Http no servico estamos de fato
                //configurando o nosso objeto Observable
    const headers = new Headers()
    headers.append("Content-Type","application/json")//nome do header, valor

    return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
              .map(response=> response.json())
              .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                                      //para retornar apenas o ID da compra
  }

}
```

> ng generate component order-summary - -spec=false

/order-summary/order-summary.component.ts

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'mt-order-summary',
  templateUrl: './order-summary.component.html'
})
export class OrderSummaryComponent implements OnInit {
```

```
  constructor() { }

  ngOnInit() {
  }
}
```

/order-summary/order-summary.component.html

```
<!-- INICIO DO conteúdo -->
        <section class="content-header">
        </section>

        <section class="content">
        <div class="jumbotron">
          <h2>Pedido Conluído</h2>
          <p>Seu pedido foi recebido pelo restaurante. Prepare a mesa que a comida está chegando!</p>
          <p><b>Avalie</b> sua experiência:</p>
          <p><i class="fa fa-star"></i><i class="fa fa-star"></i><i class="fa fa-star"></i><i class="fa fa-star"></i><i class="fa fa-star"></i></p>
        </div>
        </section>
<!-- FIM DO conteúdo -->
```

/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
import { OrderComponent } from './order/order.component'
import { OrderSummaryComponent } from './order-summary/order-summary.component';

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        {path: 'about', component: AboutComponent},
        {path: 'restaurants', component: RestaurantsComponent},
        {path: 'order', component: OrderComponent},
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                    {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a página menu
                            {path: 'menu', component: MenuComponent},          //como padrão quando for especificado
                            {path: 'reviews', component: ReviewsComponent}   //na url apenas o id do restaurante.
                ]},                                                            //Veja: http://localhost:4200/restaurants/green-food
        {path: 'order-summary', component: OrderSummaryComponent}             //eh igual a http://localhost:4200/restaurants/green-food/menu
]                                                                              //devido ao redirectTo.
```

order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';

import {Router} from '@angular/router';

@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
      {label: 'Dinheiro', value: 'MON'},
      {label: 'Cartão de Débito', value: 'DEB'},
      {label: 'Cartão Refeição', value: 'REF'}
  ]

  constructor(private orderService : OrderService,
              private router : Router) { }

  ngOnInit() {

  }

  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQtyOrder(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQtyOrder(item : CartItem){
    this.orderService.decreaseQty(item)
  }
```

```
 removeOrder(item : CartItem){
  this.orderService.remove(item)
 }

 itemsValueOrder(): number { //valor total da compra dos items
  return this.orderService.itemsValue()
 }

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                          //A classe Order possui alem dos dados do endereco contera tambem um array
//                          //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                          //e o "id >> atributo contido em MenuItem".

                         //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                         //apenas a "quantidade >> atributo contido em CartItem" e o
                         //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                         //em MenuItem ja se encontram em nossa base de dados json

//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
// }

checkOrder(order: Order): void{ //metodo que finaliza a compra
                         //A classe Order possui alem dos dados do endereco contera tambem um array
                         //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                         //e o "id >> atributo contido em MenuItem".

                         //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                         //apenas a "quantidade >> atributo contido em CartItem" e o
                         //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                         //em MenuItem ja se encontram em nossa base de dados json

   order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

   this.orderService.checkOrder(order)
              .subscribe( (orderId:string) => {
                         this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                         console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                         this.orderService.clear() //para limpar a compra depois de concluir
              })
   console.log(order)
 }
}
```

Seção 6

Vídeo 70

Explicação sobre o componente: rating

O objetivo de criar este componente é para nos permitir fazermos avaliações. Por exemplo se clicarmos na estrela 3 a avaliação será três, se clicarmos na estrela 4 a avaliação será quatro e assim por diante. Este recurso já faz parte do css mais especificamente da biblioteca "Font Awesome" que estamos usando. Neste componente não utilizaremos ControlValueAccessor uma vez que não o utilizaremos dentro de um formulário.

70. Criando Um Componente de Rating

/shared/rating/rating.component.ts

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

@Component({
  selector: 'mt-rating',
  templateUrl: './rating.component.html'
})
export class RatingComponent implements OnInit {

   rates: number[] = [1,2,3,4,5]

   rate: number = 0 //este eh o valor que define a avaliacao na tela

   previousRate: number //este será um valor temporario que será utilizado
                        //na passagem do mouse sobre as estrelas. Inicialmente
                        //esta variavel, previousRate, estara undefined, mas quando
                        //passarmos o mouse sobre as estrelas o valor de rate será
                        //passado para previousRate e quando tirarmos o mouse das estrelas
                        //variavel previousRate passara novamente o valor original
                        //para a variavel rate.

  @Output() rated = new EventEmitter<number>() //evento criado para exibir uma mensagem ao usuario
                                  //agradecendo a avaliacao.Veja order-summary.component.html

  constructor() { }

  ngOnInit() {
  }

  setRate(rt: number){//ao clicar em uma estrela
            this.rate = rt
            console.log(">>>>>>>>>>setRate>>>>>>>>>>>>> rate:"+this.rate)
```

```
  this.previousRate = undefined
            console.log(">>>>>>>>>>setRate>>>>>>>>>>>>> rate:"+this.previousRate)
            this.rated.emit(this.rate) //linha utilizada para emitir um evento e exibir uma mensagem
 }

 setTemporaryRate(rt: number){//ao passar o mouse em uma estrela
            if(this.previousRate === undefined){
              this.previousRate = this.rate
              console.log(">>>>>>>>>>setTemporaryRate>>>>>>>>>>>>> previousRate:"+this.rate)
            }
            this.rate = rt
            console.log(">>>>>>>>>>setTemporaryRate>>>>>>>>>>>>> rate:"+this.rate)
 }

 clearTemporaryRate(){ //ao tirar o mouse de uma estrela
            if(this.previousRate != undefined){
              this.rate = this.previousRate
              this.previousRate = undefined
              console.log(">>>>>>>>>>clearTemporaryRate>>>>>>>>>>>>> rate:"+this.rate)
              console.log(">>>>>>>>>>clearTemporaryRate>>>>>>>>>>>>> previousRate:"+this.previousRate)
            }
 }

}
```

/shared/rating/rating.component.html

```
<i *ngFor="let rt of rates"
class="fa" [class.fa-star]="rt <= rate" [class.fa-star-o]="rt > rate"
(click)="setRate(rt)"
(mouseenter)="setTemporaryRate(rt)"
(mouseleave)="clearTemporaryRate()"></i>

<!-- Inicialmente rate vale zero. Veja em rating.component.ts -->
<!-- [class.fa-star] => estrela cheia. -->
<!-- [class.fa-star-o] => estrela vazada -->
```

```
<!-- cada class = "fa fa-star" eh uma classe css de uma estrela -->
<!--
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
<i class="fa fa-star"></i>
-->
```

```
<!-- Estamos usando Font Awesome em nossas classes de css. -->
<!-- Font Awesome é um conjunto de ferramentas de fontes e ícones com base em CSS e LESS. Foi feito por Dave Gandy para uso com o Twitter Bootstrap e mais tarde foi incorporado no BootstrapCDN - - >
```

/order-summary/order-summary.component.html

```
<!-- INICIO DO conteúdo -->
        <section class="content-header">
        </section>
        <section class="content">
        <div class="jumbotron">
          <h2>Pedido Conluído</h2>
          <p>Seu pedido foi recebido pelo restaurante. Prepare a mesa que a comida está chegando!</p>
          <p><b>Avalie</b> sua experiência:</p>
          <p><mt-rating (rated)="rated = true"></mt-rating></p>
          <p [hidden]="!rated">
                Obrigado pela sua avaliação!!!
          </p>
        </div>
        </section>
<!-- FIM DO conteúdo →
```

Seção 7

Vídeo 71

71. O que são Reactive Forms

Reactive Form é uma nova forma de implementar formulários sem a utilização da diretiva ngModel e o uso de validações nos templates dos componentes. Com Reactive Form serão criadas instâncias de FormGroup dentro de um componente.

Validators: required, minlength(...), maxlength(...) e pattern(...). Todos estes validadores são funções estáticas da classe Validators.

Exemplo de componente com FormGroup

```
@Component({...})
```

```
export class UserComponent implements OnInit {

        userForm: FormGroup //FormGroup permite agrupar um ou mais campos dentro dele

        constructor(private fb: FormBuilder) { }

        ngOnInit() {
          this.userForm = this.fb.group({
              username: this.fb.control(''),
              password: this.fb.control('')
        })
      }
  }
```

Exemplo de componente com FormGroup usando validadores

```
@Component({...})
export class UserComponent implements OnInit {

        userForm: FormGroup //FormGroup permite agrupar um ou mais campos dentro dele

        constructor(private fb: FormBuilder) { }

        ngOnInit() {
          this.userForm = this.fb.group({
            username: this.fb.control('', [Validators.required] ),      //neste array podemos inserir mais de um validador
            password: this.fb.control('', [Validators.minlength(3)] ) //neste array podemos inserir mais de um validador
        })
      }
  }
```

Exemplo de componente com campos que fazem sentido serem agrupados

```
@Component({...})
export class UserComponent implements OnInit {

        userForm: FormGroup //FormGroup permite agrupar um ou mais campos dentro dele

        constructor(private fb: FormBuilder) { }

        ngOnInit() {
          this.userForm = this.fb.group({
              username: '',
              password: ''
```

```
            address: this.fb.group({   //a vantagem de um "group" ocorre no fato de aplicarmos validadores
                                       //a nivel de grupo o que possibilita validar valores entre os componentes.
                street: '',
                zip: ''
            })
        })
    }
}
```

Exibição do Template

Perceba que com o uso de formGroup o nosso template esta muito mais simples e não utilizamos a diretiva ngModel.

```
<form [formGroup]="userForm">
  <input type="text" formControlName="username"/>
  <input type="password" formControlName="password"/>
</form>
```

Exibição do Template com Grupo

```
<form [formGroup]="userForm">
  <input type="text" formControlName="username"/>
  <input type="password" formControlName="password"/>

   <div formGroupName="address">
     <input type="text" formControlName="street/>
     <input type="text" formControlName="zip"/>
   </div>
</form>
```

Seção 7

Vídeo 72

72. Refatorando a Aplicação para Reactive Forms

Nesta seção mudaremos nossa implementação de **Template Forms**, o qual é mais compatível com Angular JS, para **Reactive Forms** o que nos permite ter mais controle sobre os nossos formulários uma vez que implementamos nossos componentes de forma programática. Com **Reactive Forms** utilizamos a diretiva **[formGroup]** nos templates dos componentes e cada campo será associado também com a diretiva **formControlName** e quando houver campos que fazem sentido serem agrupados usaremos a diretiva **formGroupName**, também serão utilizados em nossos arquivos TypeScript dos componentes as classes **FormGroup** que agrupa um ou mais campos de um formulário e a classe **FormBuilder** a qual ajuda na criação de campos e outros grupos além de permitir a

utilização de validadores nestes campos. É importante ressaltar que os campos criados programaticamente através da classe **FormGroup** serão inicializados dentro do método ngOnInit().

/order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';

import {Router} from '@angular/router';

import {FormGroup, FormBuilder} from '@angular/forms';

@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
      {label: 'Dinheiro', value: 'MON'},
      {label: 'Cartão de Débito', value: 'DEB'},
      {label: 'Cartão Refeição', value: 'REF'}
  ]

  orderForm: FormGroup  //<form [formGroup]="orderForm" novalidate>

  constructor(private orderService : OrderService,
            private router : Router,
            private formBuilder: FormBuilder) { }

  ngOnInit() {
     this.orderForm = this.formBuilder.group({//Aqui, estamos passando nossos campos para o FormBuilder
                                              //Com esta configuração já podemos linkar nossos campos da
                                              //página com as configuracoes do orderForm.

       name: this.formBuilder.control(''), //name: ''
```

```
      email: this.formBuilder.control(''),
      emailConfirmation: this.formBuilder.control(''),
      address: this.formBuilder.control(''),
      number: this.formBuilder.control(''),
      optionalAddress: this.formBuilder.control(''),
      paymentOption: this.formBuilder.control('')
    })
  }

  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQtyOrder(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQtyOrder(item : CartItem){
    this.orderService.decreaseQty(item)
  }

  removeOrder(item : CartItem){
    this.orderService.remove(item)
  }

  itemsValueOrder(): number { //valor total da compra dos items
    return this.orderService.itemsValue()
  }

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                                  //A classe Order possui alem dos dados do endereco contera tambem um array
//                                  //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                                  //e o "id >> atributo contido em MenuItem".

                                 //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                                 //apenas a "quantidade >> atributo contido em CartItem" e o
                                 //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                                 //em MenuItem ja se encontram em nossa base de dados json

//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
```

```
// }

checkOrder(order: Order): void{ //metodo que finaliza a compra
                        //A classe Order possui alem dos dados do endereco contera tambem um array
                        //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                        //e o "id >> atributo contido em MenuItem".

                        //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                        //apenas a "quantidade >> atributo contido em CartItem" e o
                        //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                        //em MenuItem ja se encontram em nossa base de dados json

   order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

   this.orderService.checkOrder(order)
               .subscribe( (orderId:string) => {
                        this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                        console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                        this.orderService.clear() //para limpar a compra depois de concluir
               })
   console.log(order)
 }

}
```

/shared/input-customizado/input-customizado.component.ts

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

//import {NgModel} from '@angular/forms';

import {FormControlName} from '@angular/forms';

@Component({
  selector: 'mt-input-customizado-container', //selector: 'mt-input-customizado-container',
  templateUrl: './input-customizado.component.html'
})
export class InputCustomizadoComponent implements OnInit, AfterContentInit {


 meuInputGenerico : any
```

```
 @Input() label : string
 @Input() errorMessage : string

// @ContentChild(NgModel) model: NgModel //==> Usado em Template Forms
 @ContentChild(FormControlName) control: FormControlName //==> Usado em Reactive Forms
  constructor() { }

  ngOnInit() {
  }

  ngAfterContentInit(){
    //this.meuInputGenerico = this.model || this.control  //==> tentar pegar uma das duas
            this.meuInputGenerico = this.control  //this.meuInputGenerico = this.model
            if(this.meuInputGenerico === undefined){
                    throw new Error('Esse componente precisa ser usado com uma diretiva FormControlName')
    //throw new Error('Esse componente precisa ser usado com uma diretiva NgModel')
    //throw new Error('Esse componente precisa ser usado com uma diretiva NgModel ou FormControlName')
              }
  }

  hasSuccess() : boolean {
            return this.meuInputGenerico.valid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }

  hasError() : boolean {
            return this.meuInputGenerico.invalid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }

}
```

/app/app.module.ts

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

```
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderComponent } from './order/order.component';

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';
import { RadioComponent } from './shared/radio/radio.component';
import { OrderItemsComponent } from './order/order-items/order-items.component';

import { OrderService } from './order/order.service';
import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
import { RatingComponent } from './shared/rating/rating.component';
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
```

```
    MenuItemComponent,
    ReviewsComponent,
    OrderComponent,
    InputCustomizadoComponent,
    RadioComponent,
    OrderItemsComponent,
    DeliveryCostsComponent,
    OrderSummaryComponent,
    RatingComponent
  ],
  imports: [
    BrowserModule,
    HttpModule,
    FormsModule, //utilizado na validacao de formularios com Template Forms
    ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms
    RouterModule.forRoot(ROUTES)
  ],
  providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

/order/order.component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>

<section class="content">
<section class="invoice">

            {{orderForm.valid}}
            {{orderForm.value | json}}

<form [formGroup]="orderForm" novalidate><!-- "orderForm" é uma referencia que esta localizada no arquivo order.componente.ts no componente OrderComponent -->
<!-- novalidate = delega o procedimento de validacao do form ao Angular -->
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i>Finalize o seu pedido
    </h2>
  </div>
</div>
```

```
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Seus dados:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Nome" >
    <input type="text" class="form-control" formControlName="name" placeholder="Nome" autocomplete="off">
    </mt-input-customizado-container>
  </div>

  <div class="col-sm-3 col-xs-6">
    <mt-input-customizado-container errorMessage="Email inválido!" label="E-mail" >
    <input type="text" class="form-control" formControlName="email" placeholder="E-mail" autocomplete="off">
    </mt-input-customizado-container>
  </div>

  <div class="col-sm-3 col-xs-6">
    <mt-input-customizado-container errorMessage="Email inválido!" label="Confirmação do e-mail" >
    <input type="text" class="form-control" formControlName="emailConfirmation" placeholder="Confirmação do e-mail" autocomplete="off">
    </mt-input-customizado-container>
  </div>
</div>

<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
    <input type="text" class="form-control" formControlName="address" placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
    <input type="text" class="form-control" formControlName="number" placeholder="Número" autocomplete="off">
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" formControlName="optionalAddress" placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
```

```
<!-- /.row -->

<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>


    <mt-order-items [items]="cartItems()"
                    (increaseQty)="increaseQtyOrder($event)"
                    (decreaseQty)="decreaseQtyOrder($event)"
                    (remove)="removeOrder($event)"></mt-order-items>

  <!-- /.col -->
</div>
<!-- /.row -->

<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>

      <div class="form-group">  <!-- componente radio button customizado -->
        <mt-radio [options]="paymentOptions" formControlName="paymentOption"></mt-radio>
      </div>

  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>

  <mt-delivery-costs [delivery]="deliveryTax" [itemsValue]="itemsValueOrder()"></mt-delivery-costs>

  </div>
  <!-- /.col -->
</div>

</form>

<div class="row">
<div class="col-xs-12">
  <button (click)="checkOrder(orderForm.value)" class="btn btn-success pull-right"
          [disabled]="!orderForm.valid || cartItems().length === 0">
    <i class="fa fa-credit-card"> Concluir Pedido</i>
  </button>
</div>
</div>
```

```
</section>
</section>
<!-- FIM DO conteúdo-→
```

Seção 7

Vídeo 73

73. Validação com Reactive Forms

/order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';

import {Router} from '@angular/router';

import {FormGroup, FormBuilder} from '@angular/forms';

import {Validators} from '@angular/forms';

@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

  emailPattern = /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i

  numberPattern = /^[0-9]*$/   //==> perceba que isto não eh uma string mas uma sintaxe de expressa regular




  deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
     {label: 'Dinheiro', value: 'MON'},
     {label: 'Cartão de Débito', value: 'DEB'},
     {label: 'Cartão Refeição', value: 'REF'}
  ]
```

```
orderForm: FormGroup
constructor(private orderService : OrderService,
        private router : Router,
        private formBuilder: FormBuilder) { }

ngOnInit() {
   this.orderForm = this.formBuilder.group({//Aqui, estamos passando nossos componentes para o FormBuilder
                                    //Com esta configuração já podemos linkar nossos componentes da
                                    //página com as configuracoes do orderForm.

     //name: this.formBuilder.control(''), //name: ''
     name: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
     email: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
     emailConfirmation: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
     address: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
     number: this.formBuilder.control('',[Validators.required,Validators.pattern(this.numberPattern)]),
     optionalAddress: this.formBuilder.control(''),
     paymentOption: this.formBuilder.control('',[Validators.required])
   })
}

cartItems() : CartItem[] {
  return this.orderService.cartItems()
}

increaseQtyOrder(item : CartItem){
  this.orderService.increaseQty(item)
}

decreaseQtyOrder(item : CartItem){
  this.orderService.decreaseQty(item)
}

removeOrder(item : CartItem){
  this.orderService.remove(item)
}

itemsValueOrder(): number { //valor total da compra dos items
  return this.orderService.itemsValue()
}

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                          //A classe Order possui alem dos dados do endereco contera tambem um array
//                          //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                          //e o "id >> atributo contido em MenuItem".
```

```
                    //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                    //apenas a "quantidade >> atributo contido em CartItem" e o
                    //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                    //em MenuItem ja se encontram em nossa base de dados json

//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
// }

checkOrder(order: Order): void{ //metodo que finaliza a compra
                    //A classe Order possui alem dos dados do endereco contera tambem um array
                    //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                    //e o "id >> atributo contido em MenuItem".

                    //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                    //apenas a "quantidade >> atributo contido em CartItem" e o
                    //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                    //em MenuItem ja se encontram em nossa base de dados json

   order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

   this.orderService.checkOrder(order)
              .subscribe( (orderId:string) => {
                    this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                    console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                    this.orderService.clear() //para limpar a compra depois de concluir
              })
       console.log(order)
    }
}
```

Seção 7

Vídeo 74

74. Criando Validadores Personalizados – com Reactive Forms

/order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';

import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';
```

```
import { Order, OrderItem } from './order.model';

import {Router} from '@angular/router';

import {FormGroup, FormBuilder} from '@angular/forms';

import {Validators} from '@angular/forms';

import {AbstractControl} from '@angular/forms';


@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {

 emailPattern = /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i

 numberPattern = /^[0-9]*$/   //==> perceba que isto não eh uma string mas uma sintaxe de expressa regular

 deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
     {label: 'Dinheiro', value: 'MON'},
     {label: 'Cartão de Débito', value: 'DEB'},
     {label: 'Cartão Refeição', value: 'REF'}
  ]

  orderForm: FormGroup  //Veja no template order.component.html, o metodo hasError('chave') eh um metodo default
                                //da classe FormGroup

constructor(private orderService : OrderService,
            private router : Router,
            private formBuilder: FormBuilder) { }

  ngOnInit() {
     this.orderForm = this.formBuilder.group({//Aqui, estamos passando nossos campos para o FormBuilder
                                  //Com esta configuração já podemos linkar nossos campos da
                                  //página com as configuracoes do orderForm.

       //name: this.formBuilder.control(''), //name: ''
       name: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
       email: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
       emailConfirmation: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
       address: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
       number: this.formBuilder.control('',[Validators.required,Validators.pattern(this.numberPattern)]),
       optionalAddress: this.formBuilder.control(''),
```

```
      paymentOption: this.formBuilder.control('',[Validators.required])
    }, {validator: OrderComponent.equalsTo}) //neste ponto estamos associando validadores ao nosso grupo (group)
  }     //validator ==> para um validador //validators ==> para mais de um validador

  //funcao
  //o retorno boolean será utilizado no template para dar um retorno visual
  static equalsTo(group: AbstractControl) : {[key: string] : boolean} {
    const emailZ = group.get('email') //pegando a referencia do campo a partir do group
    const emailConfirmationZ = group.get('emailConfirmation')

    if(!emailZ || !emailConfirmationZ){ //caso um dos campos(atributos) não existam no group
      return undefined
    }

    if(emailZ.value !== emailConfirmationZ.value){ //retornando uma chave(emailsNotMatch) com um valor
      return {emailsNotMatch: true}                //caso o valor seja true uma mensagem de erro será exibida
    }                                              //Perceba que a chave 'emailsNotMatch' será usada no template

    return undefined //caso o valor dos campos(atributos) sejam iguais, logo não exibo mensagem de erro

  }

  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQtyOrder(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQtyOrder(item : CartItem){
    this.orderService.decreaseQty(item)
  }

  removeOrder(item : CartItem){
    this.orderService.remove(item)
  }

  itemsValueOrder(): number { //valor total da compra dos items
    return this.orderService.itemsValue()
  }

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                              //A classe Order possui alem dos dados do endereco contera tambem um array
//                              //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
```

```
//                          //e o "id >> atributo contido em MenuItem".

                        //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                        //apenas a "quantidade >> atributo contido em CartItem" e o
                        //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                        //em MenuItem ja se encontram em nossa base de dados json

//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
// }

checkOrder(order: Order): void{ //metodo que finaliza a compra
                        //A classe Order possui alem dos dados do endereco contera tambem um array
                        //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                        //e o "id >> atributo contido em MenuItem".

                        //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                        //apenas a "quantidade >> atributo contido em CartItem" e o
                        //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                        //em MenuItem ja se encontram em nossa base de dados json

   order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

   this.orderService.checkOrder(order)
               .subscribe( (orderId:string) => {
                     this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                     console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                     this.orderService.clear() //para limpar a compra depois de concluir
               })

   console.log(order)
 }
}
```

/order/order.component.html

```
<!--INICIO DO conteúdo-->
<section class="content-header">
</section>

<section class="content">
<section class="invoice">

               {{orderForm.valid}}
               {{orderForm.value | json}}

<form [formGroup]="orderForm" novalidate><!-- novalidate = delega o procedimento de validacao do form ao Angular -->
```

```
<div class="row">
  <div class="col-xs-12">
    <h2 class="page-header">
      <i class="fa fa-shopping-cart"></i>Finalize o seu pedido
    </h2>
  </div>
</div>

<div class="row">
  <div class="col-xs-12 col-sm-9"> <!-- alteracoes no css para atender a validacao de email -->
    <p class="lead">Seus dados:</p>
  </div>

<div class="col-xs-12 col-sm-3"> <!-- alteracoes no css para atender a validacao de email -->
  <span class="help-block pull-right has-error-block" *ngIf="orderForm.hasError('emailsNotMatch')">
 <i class="fa fa-remove"></i>E-mails não conferem!</span>  <!-- o metodo hasError('chave') eh um metodo default da classe
                                                                    FormGroup. Perceba que orderForm eh uma instancia de FormGroup -->
  </div>


  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Nome" >
    <input type="text" class="form-control" formControlName="name" placeholder="Nome" autocomplete="off">
    </mt-input-customizado-container>
  </div>

  <div class="col-sm-3 col-xs-6">
    <mt-input-customizado-container errorMessage="Email inválido!" label="E-mail" >
    <input type="text" class="form-control" formControlName="email" placeholder="E-mail" autocomplete="off">
    </mt-input-customizado-container>
  </div>

  <div class="col-sm-3 col-xs-6">
    <mt-input-customizado-container errorMessage="Email inválido!" label="Confirmação do e-mail" >
    <input type="text" class="form-control" formControlName="emailConfirmation" placeholder="Confirmação do e-mail" autocomplete="off">
    </mt-input-customizado-container>
  </div>
</div>

<div class="row invoice-info">
  <div class="col-xs-12">
    <p class="lead">Endereço de Entrega:</p>
  </div>
  <div class="col-sm-6 col-xs-12">
    <mt-input-customizado-container errorMessage="Campo obrigatório e com no mínimo 5 caracteres!" label="Endereço" >
    <input type="text" class="form-control" formControlName="address" placeholder="Endereço" autocomplete="off">
    </mt-input-customizado-container>
```

```
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-2 col-xs-6" -->
   <div class="form-group">
    <mt-input-customizado-container errorMessage="Obrigatório e somente números!" label="Número">
    <input type="text" class="form-control" formControlName="number" placeholder="Número" autocomplete="off">
    </mt-input-customizado-container>
    </div>
  </div>
  <div class="col-sm-3 col-xs-6"><!-- class="col-sm-4 col-xs-6" -->
  <div class="form-group">
    <mt-input-customizado-container>
    <input type="text" class="form-control" formControlName="optionalAddress" placeholder="Complemento">
    </mt-input-customizado-container>
  </div>
  </div>
  <!-- /.col -->
</div>
<!-- /.row -->

<!-- Table row -->
<div class="row">
  <div class="col-xs-12">
    <p class="lead">Itens do Pedido:</p>
  </div>


    <mt-order-items [items]="cartItems()"
                    (increaseQty)="increaseQtyOrder($event)"
                    (decreaseQty)="decreaseQtyOrder($event)"
                    (remove)="removeOrder($event)"></mt-order-items>

  <!-- /.col -->
</div>
<!-- /.row -->

<div class="row">
  <!-- accepted payments column -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Formas de Pagamento:</p>

      <div class="form-group">  <!-- componente radio button customizado -->
        <mt-radio [options]="paymentOptions" formControlName="paymentOption"></mt-radio>
      </div>

  </div>
  <!-- /.col -->
  <div class="col-sm-6 col-xs-12">
    <p class="lead">Frete e Total:</p>
```

```
  <mt-delivery-costs [delivery]="deliveryTax" [itemsValue]="itemsValueOrder()"></mt-delivery-costs>


  </div>
  <!-- /.col -->
</div>
</form>


<div class="row">
<div class="col-xs-12">
  <button (click)="checkOrder(orderForm.value)" class="btn btn-success pull-right"
        [disabled]="!orderForm.valid || cartItems().length === 0">
    <i class="fa fa-credit-card"> Concluir Pedido</i>
  </button>
</div>
</div>
</section>
</section>
<!-- FIM DO conteúdo-->
```

Seção 8

Vídeo 75

75. O que são Módulos

Módulos em Angular são itens de configuração do framework que servem para agregar componentes, diretivas, serviços ou pipes, mas não apenas com o objetivo de organizar o seu código. Com a utilização de módulos você poder decidir o que deverá ser exportado a partir do seu modulo e como a injeção de dependência será configurada. Outro ponto interessante é o Lazy Loading, ou seja, carregamento tardio esse recurso permite que sua aplicação seja carregada em partes de acordo o acesso dos usuários, ou seja, sob demanda.

Tipos de Módulos

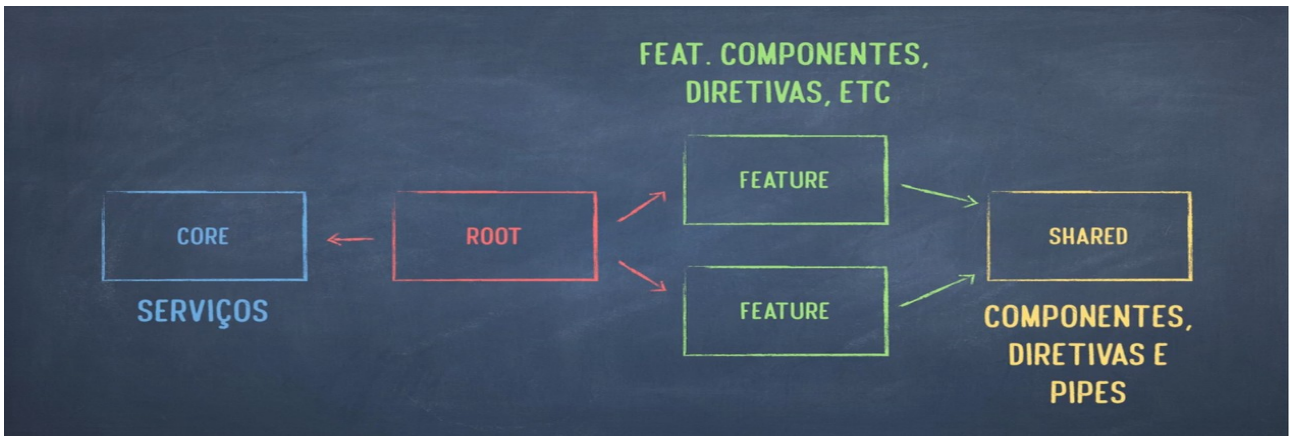
Root Module (Modulo Raiz): possui a **declaração** dos componentes que serão usadas na aplicação inteira, este modulo e responsável por fazer o bootstrap da aplicação. Quando a aplicação é de pequeno porte não há problemas em termos todos os recursos em um mesmo modulo, porém para aplicações maiores pode ser necessário organizar os componentes em outros módulos.

Shared Module (Modulo Compartilhado): neste tipo de módulo os componentes declarados serão compartilhados por outros módulos da aplicação.

Core Module (Modulo Núcleo): neste tipo de módulo teremos apenas os serviços da aplicação, neste modelo de modulo não teremos componentes. Exemplo de Core Module: HttpModule (from '@angular/http') do Angular, neste módulo temos apenas serviços.

// HttpModule.class => https://github.com/angular/angular/blob/master/packages/http/src/http_module.ts

Feature Module (Modulo de Recurso): módulo que contém as implementações das funcionalidades da sua aplicação.



Exemplo de Declaração de Módulo

```
@NgModule({  //@NgModule => decorate que define que um módulo
  declarations: [ … ],
  imports: [ … ],
  providers: [ … ],
  exports: [ … ]   //==> componentes que serão visíveis de fora do módulo, contudo sem a utilização do exports
                   //os componentes criados neste módulo serão visíveis apenas internamente.
})
export class MyModule { }  //Perceba que apenas o módulo principal possui a propriedade "bootstrap": [...]".
```

Seção 8

Vídeo 76

76. Carregamento Tardio de Modulos (Lazy-Loading)

O Lazy Loading é controlado pelo Routes o qual é nosso o modulo de roteamento, veja o arquivo app.modulo.ts e apenas no momento em que a rota que contém o módulo é acionada teremos o carregamento do Modulo AboutModule.

Nesta seção criaremos o módulo AboutModule que ficará responsável pela seção "Sobre" da aplicação que conterá apenas o componente AboutComponent.

/about/about.module.ts [Este é um exemplo de Feature Modulo]

Obs.: Toda vez que criarmos um módulo que carregue sob demanda, ou seja, um Módulo Lazy Loading teremos obrigatoriamente que definir também as rotas internas deste Módulo Lazy

Loading além de definir qual será o componente padrão quando o módulo for carregado; poderemos notar isso através do {path: ‘’}. Lembre-se também que este tipo de módulo não é importado no módulo raiz, mas apenas no Módulo de Rotas do componente principal; neste caso veja o arquivo ***app.routing.ts*** . Veja abaixo:

```
import {NgModule} from '@angular/core';
import {AboutComponent} from './about.component';
import {RouterModule, Routes} from '@angular/router';
export const ROUTES: Routes = [
        {path: '', component: AboutComponent} //Com [path: ''] estamos informando qual eh o componente padrão quando
                                                //este módulo for carregado.
]

@NgModule({  //@NgModule => decorate que define que esta classe eh um módulo
  declarations: [AboutComponent],  //=> lista todos os componentes que devem fazer parte deste modulo
  imports: [RouterModule.forChild(ROUTES)] //=> importando a rota
 //providers: [ … ],//=> nao foi utilizado uma vez que este componente nao usa nenhum tipo de servico
 //exports: [ … ],  //=> componentes que serão visíveis de fora, contudo sem a utilização do exports
            //os componentes criados neste módulo serão visíveis apenas internamente.
 //bootstrap: [...] //=> utilizado apenas no modulo principal
})
export class AboutModule {  //classe que reprensentara o modulo About
}
```

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component'; //comentado devido ao componente fazer
                                                          //parte de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
```

```
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderComponent } from './order/order.component';

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';
import { RadioComponent } from './shared/radio/radio.component';
import { OrderItemsComponent } from './order/order-items/order-items.component';
import { OrderService } from './order/order.service';
import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
import { RatingComponent } from './shared/rating/rating.component';
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    OrderComponent,
    InputCustomizadoComponent,
    RadioComponent,
    OrderItemsComponent,
    DeliveryCostsComponent,
    OrderSummaryComponent,
    RatingComponent
  ],
  imports: [
    BrowserModule,
    HttpModule,
    FormsModule, //utilizado na validacao de formularios com Template Forms
    ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicialização. Por fim, vale comentar que utilizar o
    //modulo aqui, seria o mesma coisa que inserir o componente AboutComponent em  “declarations”, pois o modulo seria carregado logo na
    //inicialização mas se utilizarmos o loadChildren no arquivo de rotas chamando o modulo AboutModule  neste caso o carregamento seria sobre
    //demanda, ou seja, seria carregado apenas quando o componente do modulo AboutModule fosse chamado.
```

```
  RouterModule.forRoot(ROUTES) //importa as rotas através do metodo forRoot uma vez que este modulo é o modulo raiz
  //(root ) em outros modulos devemos usar RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
 ],
 providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de
                                                                                                          //providers com escopo para este modulo.
 bootstrap: [AppComponent]
})
export class AppModule { }
```

/app/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
import {OrderComponent} from './order/order.component'
import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        //{path: 'about', component: AboutComponent},//esta rota não será mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule – rota responsavel pelo Lazy
        {path: 'restaurants', component: RestaurantsComponent}, // loadChildren => carregará o modulo sobre demanda.
        {path: 'order', component: OrderComponent},
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                  {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a página menu
                        {path: 'menu', component: MenuComponent},        //como padrão quando for especificado
                        {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.
                ]},                                  //Veja: http://localhost:4200/restaurants/green-food
        {path: 'order-summary', component: OrderSummaryComponent}
]                                       //eh igual a http://localhost:4200/restaurants/green-food/menu
                                        //devido ao redirectTo.
```

Seção 8

Vídeo 77

77. Criando Um Modulo Compartilhado

Este módulo será apenas compartilhado com outros módulos, não teremos navegação.

/shared/shared.module.ts

```
import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'

import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                    //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)


@NgModule({
  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent],//componentes que seráo utilizados neste modulo
  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que será importado como dependencia deste modulo
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
                RadioComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
               RatingComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
                  FormsModule,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
           ReactiveFormsModule,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
                  CommonModule//fazendo o export desta diretiva não precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que seráo exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado

//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para não ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
```

```
//de enxugar os outros modulos.
//Perceba que neste modulo não teremos navegacao.

}
```

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer  parte  de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderComponent } from './order/order.component';

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                                                        //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule, que contém este componente
import { OrderItemsComponent } from './order/order-items/order-items.component';

import { OrderService } from './order/order.service';
import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule, que contém este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import {SharedModule} from './shared/shared.module' //modulo customizado por mim
```

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    OrderComponent,
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    OrderItemsComponent,
    DeliveryCostsComponent,
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    HttpModule,
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                                  //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    SharedModule //este modulo customizado contem componentes criados por mim, alem de exportar diretivas do angular; não
                 //sendo necessario portanto fazer imports novamente neste modulo.
  ],
  providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um
                                                                    //servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Seção 8

Vídeo 78

78. Criando um Feature Module (Modulo de Compra)

/order/order.module.ts

```
import {NgModule} from '@angular/core';

import {OrderComponent} from './order.component';
import {OrderItemsComponent} from '../order-items/order-items.component';
import {DeliveryCostsComponent} from './delivery-costs/delivery-costs.component';

import {SharedModule} from '../shared/shared.module'; //estamos importando um modulo compartilhado criado por mim

import {RouterModule, Routes} from '@angular/router';

const ROUTES : Routes = [
        {path: '', component: OrderComponent} //caminho padrão que carregara o componente principal
]

@NgModule({//define esta classe como um modulo
        declarations : [OrderComponent, OrderItemsComponent, DeliveryCostsComponent], //componentes usados neste modulo
        imports : [SharedModule, RouterModule.forChild(ROUTES)], //estamos importando um modulo compartilhado criado por mim
  //imports : [SharedModule,RouterModule.forChild(ROUTES)] => retorna o modulo ja com as rotas além dos recursos de RouterModule. ou seja, não é necessário //importar RouterModule novamente.
  //RouterModule.forRoot(ROUTES) => utilizado apenas no modulo raiz
  //exports : [] //exportamos apenas os componentes que seráo usados fora deste modulo
})
export class OrderModule{
 //Depois de aprontarmos o nosso modulo devemos configura-lo
 //como um lazy loading no arquivo app.routes.ts e no arquivo
 //app.module.ts teremos que remover as dependencias, componentes,
 //que ja sao carregadas em outros modulos.
}
```

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

import { RestaurantsService } from './restaurants/restaurants.service';
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
```

```
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                                                    //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                                                    //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                    //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                                                    //de um novo modulo OrderModule

import { OrderService } from './order/order.service';

import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                    //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import {SharedModule} from './shared/shared.module' //modulo customizado por mim

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
```

```
  imports: [
    BrowserModule,
    HttpModule,
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
    //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    SharedModule //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular não
            //sendo necessario portanto fazer imports novamente neste modulo.

    //OrderModule => este modulo não foi inserido aqui, pois não queremos
    //que o mesmo seja carregado na inicializacao do projeto, porém faremos
    //este carregamento como lazy loading no app.routes.ts
  ],
   providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com
                                                                                                                                    //escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

/app/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
//import {OrderComponent} from './order/order.component'
import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        //{path: 'about', component: AboutComponent},//esta rota não será mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule – rota responsavel pelo Lazy Loading
        {path: 'restaurants', component: RestaurantsComponent},
        //{path: 'order', component: OrderComponent},//esta rota não será mais usada uma vez que criamos o OrderModule
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                  {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a página menu
                        {path: 'menu', component: MenuComponent},        //como padrão quando for especificado
                        {path: 'reviews', component: ReviewsComponent}    //na url apenas o id do restaurante.
                ]},                                  //Veja: http://localhost:4200/restaurants/green-food
                                                     //eh igual a http://localhost:4200/restaurants/green-food/menu
```

```
                                        //devido ao redirectTo.
  {path: 'order-summary', component: OrderSummaryComponent},
  {path: 'order', loadChildren: './order/order.module#OrderModule'}//rota para o Modulo OrderModule
]
```

Seção 8

Vídeo 79

79. Criando um Core Module Com Servicos

Providers e Módulos:

Sempre que declaramos um provider na lista de providers de um componente limitaremos a instância daquele provider, ou seja, de um serviço e disponibilizando-o para um componente e seus filhos sempre que precisarem.

Cada componente terá o seu contexto de injeção de dependência. O contexto global de injeção de dependência é o maior contexto da hierarquia e é obtido a partir do modulo root da aplicação, o que o torna disponível inclusive para os outros componentes do modulo root bem como outros módulos. Nesta aplicação o nosso modulo root chama-se AppModule e pode ser encontrado no arquivo app.module.ts. Os serviços declarados na lista de providers do modulo AppModule terão uma instância Singleton o qual será compartilhada por todos os outros componentes e módulos da aplicação. Por padrão um modulo, exceto o modulo root, não tem um contexto de injeção de dependência como tem um componente, porém tudo dependerá de como o modulo será carregado que poderá ser Lazy (forma tardia) ou Eager (na inicialização da aplicação). Se o modulo criado for declarado na lista de imports do modulo principal então todo serviço declarado na lista de providers do módulo terá uma instância global, ou seja, Singleton em outras palavras adicionaremos de certa forma mais serviços na lista de providers do modulo root, por assim dizer, contudo se o modulo criado for carregado de forma Lazy os providers declarados terão um contexto de injeção de dependência isolado do restante da aplicação.

A idéia de um Core Module é evitar uma mudança de comportamento caso seja trocado o módulo de carregamento de Eager (junto com o modulo raiz) para Lazy. Com a criação de um Core Module teremos todos os providers, nossos serviços, declarados neste modulo e o carregamento dos serviços dependerá de como o Core Module foi declarado na aplicação.

/core/core.module.ts

```
import {NgModule} from '@angular/core';

import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';

@NgModule({
  providers: [RestaurantsService, ShoppingCartService, OrderService] //declarando servicos na lista de providers
})
export class CoreModule {//modulo do tipo: modulo core
                                //Para esta aplicação este modulo será importado no modulo root
}
```

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                                //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente
```

```
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms


import { SharedModule } from './shared/shared.module' //modulo customizado por mim


import { CoreModule } from './core/core.module'


@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    HttpModule, //==> exemplo de Core Module. Veja o github do Angular
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
    //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular não
                  //sendo necessario portanto fazer imports novamente neste modulo.


    //OrderModule => este modulo não foi inserido aqui, pois não queremos
    //que o mesmo seja carregado na inicializacao do projeto, porém faremos
    //este carregamento como lazy loading no app.routes.ts


    CoreModule  //importando um modulo core
  ],



   providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],


  //transferido para CoreModule
   //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
```

```
export class AppModule { }
```

Seção 8

Vídeo 80

80. Adicionando Serviços a um Módulo Compartilhado

Podemos substituir um Core Modulo por um Modulo Compartilhando bastando apenas criarmos uma função static a qual será capaz de retornar os nossos serviços que foram declarados no Modulo Compartilhando. Com isso podemos excluir completamente o Core Modulo da aplicação.

/core/core.module.ts (módulo removido)

```
//import {NgModule} from '@angular/core';

//import { RestaurantsService } from '../restaurants/restaurants.service';
//import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
//import { OrderService } from '../order/order.service';

//@NgModule({
//  providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//})
//export class CoreModule {//modulo do tipo: modulo core
                //Para esta aplicação este módulo será importado no módulo root
//}
//////////// Este módulo não é mais usado uma vez que utilizamos o Modulo Compartilhado
```

/shared/shared.module.ts

```
import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'
```

```
import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)


import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                //permitirá criar uma funcao capaz de retornar os
                                //nosso servicos

import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';

@NgModule({
  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent],//componentes que seráo utilizados neste modulo
  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que será importado como dependencia deste modulo
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
              RadioComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
             RatingComponent,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
                FormsModule,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
           ReactiveFormsModule,//fazendo o export desta diretiva não precisarei importa-la em outro modulo
                CommonModule//fazendo o export desta diretiva não precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que seráo exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para não ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
//Perceba que neste modulo não teremos navegacao.

static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou não os serviços alem dos componentes
                                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo app.module.ts
  return {
        ngModule : SharedModule,
        providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
       }
 }

//FUNCAO ORIGINAL

//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou não os serviços alem dos componentes
                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                //app.module.ts
```

```
//   return {
//            ngModule : SharedModule,
//            providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//          }
//  }

}
```

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                        //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
```

```
                                                    //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                    //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                    //uma vez que mudamos os providers para o SharedModule
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
     OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    HttpModule, //==> exemplo de Core Module. Veja o github do Angular
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                            //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular não
                  //sendo necessario portanto fazer imports novamente neste modulo.

    SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou não.Neste caso
                                //estamos importando. Veja o Modulo OrderModule onde não importamos os
```

```
                    //serviços apenas os componentes declarados no SharedModule.
    //OrderModule => este modulo não foi inserido aqui, pois não queremos
    //que o mesmo seja carregado na inicializacao do projeto, porém faremos
    //este carregamento como lazy loading no app.routes.ts

    //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
  ],

   providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],
  //transferido para CoreModule
   //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

/order/order.module.ts

```
import {NgModule} from '@angular/core';

import {OrderComponent} from './order.component';
import {OrderItemsComponent} from './order-items/order-items.component';
import {DeliveryCostsComponent} from './delivery-costs/delivery-costs.component';

import {SharedModule} from '../shared/shared.module'; //estamos importando um modulo compartilhado criado por mim

import {RouterModule, Routes} from '@angular/router';

const ROUTES : Routes = [
        {path: '', component: OrderComponent} //caminho padrão que carregara o componente principal
]

@NgModule({//define esta classe como um modulo
        declarations : [OrderComponent, OrderItemsComponent, DeliveryCostsComponent], //componentes usados neste modulo
        imports : [SharedModule, RouterModule.forChild(ROUTES)], //estamos importando um modulo compartilhado criado por mim
    //imports : [SharedModule,RouterModule.forChild(ROUTES)] => retorna o modulo ja com as rotas
    //RouterModule.forRoot(ROUTES) => utilizado apenas no modulo raiz
    //exports : [] //exportamos apenas os componentes que seráo usados fora deste modulo
})
export class OrderModule{
  //Depois de aprontarmos o nosso modulo devemos configura-lo
  //como um lazy loading no arquivo app.routes.ts e no arquivo
  //app.module.ts teremos que remover as dependencias, componentes,
  //que ja sao carregadas em outros modulos.
}
```

Seção 8

Vídeo 81

81. Realizando o Pre-Carregamento de Todos os Módulos

Realizando carregamento em background dos módulos lazy após os módulos eager serem carregados na inicialização.

/app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                        //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                        //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                        //de um novo modulo OrderModule
```

```
//import { OrderService } from './order/order.service';//transferido para CoreModule


import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms


import { SharedModule } from './shared/shared.module' //modulo customizado por mim


//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule


import { PreloadAllModules } from '@angular/router'//Import necessario para fazer o carregamento em background dos modulos Lazy


@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    HttpModule, //==> exemplo de Core Module. Veja o github do Angular
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
    //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                                //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
```

```
  //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular não
            //sendo necessario portanto fazer imports novamente neste modulo.


  SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou não.Neste caso
                        //estamos importando. Veja o Modulo OrderModule onde não importamos os
                        //serviços apenas os componentes declarados no SharedModule.
  //OrderModule => este modulo não foi inserido aqui, pois não queremos
  //que o mesmo seja carregado na inicializacao do projeto, porém faremos
  //este carregamento como lazy loading no app.routes.ts


  //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
 ],


  providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],


 //transferido para CoreModule
  //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Observação:

É possível ver o carregamento em backgroud dos módulos lazy se observarmos o Debug do Chrome, se olharmos atentamente na aba Network veremos arquivos com o nome chunk.js, estes arquivos referem-se a cada um dos módulos lazy da aplicação que foram carregados em background, em exercícios anteriores estes arquivos apenas eram carregados e vistos quando eram chamados sob demanda através do click de um link por exemplo.

Seção 9

Vídeo 82

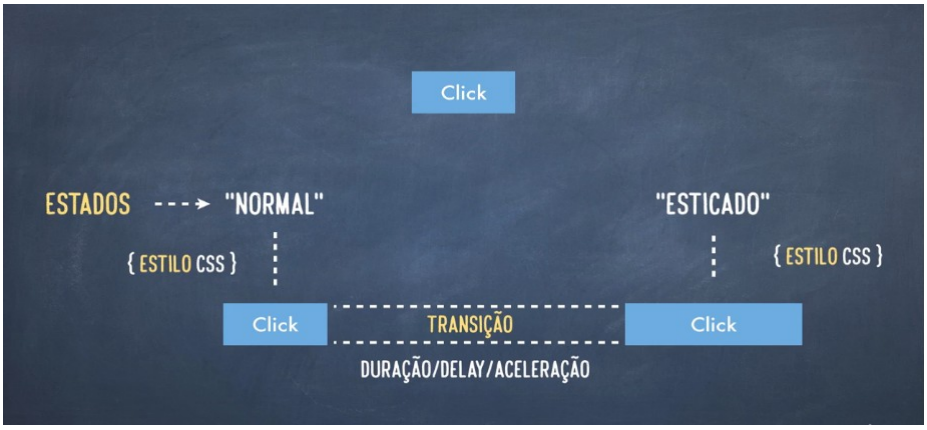
Introdução a Animações

Animações são importantes em uma aplicação por que trazem movimento a ela. Angular tem um módulo específico para animações que é baseada numa espécie chamada Web Animation API.

Atualmente, apenas Chrome e o Firefox implementam parte das features mas a Web Animation API também está disponível como um Polyfil, é um trecho de código neste caso em JavaScript utilizado para "compensar" o fato de navegadores antigos não possuirem funcionalidades mais atuais; para o restante dos browsers o Angular 2 fornece as funções de animação no modulo core, mas a partir do Angular 4 as funções de animações podem ser importadas através

do modulo @angular/animation. Isso tem dois efeitos quem vai usar precisa importá lo mas em compensação quem não usa não leva aquele código que não usaria de qualquer forma deixando a aplicação final menor.

Então vamos ver um exemplo de animação. Imagine que você tem um botão que quando clicado ele aumenta de largura. Se a gente quebrar os passos dessa animação veremos que o botão terá duas configurações que podemos chamar de estados onde teremos o estado normal e outra o estado esticado. Esses estados possuem estilos CSS associados que determinam as características do botão em cada momento; para mudar de um Estado para o outro o botão sofre uma transição de estados e essa transição tem algumas características também tais como a **duração**, o **delay** e a aceleração que chamamos de **Ease.**



Imagine o componente no seu template o qual queremos animar no caso o botão para definir as configurações da animação.

```
@Component({
        template: `<button [@stretch] = "buttonState">Click</button>`  // [@stretch] ==> aqui estamos associando a animacao com o elemento do botao
                                                                        // [@stretch] ==> similar a property binding onde usamos o nome da trigger precedido de uma @
                                                                          //buttonState ==> modifica o estado da trigger
                                                                          //buttonState ==> atributo criado no proprio componente
          animations: [ // animations ==> atributo do decorate @Component
        //trigger('stretch', [ ]) // trigger => define a animacao - onde o primeiro parametro define o nome da animacao = stretch e
                                // o segundo eh um array  usado para definir os estados que a animacao tera.
          trigger ('stretch', [
                state('normal', style({width: '40px'})),   //state funcao que define o estado do componente
                state('stretched', style({width: '120px'})),
                transition('normal => stretched', animate('300ms')), //transicoes entre os estado e tempo de duracao com sentido
                transition('stretched => normal ', animate('500ms')), //'stretched => normal ' ==> sentido de esticado para normal
          ])
        ]
})
export class ButtonComponent {}
```

```
@Component({
  template: `<button [@stretch]="buttonState">Click</button>`
  animations: [
    trigger('stretch', [
      state('normal', style({width: '40px'})),
      state('stretched', style({width: '120px'})),
      transition('normal => stretched', animate('300ms')),
      transition('stretched => normal', animate('500ms'))
    ])
  ]
})
export class ButtonComponent { buttonState = 'normal'}
```

Estados de Animação especiais em Angular:

WildCard: (*) utilizado para representar qualquer estado.

Void: usado para representar uma transição de um elemento que não esta na árvore de componentes ainda ou que vai sair da árvore de componentes.

```
@Component({
    template: `<button [@stretch] = "buttonState">Click</button>`  // [@stretch] ==> aqui estamos associando a animacao com o elemento do bota
                                                                   // [@stretch] ==> similar a property binding onde usamos o nome da trigger precedido de uma @
                                                                   //buttonState ==> modifica o estado da trigger
                                                                   //buttonState ==> atributo criado no proprio componente
     animations: [
    //trigger('stretch', [ ]) // trigger => define a animacao - onde o primeiro parametro define o nome da animacao = stretch e
                            // o segundo eh um array  usado para definir os estados que a animacao tera.
      trigger ('stretch', [
          state('normal', style({width: '40px'})),   //state funcao que define o estado do componente – função de nome "normal"
          state('stretched', style({width: '120px'})),
          transition('* => *', animate('500ms'))
      ])
    ]
})
export class ButtonComponent { buttonState = 'normal'}
```

```
@Component({
  template: `<button [@stretch]="buttonState">Click</button>`
  animations: [
    trigger('stretch', [
      state('normal', style({width: '40px'})),
      state('stretched', style({width: '100px'})),
      transition('* => *', animate('500ms'))
    ])
  ]
})
export class ButtonComponent { buttonState = 'normal'}
```

Se você tiver duas transições onde as animações são idênticas onde a função animated configura as transições com 500ms segundos de duração nós podemos escrevê-las da forma que você está vendo acima com apenas uma transição. A gente informa que a duração será aplicada na mudança de qualquer estado para qualquer outro estado usando wildcard.

Seção 9

Vídeo 83

Instalação do Módulo de Animações

Importando o módulo de animações:

> npm install - - save @angular/animations@4.0.0

//4.0.0 ==> versão para manter a compatibilidade com outros módulos já instalados.

//Antes de instalar a versão do @angular/animations@XXX verique a compatibilidade com a versão do angular que consta no arquivo **package.json**

//na dependencia **"@angular/core": "X.X.X"**

//--save ==> salva nas dependências do projeto

Importando o módulo de Polyfil:

> npm install - - save web-animations-js

//Polyfil: módulo necessário para manter a compatibilidade com browsers antigos que não suportam Web Animations API.

Configurando o arquivo polyfills.ts:

// This file includes polyfills needed by Angular and is loaded before

```
// the app. You can add your own extra polyfills to this file.
import 'core-js/es6/symbol';
import 'core-js/es6/object';
import 'core-js/es6/function';
import 'core-js/es6/parse-int';
import 'core-js/es6/parse-float';
import 'core-js/es6/number';
import 'core-js/es6/math';
import 'core-js/es6/string';
import 'core-js/es6/date';
import 'core-js/es6/array';
import 'core-js/es6/regexp';
import 'core-js/es6/map';
import 'core-js/es6/set';
import 'core-js/es6/reflect';
import 'core-js/es7/reflect';
import 'zone.js/dist/zone';
import 'intl'; //aqui estamos incluindo a depedencia do pacote intl e na linha abaixo o script
import 'intl/locale-data/jsonp/pt-BR.js'; //script necssario para fazer a conversao para moeda brasileira

import 'web-animations-js/web-animations.min.js'//configurando o arquivo polyfills.ts para o modulo de animacoes – video-aula 83
```

Importando módulo para o arquivo app.module.ts:

//inserir no arquivo app.module.ts

```
import {BrowserAnimationsModule} from '@angular/platform-browser/animations' //modulo de animacao do Angular 4
```

//inserir no arquivo app.module.ts

```
imports: [
    BrowserAnimationsModule  //modulo de animacao a disposicao de toda a aplicacao
]
```

Seção 9

Vídeo 84

Definindo um Snackbar

1) Criando o diretório message que conterá o nosso componente.

.../shared/message/ng generate component snackbar –spec=false

2) Configurando o import do componente snackbar.

```
import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'

import {RadioComponent} from './radio/radio.component'

import {RatingComponent} from './rating/rating.component'

import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel

import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup

import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas

                                  //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)

import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos

                                        //permitira criar uma funcao capaz de retornar os

                                        //nosso servicos

import { RestaurantsService } from '../restaurants/restaurants.service';

import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';

import { OrderService } from '../order/order.service';

import { SnackbarComponent } from './messages/snackbar/snackbar.component'; // componente de animacao criado por marques


@NgModule({

  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent, SnackbarComponent],//componentes que serao utilizados neste modulo

  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo

  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

                 RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

                RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

                FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

                ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

                CommonModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo

        SnackbarComponent //componente de animacao criado por marques - fazendo o export desta diretiva nao precisarei importa-la em outro modulo

        ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos



})

export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima

//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa

//de enxugar os outros modulos.

//Perceba que neste modulo nao teremos navegacao.


static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes

                              //bastando para isso usar a funcao no import do modulo root. Veja o arquivo

                              //app.module.ts

  return {

          ngModule : SharedModule,

          providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers

        }

  }


//FUNCAO ORIGINAL

//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes

                              //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
```

```
                                //app.module.ts
//   return {
//            ngModule : SharedModule,
//            providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//          }
//  }

}
```

…/shared/message/snackbar.component.css

```
/* css criado por marques e utilizado no nosso componente de animacao */
.snackbar {
    min-width:  250px;
    margin-left: -125px;
    background-color: #333;
    color: #fff;
    text-align: center;
    border-radius: 2px;
    padding: 16px;
    position: fixed;
    z-index: 1;
    left: 50%;
    bottom: 30px;
}
```

…/shared/message/snackbar.component.ts

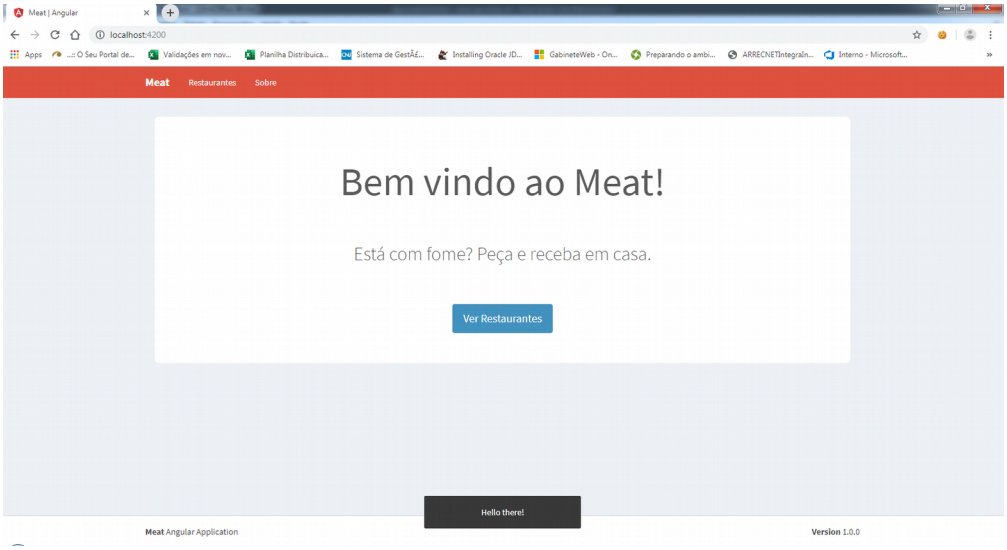
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'mt-snackbar',
  templateUrl: './snackbar.component.html',
  styleUrls: ['./snackbar.component.css']
})
export class SnackbarComponent implements OnInit {
  message: string = 'Hello there!';
  constructor() { }

  ngOnInit() {
  }

}
```

…/shared/message/snackbar.component.html

```
<div class="snackbar">
  {{message}}
</div>
```

/app/app.component.html

```
<div class="wrapper">

  <mt-header></mt-header>

  <!-- Full Width Column -->
  <div class="content-wrapper">
    <div class="container">


      <!-- <mt-home></mt-home> -->
      <router-outlet></router-outlet> <!-- tornando a paginacao dinamica.
                                Veja tambem o arquivo de rotas app.routes.ts -->

    </div>
  </div>

  <!-- snakbar deve ser aqui -->
  <mt-snackbar></mt-snackbar>

  <footer class="main-footer">
    <div class="container">
      <div class="pull-right hidden-xs">
        <b>Version</b> 1.0.0
      </div>
      <strong>Meat</strong> Angular Application
    </div>
  </footer>
</div>
```



Seção 9

Vídeo 85

Aplicando Animações no Snackbar

Nesta seção criaremos uma animação que dispara a visibilidade de um componente snackbar para isso teremos dois estados. Um estado chamado ***hidden*** e outro ***visible*** para controlar exatamente essa mudança da visibilidade da mensagem na tela.

…/shared/messages/snackbar/snackbar.component.ts

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

import { trigger, state, style, transition, animate} from '@angular/animations' //importando as funcoes de animacao

@Component({
  selector: 'mt-snackbar',
  templateUrl: './snackbar.component.html',
  styleUrls: ['./snackbar.component.css'],


    animations: [
      //trigger => funcao que define a animaçao
      //'snack-visibility' ==> nome da animacao
      //[ ] ==> array que contem os estados da animacao

      trigger('snack-visibility', [
          state('hidden', style({
              opacity: 0,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('0px')
              bottom: '0px'
          })),
          state('visible', style({
             opacity: 1,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('30px')
             bottom: '30px'
          })),
          transition('hidden => visible', animate('500ms 0s ease-in')), //animate('duracao delay tipo_de_velocidade')
          transition('visible => hidden', animate('500ms 0s ease-out'))
    ])
  ]
})
export class SnackbarComponent implements OnInit {


  message: string = 'Hello there!';


  snackVisibility: string = 'hidden';


  constructor() { }


  ngOnInit() {
  }
```

```
  toggleSnack(){
    this.snackVisibility = this.snackVisibility === 'hidden' ?  'visible' : 'hidden';
  }

}
```

…/shared/messages/snackbar/snackbar.component.html

```
<button title="Botao Esconder" (click)="toggleSnack()">Click me</button>

<div class="snackbar" [@snack-visibility]='snackVisibility'>
  {{message}}
</div>

<!--
<div class="snackbar" [@snack-visibility]>  aqui estamos associando o template do componente com a trigger de nome snack-visibility
  {{message}}
</div>
-->
```

Seção 9

Vídeo 86

Usando Observable com Snackbar

Nesta seção criaremos um mecanismo de notificação para que o nosso snackbar saiba o momento certo para poder exibir uma mensagem.

Lembra quando a gente implementou um vento lá no item do menu. A gente clicava no botão Adicionar e ele produzia um evento que é ouvido por outro componente que era o componente do carrinho de compras. Então a gente usava uma classe chamada EventEmitter e essa classe permitia deixar as duas partes completamente independentes e ela criava um mecanismo de publish e subscribe. A gente precisa exatamente disso mas naquele momento a gente implementou apenas no escopo dos componentes. Agora a gente precisa de um esquema um pouco maior porque eu não sei quem vai publicar a mensagem. Então como é que eu faria para deixar o EventEmitter disponível para a aplicação inteira, neste caso podemos usar um serviço para isso.

Então a gente pode criar um serviço que vai conter um EventEmitter e as outras partes da minha aplicação podem publicar as mensagens usando o EventEmitter e o que o nosso snackbar precisaria fazer? Inscrever-se neste EventEmitter e quando a mensagem chegar ele estara ouvindo e ativara a animação para poder aparecer a mensagem na tela. Então vamos implementar um serviço de notificações.

…/shared/messages/snackbar/snackbar.component.ts

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

```
import { trigger, state, style, transition, animate} from '@angular/animations' //importando as funcoes de animacao
import { NotificationService } from 'app/shared/messages/notification.service'
import { Observable } from 'rxjs/Observable'
import 'rxjs/add/observable/timer'

@Component({
  selector: 'mt-snackbar',
  templateUrl: './snackbar.component.html',
  styleUrls: ['./snackbar.component.css'],


    animations: [
      //trigger => funcao que define a animaçao
      //'snack-visibility' ==> nome da animacao
      //[ ] ==> array que contem os estados da animacao

      trigger('snack-visibility', [
          state('hidden', style({
            opacity: 0,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('0px')
            bottom: '0px'
          })),
          state('visible', style({
           opacity: 1,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('30px')
           bottom: '30px'
          })),
          transition('hidden => visible', animate('500ms 0s ease-in')),//animate('duracao delay tipo_de_velocidade')
          transition('visible => hidden', animate('500ms 0s ease-out'))
    ])
  ]
})
export class SnackbarComponent implements OnInit {

  message: string = 'Hello there!';
  snackVisibility: string = 'hidden';
  //NotificationService ==> faz a notificacao
 //ShoppingCartService e outros caso haja ==> publicam as mensagens
 //SnackbarComponent ==> consome a mensagem exibindo-a

constructor(private notificationService: NotificationService){}


   // No ngOnInit() será o ponto ideal para fazermos o subscribe(...) uma vez que apos as dependencias do componente
    // SnackbarComponent forem sanadas o metodo ngOnInit() sera carregado inscrevendo um Listener no Observable
    // na propriedade notifier do NotificadorService. Note que na propriedade notifier temos um EventEmitter que é um Observable.
    //Observação: os metodos do servico Http tambem são Observable.
    ngOnInit() {
            this.notificationService.notifier.subscribe(message=>{
             this.message = message; //mensagem recebida
             this.snackVisibility = 'visible'
```

```
        //4000ms => tempo que a mensagem ficara na tela
        Observable.timer(4000).subscribe(timer=> this.snackVisibility = 'hidden')
    })
  }

  toggleSnack(){
    this.snackVisibility = this.snackVisibility === 'hidden' ?  'visible' : 'hidden';
  }

}
```

…/shared/shared.module.ts

```
import {NgModule} from '@angular/core'
import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'
import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                  //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)


import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                  //permitira criar uma funcao capaz de retornar os
                                  //nosso servicos
import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';
import { SnackbarComponent } from './messages/snackbar/snackbar.component';//componente de animacao
import { NotificationService } from './messages/notification.service'

@NgModule({
  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent, SnackbarComponent],//componentes que serao utilizados neste modulo
  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
               RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
            ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  CommonModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
              SnackbarComponent//componente de animacao criado por marques - fazendo o export desta diretiva nao precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos


})
```

```
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
//Perceba que neste modulo nao teremos navegacao.

static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                                //app.module.ts
  return {
          ngModule : SharedModule,


          //declarando servicos na lista de providers
          providers: [RestaurantsService,ShoppingCartService, OrderService, NotificationService]
        }
  }



//FUNCAO ORIGINAL
//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                                //app.module.ts
//   return {
//            ngModule : SharedModule,
//            providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//          }
//  }

}
```

…/shared/message/notification.service.ts

```
import {EventEmitter} from '@angular/core'

export class NotificationService { //classe para notificar as mensagens de erro da aplicacao

  notifier = new EventEmitter<string>()


  notify(message: string){
    this.notifier.emit(message);
  }

}
```

…/restaurant-detail/shopping-cart/shopping-cart.service.ts

```
import {CartItem} from './cart-item.model'
import {MenuItem} from '../menu-item/menu-item.model'
import { Injectable } from '@angular/core'
import { NotificationService } from 'app/shared/messages/notification.service'

@Injectable()
export class ShoppingCartService {

        //items: CartItem[] //==>aqui nos so temos uma declaracao do array
        items: CartItem[] = [] //==> aqui podemos dizer que estamos inicializando com um array vazio



        //NotificationService ==> faz a notificacao
        //ShoppingCartService e outros caso haja ==> publicam as mensagens
        //SnackbarComponent ==> consome a mensagem exibindo-a
        constructor(private notificationService: NotificationService){}

        clear(){
           this.items = [] //recebe um array vazio
        }

        addItem(item: MenuItem){
                let foudItem = this.items.find((mItem)=>mItem.menuItem.id === item.id)
                //(mItem)=>mItem.menuItem.id === item.id //verificando se ja existe um objeto no array atrave do id


                if(foudItem){// foundItem eh um CartItem
                        foudItem.qtd = foudItem.qtd + 1  //incrementando a quantidade em 1 //foudItem.qtd = foudItem.qtd + 1
                }else{
                        this.items.push(new CartItem(item))//item eh um MenuItem
                }

                this.notificationService.notify(`Voce adicionou o item ${item.name}`);//notificador de mensagem

        }

        removeItem(item: CartItem){
            this.items.splice(this.items.indexOf(item),1)//removera 1 objeto CartItem a partir do indice que o objeto passado como parametro ocupa neste //caso "item".
            this.notificationService.notify(`Voce removeu o item ${item.menuItem.name}`);//notificador de mensagem
        }

        total(): number {
                return this.items.map(itemA => itemA.value()).reduce((prev, value) => prev+value,0)
```

```
                //map --> trabalha o parametro refatorando um item pegando apenas o valor neste caso price
                //reduce --> faz um somatorio incializando com 0
        }

        //================================= trabalhando com o componente order ================================//

        increaseQty(item : CartItem){
                item.qtd = item.qtd + 1
        }

        decreaseQty(item : CartItem){
                item.qtd = item.qtd - 1

                if(item.qtd === 0){  // se a quantidade de itens chegar a zero o carItem devera
                        // ser removido da lista em order.component.html
                        this.removeItem(item)
                }
        }
        //Os metodos increaseQty e decreaseQty criados em shopping-cart.service.ts devem ser expostos no OrderSevice
        //e depois seguem para que sejam encapsulados no componente order.componente.html uma vez que nao podemos
        //chamar o servico direto no template de order.componente.html
}
```

Seção 9

Vídeo 87

Usando os Operadores Do e SwitchMap

Resolvendo o problema da concorrência de múltiplas mensagens disputando para serem exibidas a partir do operador timer() do Observable. O problema ocorre, pois estamos trabalhando com dois Observable um que se inscreve para receber a mensagem (this.notificationService.notifier.subscribe(…) - SnackbarComponent) e outro para gerar o time para esconder a mensagem (Observable.timer(3000).subscribe(…) - SnackbarComponent) deixando assim o comportamento do nosso componente completamente confuso. Mas há uma maneira de resolver isso; o Observable possui outros operadores que conseguem encadeá-los deixando-os em uma única configuração diferentemente do que estava ocorrendo onde tínhamos os Observable trabalhando de forma independente. Perceba que na seção anterior tínhamos dois subscribe um dentro do outro o que evidencia dois Observable.

O método ***subscribe(…)*** coloca um listener no Observable e só a partir deste ponto o Observable ira notificar você.

O operador ***do(...)*** permite fazer uma ação mas sem utilizar Observable.

…/shared/messages/snackbar/snackbar.component.ts

```
import { Component, OnInit } from '@angular/core';
import { trigger, state, style, transition, animate} from '@angular/animations' //importando as funcoes de animacao
import { NotificationService } from 'app/shared/messages/notification.service'
import { Observable } from 'rxjs/Observable'
import 'rxjs/add/observable/timer' //timer eh um dos operadores do Observable
```

```
import 'rxjs/add/operator/do'


import 'rxjs/add/operator/switchMap'


@Component({
  selector: 'mt-snackbar',
  templateUrl: './snackbar.component.html',
  styleUrls: ['./snackbar.component.css'],


    animations: [
      //trigger => funcao que define a animaçao
      //'snack-visibility' ==> nome da animacao
      //[ ] ==> array que contem os estados da animacao


      trigger('snack-visibility', [
          state('hidden', style({
            opacity: 0,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('0px')
            bottom: '0px'
          })),
          state('visible', style({
           opacity: 1,   //quando trabalhos com pixel podemos passar apenas o valor (0) ou o seu literal ('30px')
           bottom: '30px'
          })),
          transition('hidden => visible', animate('500ms 0s ease-in')),//animate('duracao delay tipo_de_velocidade')
          transition('visible => hidden', animate('500ms 0s ease-out'))
    ])
  ]
})
export class SnackbarComponent implements OnInit {
  message: string = 'Hello there!';
  snackVisibility: string = 'hidden';
  //NotificationService ==> faz a notificacao
  //ShoppingCartService e outros caso haja ==> publicam as mensagens
  //SnackbarComponent ==> consome a mensagem exibindo-a

  constructor(private notificationService: NotificationService){}



    // No ngOnInit() será o ponto ideal para fazermos o subscribe(...) uma vez que apos as dependencias do componente
    // SnackbarComponent forem sanadas o metodo ngOnInit() sera carregado inscrevendo um Listener no Observable
    //criado a partir da propriedade notifier do NotificadorService. Note que na propriedade notifier temos um EventEmitter que é um Observable
    //Observação: os metodos do servico Http tambem são Observable.
    ngOnInit() {
            //this.notificationService.notifier.subscribe(message=> { ==> subscribe usa Observable
            this.notificationService.notifier.do(message=>{ //==> do nao usa Observable
            this.message = message; //mensagem recebida
```

```
            this.snackVisibility = 'visible' // mudando a visibilidade para "visible"


            //3000ms => tempo que a mensagem ficara na tela
            //import { Observable } from 'rxjs/Observable'
            //import 'rxjs/add/observable/timer'
            //Operadores de Observable: from, map, filter, catch, throw, timer
        //Observable.timer(3000).subscribe(timer=> this.snackVisibility = 'hidden')
       //switchMap ==> troca o Observable inteiro e nao mas a mensagem como no exercicio anterior
      }).switchMap(message=>Observable.timer(3000))
        .subscribe(timer=> this.snackVisibility = 'hidden')
  }

  toggleSnack(){
    this.snackVisibility = this.snackVisibility === 'hidden' ?  'visible' : 'hidden';
  }

}
```

Seção 9

Vídeo 88

Animação com Void e Estilos em Transições (Restaurantes)

…/restaurants/restaurant/restaurant.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant.model';

import { trigger, state, style, transition, animate } from '@angular/animations'; //funcoes de animacao

@Component({
  selector: 'mt-restaurant',
  templateUrl: './restaurant.component.html',
  styleUrls: ['./restaurant.component.css'],
  animations : [
    trigger('restaurantAppeared', [ //trigger => funcao / restaurantAppeared => nome da funcao
      state('ready', style({opacity: 1})), //opacity: 1 deixara o nosso componente visivel
      transition ('void => ready', [ //void => significa que o componente nao esta na árvore de componentes / ready => esta na arvore decomponentes
        style({opacity: 0, transform: 'translate(-30px, -10px)'}),
        animate('500ms 0s ease-in-out')//ease-in-out => entra acelerando e sai desacelerando
      ])//Aqui estamos aplicando estilo na transicao diferente do exercicio anterior onde aplicavamos no estado
    ])//a transicao que estamos insteressados eh de void para ready
  ]  //por isso nao precisamos de dois estados (state) diferentemente do exercicio anterior
})
export class RestaurantComponent implements OnInit {
```

```
  restaurantState = 'ready'; //veja que estamos associando uma propriedade do componente a um estado da trigger


  @Input() restaurant : Restaurant


  constructor() { }
  ngOnInit() {
  }
}
```

…/restaurants/restaurant/restaurant.component.html

```
<a [routerLink]="['/restaurants', restaurant.id]"> <!-- de acordo com a rota no arquivo app.routes.ts -->
  <div class="place-info-box" [@restaurantAppeared]="restaurantState"><!-- associando a trigger a propriedadeo do componente que representa
                                                o estado da trigger atraves do template-->
    <span class="place-info-box-icon"><img [src]="restaurant?.imagePath" /></span>


    <div class="place-info-box-content">
      <span class="place-info-box-text">{{restaurant?.name}}</span>
      <span class="place-info-box-star"><i class="fa fa-star"></i>{{restaurant?.rating}}</span>
      <span class="place-info-box-detail">{{restaurant?.category}}</span>
      <span class="place-info-box-detail">{{restaurant?.deliveryEstimate}}</span>
    </div>
    <!-- /.info-box-content -->
  </div>
</a>
```

Seção 9

Vídeo 89

Animação nos Itens do Menu

…/restaurant-detail/menu-item/menu-item.component.ts

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

import { MenuItem } from './menu-item.model';

import { trigger, state, style, transition, animate } from '@angular/animations'; //funcoes de animacao

@Component({
  selector: 'mt-menu-item',
  templateUrl: './menu-item.component.html',
  animations : [
    trigger('menuItemAppeared', [ //trigger => funcao / restaurantAppeared => nome da funcao
      state('ready', style({opacity: 1})), //opacity: 1 deixara o nosso componente visivel
      transition ('void => ready', [ //void => significa que o componente nao esta na arvore de componentes / ready => esta na arvore decomponentes
        style({opacity: 0, transform: 'translateY(-20px)'}),//apenas no eixo Y
        animate('800ms 0s ease-in')//ease-in => entra acelerando
                    ])//Aqui estamos aplicando estilo na transicao
                  ])//a transicao que estamos insteressados eh de void para ready
  ]  //por isso nao precisamos de dois estados (state)
})
export class MenuItemComponent implements OnInit {
```

```
        menuItemState = 'ready'; //veja que estamos associando uma propriedade do componente
            //a um estado da trigger

        @Input() menuItem: MenuItem
        //Sempre que tivermos uma propriedade onde o componente parent, MenuComponent,
        //informará um valor para o componente filho, MenuItemComponent, precisaremos
        //marcar esta propriedade com o "decorate input" neste caso a propriedade "menuItem".
        //Agora nosso componente menu-item.component está pronto, a seguir
        //trabalharemos no nosso template neste caso o "menu-item.componente.html".

        @Output() add = new EventEmitter() //este objeto sera acionado como um evento no componente parent MenuComponent

        constructor() { }

        ngOnInit() {

        }

        emitAddEvent(){
                this.add.emit(this.menuItem)//pegando o objeto clicado no componente em menu-item-component.html
        }                    //o qual sera relancado, perceba o metodo emit(...) para o
                            //menu.component.html e sera pego atraves de $event

}
```

…/restaurant-detail/menu-item/menu-item.component.html

```
<div class="menu-item-info-box" [@menuItemAppeared]="menuItemState"><!-- associando a trigger a propriedadeo do componente que representa
                                                o estado da trigger atraves do template-->

        <span class="menu-item-info-box-icon"><img [src]="menuItem.imagePath"></span>

        <div class="menu-item-info-box-content">
          <span class="menu-item-info-box-text">{{menuItem.name}}</span>
          <span class="menu-item-info-box-detail">{{menuItem.description}}</span>
          <span class="menu-item-info-box-price">{{menuItem.price | currency: 'BRL': true}}</span>
          <a (click)="emitAddEvent()" class=""><i class="fa fa-plus-circle"></i> Adicionar</a>
        </div>
        <!-- /.info-box-content -->
</div>
<!-- /.info-box -->
```

Seção 9

Vídeo 90

Animação com Keyframes

Exemplo 1:

```
 animations : [
    trigger('trg', [ //trigger => funcao / trg => nome da funcao
    ...
   transition ('st1 => st2', [
   animate('500ms 0s ease-in-out', keyframes ([//ease-in-out => entra acelerando e sai desacelerando
            style({transform: 'transform(0px,0px)', offset:0})
            style({transform: 'transform(100px,0px)', offset:0.6})//offset => indica onde dentro da animacao  o keyframe estara
            style({transform: 'transform(100px,0px)', offset:1})
     ]))
   ])
 ] })
```

Exemplo 2:

```
 animations : [
    trigger('trg', [ //trigger => funcao / trg => nome da funcao
    ...
   transition ('st1 => st2', [
   animate('500ms 0s ease-in-out', keyframes ([//ease-in-out => entra acelerando e sai desacelerando
            style({opacity: 0, width:'100px', offset:0})
```

```
            style({opacity: 0.8, width:'120px', offset:0.8}) //offset => indica onde dentro da animacao  o keyframe estara
            style({opacity: 1, width:'100px', offset:1})
        ]))
    ])
 ] })
```

Seção 9

Vídeo 91

Usando Animação com Keyframes nos Itens do Carrinho

…/restaurant-detail/shoping-cart/shoping-cart.component.ts

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

import {ShoppingCartService} from './shopping-cart.service'

import {CartItem} from './cart-item.model'

import { MenuItem } from '../menu-item/menu-item.model'

import { trigger, state, style, transition, animate } from '@angular/animations'; //funcoes de animacao

//import { trigger, state, style, transition, animate, keyframes } from '@angular/animations'; //funcoes de animacao

import { keyframes } from '@angular/animations'; //funcao de animacao

@Component({
  selector: 'mt-shopping-cart',
  templateUrl: './shopping-cart.component.html',
  animations : [
    trigger('row', [ //trigger => funcao / row => nome da funcao
      state('ready', style({opacity: 1})), //opacity: 1 deixara o nosso componente visivel
      transition ('void => ready', animate('800ms 0s ease-in', keyframes([//trasition => o item entra no carrinho
        style({opacity: 0,   transform: 'translateX(-30px)', offset:0}),
        style({opacity: 0.8, transform: 'translateX(10px)', offset:0.8}),
        style({opacity: 1,   transform: 'translateX(0px)',   offset:1})
      ])), //void => significa que o componente nao esta na arvore de componentes / ready => esta na árvore decomponentes, ou seja, no DOM


      transition ('ready => void', animate('800ms 0s ease-in', keyframes([//trasition => o item sai do carrinho
        style({opacity: 1,   transform: 'translateX(0px)', offset:0}),
        style({opacity: 0.8, transform: 'translateX(-10px)', offset:0.2}),
        style({opacity: 0,   transform: 'translateX(30px)',   offset:1})
      ])) //void => significa que o componente nao esta na arvore de componentes / ready => esta na árvore decomponentesn, ou seja, no DOM
    ])//offset => indica onde dentro da animacao  o keyframe estara
                ]
})
export class ShoppingCartComponent implements OnInit {


  rowState = 'ready';


  constructor(private shoppingCartService: ShoppingCartService) { }


  ngOnInit() {
  }


// MEU METODO ALTERNATIVO DO EXERCICIO
  items(): CartItem[] { //retorna a lista
                return this.shoppingCartService.items
  }


//  METODO ORIGINAL DO EXERCICIO
//  items(): any[] { //retorna a lista
```

```
//                  return this.shoppingCartService.items
//  }


  total(): number {//este metodo sera chamado direto na pagina: shopping-cart.component.html
                  return this.shoppingCartService.total()
  }


  clear(){
    return this.shoppingCartService.clear()
  }


  removeItem(item: any){
    this.shoppingCartService.removeItem(item)
  }


  addItem(item: MenuItem){
    this.shoppingCartService.addItem(item)
  }


}
```

…/restaurant-detail/shoping-cart/shoping-cart.component.html

```
<!--INICIO DO CARRINHO-->

<div class="box box-solid">



<div class="box-header">

  <i class="fa fa-shopping-cart"></i>



  <h3 class="box-title">Carrinho</h3>

</div>



<!-- /.box-header -->

<div class="box-body">



  <div *ngIf="!items().length">

    <p class="text-center">

      Seu carrinho está vazio!!!

    </p>

  </div>

  <div class="table-responsive" *ngIf="items().length">

      <table class="table">

        <tbody>

        <tr *ngFor="let item of items()" [@row]="rowState"><!-- associando a trigger a propriedadeo do componente que representa

                                                                o estado da trigger atraves do template -->



          <th>({{item.qtd}}x) {{item.menuItem.name}}</th>

          <td class="text-right">{{item.value() | currency: 'BRL': true}}</td>

          <td class="text-right">

            <a (click)="removeItem(item)" class="btn btn-sm danger"><i class="fa fa-remove"></i></a>

          </td>

        </tr>

        <tr>
```

```
        <th>Total:</th>
        <td class="text-right">{{total() | currency: 'BRL': true}}</td>
      </tr>
    </tbody></table>
  </div>
</div>
<div class="box-footer" *ngIf="items().length">
  <div class="pull-right">
    <button (click)="clear()" type="button" class="btn btn-danger"><i class="fa fa-trash"></i> Limpar</button>
    <a [routerLink]="['/order']" class="btn btn-success"><i class="fa fa-credit-card"></i> Fechar Pedido</a>
  </div>
</div>
</div>
<!-- FIM DO CARRINHO -->
```

Seção 9

Vídeo 92

Resumo de Animações

Instalação do Modulo de Animação:

@angular/animations => modulo disponivel a partir do Angular 4

web-animations-js => polyfill para browsers antigos que nao suportam animaçao arquivo que fornecerá suporte a funcionalidades novas aos navegadores antigos.

BrowserAnimationsModule //No arquivo app.module.ts => import { BrowserAnimationsModule } from '@angular/platform-browser/animations' //modulo de animacao do Angular 4

Definição da função de animação:

trigger => define a função de animação

state => define o estado da animação onde podemos definir o estilo css

transition => define a transição de estado onde também pode ter seu próprio estilo css

void => estado especial utilizado quando o componente não esta na árvore, ou seja, no DOM

wildcard '*'=> representa uma transição para qualquer estado

animate => define a duração, delay e a aceleração da animação(easing) //duração e delay podem ser informados em segundos(s) ou milisegundos(ms)

keyframes => animação onde utilizamos a propriedade **offset** como marcos de animação

Para outras propriedades de animação veja o endereço:

w3.org/TR/css3-transitions/#animatable-properties

Rotas com Wildcard – Página Não Encontrada (Page Not Found)

No caso de digitarmos uma url na barra de endereços do navegador que não coincida com nenhuma das rotas configuradas na nossa aplicação será exibida uma "Página Não Encontrada" (Page Not Found) ao usuário. A Rota com Wildcard (**{path: '**', component: NotFoundComponent}**) deve ser inserida no final das rotas porque a primeira correlação entre o que foi digitado no navegador com uma das regras inseridas no Routes será a página retornada ao usuário. Em geral as rotas mais específicas estão no início do Routes enquanto as mais genéricas ficam no final.

1)

> ng generate component not-found - -spec=false

2)

/not-found/not-found.component.ts

```
<!-- INÍCIO DO CONTEÚDO -->
    <section class="content-header">
    </section>

    <!-- Main content -->
    <section class="content">

      <div class="jumbotron">
        <img src="assets/img/404.png" /><h2>Página Não Encontrada</h2>
        <p>
          Está com fome? Acesse a página de <a href="#">restaurantes</a>.
        </p>
      </div>
    </section>
  <!-- FIM DO CONTEÚDO -->
```

3)

/app/app.modules.ts

```
import { Component } from '@angular/core';
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
```

```
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
//import {OrderComponent} from './order/order.component'
import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'
import { NotFoundComponent } from './not-found/not-found.component';

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        //{path: 'about', component: AboutComponent},//esta rota não será mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule - rota responsavel pelo Lazy Loading
        {path: 'restaurants', component: RestaurantsComponent},
        //{path: 'order', component: OrderComponent},//esta rota não será mais usada uma vez que criamos o OrderModule
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                  {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a página menu
                        {path: 'menu', component: MenuComponent},        //como padrão quando for especificado
                        {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.
                ]},                                  //Veja: http://localhost:4200/restaurants/green-food
                                        //eh igual a http://localhost:4200/restaurants/green-food/menu
                                 //devido ao redirectTo.
         {path: 'order-summary', component: OrderSummaryComponent},
        {path: 'order', loadChildren: './order/order.module#OrderModule'},//rota para o Modulo OrderModule - rota responsavel pelo Lazy Loading
        {path: '**', component: NotFoundComponent} //rota para a página não encontrada - regra com wildcard
]
```

Seção 10

Vídeo 94

Construindo a Aplicação – Gerando o Build da Aplicação

Preparando a Aplicação para Build de Desenvolvimento

O build compilará e empacotará nossa aplicação.

> ng build //o build gerará um pacote da aplicação com os arquivos que o webpack cria e com as mesmas //configurações do comando ***ng serve***,ou seja, com as mesmas configurações de desenvolvimento. // Perceba que será criada uma pasta ***dist*** com todos os arquivos desta compilação que só existem //no build de desenvolvimento. Por exemplo os arquivos com extensão ".map".

Preparando a Aplicação para Build de Produção

> ng build - -prod //ng build - -target=production

Build de Desenvolvimento e Build de Produção:

Ambas as formas criarão os bundles da nossa aplicação e tentarão realizar o “tree shaking” que é o ato de eleminar código que fora importado desnecessariamente tornando os bundles mais enxutos.

Build de Desenvolvimento vs Build de Produção:

O Build de Produção removerá os espaços em branco tornando os bundles menores e permitindo ao browser um acesso mais rápido. Além da compilação das classes dos componentes para javascript há também no Build de Produção a compilação dos templates em html para typescript e em seguida para javascript antes mesmo da aplicação ser servida ao browser o que permite detectar erros de tipagem entre outros erros. A detecção de erros só é possível por que no Build de Produção ocorre a transcrição de html para typescript e em seguida para javascript e é ***nesta segunda fase que é possível detectar os erros antes de chegar ao browser (Ahead of Time)*** o que não ocorre em Build de Desenvolvimento embora ocorra a compilação dos arquivos typescript em javascript, o mesmo não ocorre para os arquivos em html, ou seja, não ocorre a compilação de html para typescrpt, mas sim para javascript no browser não permitindo detectar erros de compilação. ***A compilação de html direto para javascript no browser é chamada neste caso de Just in Time.***

Uma outra característica do Build de Produção é não exibir mais arquivos de extensão do tipo ".map". A inexistência deste tipo de arquivo acarreta não permitir o debug da aplicação em produção o que na realidade não é um problema, pois uma aplicação em produção não tem esta função e deve ser verdadeiramente enxuta.

Onde esta localizada a configuração de build de produção

/angular-cli.json

```
{
 "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
 "project": {
  "version": "1.0.0",
  "name": "meat"
 },
 "apps": [
  {
   "root": "src",
   "outDir": "dist",
   "assets": [
    "assets",
    "favicon.ico"
   ],
   "index": "index.html",
   "main": "main.ts",
   "polyfills": "polyfills.ts",
   "test": "test.ts",
   "tsconfig": "tsconfig.json",
   "testTsconfig": "tsconfig.spec.json",
   "prefix": "mt",
```

```
    "mobile": false,
    "styles": [
      "../node_modules/font-awesome/css/font-awesome.min.css",
      "../node_modules/admin-lte/bootstrap/css/bootstrap.min.css",
      "../node_modules/admin-lte/dist/css/AdminLTE.min.css",
      "../node_modules/admin-lte/dist/css/skins/skin-red.min.css",
      "../node_modules/admin-lte/plugins/iCheck/flat/red.css",
      "styles.css"
    ],
    "scripts": [
      "../node_modules/jquery/dist/jquery.min.js",
      "../node_modules/admin-lte/bootstrap/js/bootstrap.min.js",
      "../node_modules/admin-lte/dist/js/app.min.js"
    ],
```

"environmentSource": "environments/environment.ts",

"environments": {

"dev": "environments/environment.ts", //aponta para fazer o build de desenvolvimento de acordo com o comando **"ng build"**

"prod": "environments/environment.prod.ts" //aponta para fazer o build de produção de acordo com o comando **"ng build - -drop"**

}

```
  }
],
"addons": [],
"packages": [],
"e2e": {
  "protractor": {
    "config": "./protractor.conf.js"
  }
},
  "lint": [
  {
    "project": "src/tsconfig.app.json"
  },
  {
    "project": "src/tsconfig.spec.json"
  },
  {
    "project": "e2e/tsconfig.e2e.json"
  }
],
"test": {
  "karma": {
    "config": "./karma.conf.js"
  }
},
"defaults": {
  "styleExt": "css",
```

```
    "prefixInterfaces": false,
    "inline": {
      "style": false,
      "template": false
    },
    "spec": {
      "class": false,
      "component": true,
      "directive": true,
      "module": false,
      "pipe": true,
      "service": true
    }
  }
}
```

Seção 10

Vídeo 95

Modificando a Estratégia de Navegação (Hash)

Estratégia de navegação e caminhos de url no Angular:

Angular I:

localhost:4200/#/outros-caminhos

Angular II:

localhost:4200/outros-caminhos

No Angular I temos um padrão de url onde a mesma possui um Hash e apos este viria todos os caminhos, rotas, da nossa aplicacao, porém a partir do Angular II que passou-se a utilizar o padrão da API da Especificação Html5 e este padrão de navegação não utiliza mais o Hash. Quando estamos em um ambiente de desenvolvimento isto não seria um problema pois perceb-se que a própria url é montada de acordo com as nossas rotas. No caso do Angular II é utilizado a API da Especificação Html5 para fazer push do histórico das nossas rotas navegadas de modo que se usarmos os botoes do browser, back ou forward, o Angular intercepta os clicks de navegação para saber as rotas que foram usadas.Perceba que todo este processo é feito na noss aplicação cliente, ou seja, no browser e quem faz o controle das rotas navegads é o próprio Framework Angular. Quando acessamos o endereço localhost:4200 obtemos como resposta é a página index.html que contém os caminhos de todos os scripts e estes disparam o bootsrap da aplicação carregando as classes necessárias e os módulos necessários. Após carregados as classes e modulos o Angular verifica se o endereço digitado no browser coincide com uma das rotas no nosso arquivo de rotas da aplicação caso encontre o conteúdo do componente correspondente será exibido, porém é importante salientar que toda esta navegação ocorre no cliente, navegador.

Agora, o que aconteceria se levassemos o nosso build de produção, conteúdo da pasta dist, para um servidor http básico?

Veremos que este padrão de navegação apresenta problemas quando a aplicação gerada pelo build de produção é deploiada em um servidor http básico o que não é o ideal. Se observarmos atentamente todas as operações que ocorrem na aplicação deploiada no servidor webpack para desenvolvimento também ocorrerá neste servidor http básico, porém existe uma diferença. Quando tentarmos fazer bookmark de uma de nossas páginas e voltarmos acessar a página novamente neste servidor http básico será exibida uma página de erro 404 enquanto no servidor webpack de desenvolvimento a página que fora feita o bookmark será acessada normalmente. O erro acontece por que toda a navegação de rotas esta disponível apenas na nossa aplicação e a mesma roda no browser e não no servidor, quando foi feito o bookmark foi solicitado ao servidor http básico uma rota cuja estrutura de pastas não existe na nossa

aplicação, logo o servidor http básico apresenta uma página de erro 404 que é o erro padrão de um servidor simples sem qualquer configuração customizada. Uma pergunta que dever ser feita é por que quando fazemos um bookmark de uma página e o acesso é feito em um servidor webpack de desenvolvimento conseguimos obter a página e no servidor http básico não temos este mesmo resultado?

A diferença reside no fato do servidor webpack sempre devolver uma página index.html toda vez que for solicitado uma página que não existe, quando dizemos que não existe estamos nos referindo aquele caminho de pastas pois o que existe de fato são apenas scripts depois que é feito o build da aplicação, e como a página index.html possui os scripts de todas as rotas da aplicação será feita novamente um bootsrap da nossa aplicação que permitirá ao Angular acessar a url e tentará relacionar com uma das rotas no arquivo de rotas.

Perceba que quando a página index.html for retornada aparecerá a palavra "Loading..." que só existe nesta página confirmando portanto que a página index.html foi retornada pelo servidor webpack. Para resolver o problema precisamos apenas configurar a nossa própria aplicação o que é muito bem-vindo uma vez que há situações onde não temos acesso ao servidor http, o que faremos é configurar a nossa aplicação para voltar a obedecer o padrão de navegação do Angular I onde se usa o Hash. Quando o Hash é inserido na url da nossa aplicação, localhost:4200/#/, tudo que virá após o Hash não será processado pelo servidor, mas apenas no browser que por sua vez pedirá a página index.html que possui todos os scripts e fará o bootstrap da aplicação o que permitirá por sua vez ao Angular verificar a url e comparar com o nosso arquivo de rotas; caso a url coincida com uma das rotas o conteúdo do componente correspondente será retornado. Veja abaixo o procedimento para solucionar o problema sem mesmo realizarmos qualquer configuração no servidor:

1) Faça o import abaixo:

import {LocationStrategy, HashLocationStrategy} from '@angular/common'

2) Insira a linha abaixo no "providers" do arquivo de rotas (/app.module.ts) :

providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}**]**,

//{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao

//se tentar fazer bookmark com nossas páginas

3) Gere novamente o build de produção:

> ng build --prod //ng build --target=production

//será gerada uma pasta "dist" com os bundles da nossa aplicação a qual deve ser deploiada no servidor http básico ou outro qualquer.

4) Inicialize novamente a aplicação através do servidor.

/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
import { HttpModule } from '@angular/http';
import { RouterModule } from '@angular/router';
import {ROUTES} from './app.routes'
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
```

```
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                                //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'   //==> importando um modulo core, mas foi comentado
                                                                //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router' //Import necessario para fazer o carregamento
                                                                //em background dos modulos Lazy
```

import {LocationStrategy, HashLocationStrategy} from '@angular/common'

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
```

```
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    HttpModule, //==> exemplo de Core Module. Veja o github do Angular
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
    //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                              //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular não
                  //sendo necessario portanto fazer imports novamente neste modulo.


    SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou não.Neste caso
                              //estamos importando. Veja o Modulo OrderModule onde não importamos os
                              //serviços apenas os componentes declarados no SharedModule.
    //OrderModule => este modulo não foi inserido aqui, pois não queremos
    //que o mesmo seja carregado na inicializacao do projeto, porém faremos
    //este carregamento como lazy loading no app.routes.ts

    //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
  ],


    providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}],
    //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
    //se tentar fazer bookmark com nossas páginas


  //transferido para CoreModule
  //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Seção 10

Vídeo 96

Personalizando a Construção com Variáveis de Ambientes

A aplicação pode ter seus arquivos de ambiente personalizados a partir do arquivo angular-cli.json, veja a seção environment:

/angular-cli.json

```
"environmentSource": "environments/environment.ts",
    "environments": {  //propriedade
     "dev": "environments/environment.ts",   //a partir dos parâmetros passados no build determinaresmos que arquivos serão chamados, por exemplo: ng build - - prod   /  ng build - - dev
    "prod": "environments/environment.prod.ts"   //podemos criar tambem um outra variavel apontando para um outro arquivo:  "test": "environments/environment.test.ts"
    }
```

/app.api.ts

```
export const MEAT_API = 'http://localhost:3000'  // variável que define a nossa url de back-end da aplicação
```

Alterando e adicionando variável de ambiente para dois ambientes diferentes neste caso produção e desenvolvimento.

/environments/environment.ts

```
//export const environment = {        //conteudo do arquivo original
//  production: false
//};
export const environment = {        //conteudo do arquivo modificado
  production: false,
  api: 'http://localhost:3000'          //api => nossa proprieadae criada para atender ao acesso em ambiente de desenvolvimento
};
```

/environments/environment.prod.ts

```
//export const environment = {        //conteudo do arquivo original
//  production: true
//};
export const environment = {  //conteudo do arquivo modificado
  production: true,
  api: 'https://localhost'            //api => propriedade criada para atender ao acesso em ambiente de produção em um servidor http,
                                      //na realidade não teremos o servico rodando em 'https://localhost' mas queremos apenas
                                      //verificar se ao fazermos o build de producao a url configurada sera chamada, após
```

```
                    //a tentativa de acesso verifique a ferramenta Web Tools > Network para verificar o
                    //endereço chamado.
};
```

/app.api.ts

```
import {environment} from '../environments/environment'  //importando o objeto environment localizado no arquivo "environment.ts"

//export const MEAT_API = 'http://localhost:3000'  // variável que define a nossa url de back-end da aplicação

export const MEAT_API = environment.api     //o que definira qual ambiente sera chamado sera o build, por exempo:
                                            // >ng build - -prod / ng build - -dev / ng build - -outra_variavel_criada
```

Uma forma de testar caso não tenhamos um servidor http seria utilizar o comando abaixo, para ver se a url https:localhost será realmente chamada em produção.

```
>ng serve - -prod      //o comando ao lado apenas inicializa a aplicação com as configurações de produção, mas não
                       //faz o build de produção do projeto.
                       //teremos que obter um erro ao tentar acessar esta url, pois não há nenhum servico neste
                       //endereço.
```

Seção 11

Vídeo 97

Deploy no Apache Http Server – com Estratégia de Hash

Realizando o deploy da aplicação em um Servidor Apache Http.

1) >ng builder - -prod

2) Transfira o conteúdo da pasta "dist" para o servidor

3) Inicialize o servidor

4) Acesse a aplicação pelo navegador: localhost:numero_da_porta

//Obs.: Utilizando a Estratégia de Hash não precisaremos configurar o servidor Apache Http.

Seção 11

Vídeo 98

Deploy Usando a Estratégia de Caminho - Apache Http Server

Neste deploy usaremos a estratégia de caminho onde será necessário configurar o nosso Servidor Apache Http para retornar a página index.html da nossa aplicação.

1) Remova a configuração de **Hash** localizada no arquivo app.module.ts para voltarmos a estratégia de Caminho.

providers: [**{provide: LocationStrategy, useClass: HashLocationStrategy}**,{provide: LOCALE_ID, useValue: 'pt-BR'}],

2) >ng build - -prod

3) Altera o arquivo environment.prod.ts para a configuração abaixo para podermos acessar a API Json de back-end.

/envirnoment.prod.ts

```
export const environment = {
  production: true,
 api: 'http://localhost:3000'
 // api: 'https://localhost'
};
```

4) Transfira o conteúdo da pasta "dist" para o servidor

5) Insira o trecho abaixo no arquivo "httpd.conf " do Apache na sessão ***DocumentRoot*** deste arquivo.

//fonte:https://angular.io/guide/deployment

```
RewriteEngine On
# If an existing asset or directory is requested go to it as it is
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f [OR]
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -d
RewriteRule ^ - [L]

# If the requested resource doesn't exist, use index.html
RewriteRule ^ /index.html
```

6) No arquivo httpd.conf

….

LoadModule rewrite_module modules/mod_rewrite.so //habilite esta linha

….

DocumentRoot "/caminho/ate/o/seu/diretorio/onde/esta/o/projeto"

<Directory "/caminho/ate/o/seu/diretorio/onde/esta/o/projeto">

….

…….

```
RewriteEngine On
# If an existing asset or directory is requested go to it as it is
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f [OR]
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -d
RewriteRule ^ - [L]

# If the requested resource doesn't exist, use index.html
RewriteRule ^ /index.html
```

……..

</Directory>

7) Reinicie o Apache

8) Acesse a aplicação pelo navegador: localhost:numero_da_porta

Seção 11

Vídeo 99

Deploy Usando o arquivo htaccess - Apache Http Server

Neste exemplo faremos o deploy da aplicação usando Estratégia de Caminho com Angular usando o arquivo ".htaccess" do Apache para fazer sobreposição de regras. Esta estratégia é muito usada quando não temos acesso ao arquivo principal, httpd.conf, do Apache.

1) Altere o valor da propriedade "AllowOverride" de "None" para "All" no arquivo httpd.conf

```
//AllowOverride None
AllowOverride All   //permitirá que todos os diretórios do Apache tenham o arquivo ".htaccess"
```

2) Crie o arquivo ".htaccess" na pasta onde será deploiada a nossa aplicação no servidor Apache.

3) Insira o trecho abaixo no arquivo ".htaccess" e remova o mesmo trecho caso exista no arquivo "httpd.conf" do Apache.

```
RewriteEngine On
# If an existing asset or directory is requested go to it as it is
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f [OR]
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -d
RewriteRule ^ - [L]

# If the requested resource doesn't exist, use index.html
RewriteRule ^ /index.html
```

4) Inicie o servidor para testar a nossa aplicação.

Seção 11

Vídeo 100

Modificando o base-href para Deploys em Subdiretórios

Veremos agora como publicar nossa aplicação em um subdiretório do nosso servidor ao contrário dos exercícios anteriores onde estávamos publicando nossa aplicação na pasta raiz do nosso servidor.

1) Crie uma pasta chamada "meat" no diretório raiz do nosso servidor.

2) Transfira a aplicação gerada com "**ng build - -prod - -bh=/meat/**" para a pasta "meat".

Obs.:

index.html

…

….

<base href= "/" > //==> componente localizado na página index.html que informa ao browser que todos os //recursos da nossa aplicação estarão disponíveis a partir do endereço informado em "href" //neste caso a partir da raiz do nosso domínio. Este componente será usado pelo nosso //"<router-outlet>" para montar as urls. Acontece que neste momento nossa aplicação não //esta mais disponível a partir da raiz e sim da pasta "meat" onde nossa aplicação foi //alocada e inserida no servidor. Para resolvermos este problema de forma elegante //publicaremos nossa aplicação conforme o código a seguir: "**ng build - -prod - -bh=/meat/**".

3) Acessa a aplicação a partir de: localhost:/meat/

Seção 11

Vídeo 101

Resumo do Processo de Build e Deploy

Build:

> ng build - -prod

Modificando a url base do projeto:

> ng build - -prod - -bh=/url/

Configurações da aplicação para produção devem ficar em:

src/environments/environment.prod.ts

Url location strategy:

path vs hash

Deploy no Apache:

httpd.conf

.htaccess

Seção 12

Vídeo 102

ESTA APLICAÇÃO ESTA APRESENTANDO PROBLEMAS EM /restaurants.component.ts e /restaurants.component.html SIMPLESMENTE OS RESTAURANTES NÃO SÃO EXIBIDOS QUANDO INSERIMOS [@toggleSearch]="searchBarState" E O EVENTO (click)="toggleSearch()" NÃO FUNCIONA. VEJA O PROJETO: meat-app-starter-41.

NOS PROJETOS SEGUINTES O QUE FORA IMPLEMENTADO EM "meat-app-starter-41" FOI REMOVIDO.

```
<section class="content-header">
    <h1>
      Todos os Restaurantes
      <a (click)="toggleSearch()" class="search-link pull-right"><i class="fa fa-search"></i></a>
    </h1>


    <!-- barra de busca -->
    <div class="row">
      <div class="col-xs-12" [@toggleSearch]="searchBarState">
        <form>
          <div class="form-group">
            <input type="text" class="form-control" placeholder="O que você está procurando?"/>
          </div>
        </form>
      </div>
    </div>
  </section>
```

Adicionando a Busca de Restaurantes

/restaurant.component.ts

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

```
import { Restaurant } from './restaurant.model';
import {trigger, state, style, transition, animate} from '@angular/animations';

@Component({
  selector: 'mt-restaurant',
  templateUrl: './restaurant.component.html',
  styleUrls: ['./restaurant.component.css'],
  animations: [
    trigger('restaurantAppeared', [
      state('ready', style({opacity: 1})),
      transition ('void => ready', [
          style({opacity: 0, transform: 'translate(-30px, -10px)'}),
          animate('300ms 0s ease-in-out')
        ])
      ])
    ]
 })
export class RestaurantComponent implements OnInit {

  @Input() restaurant : Restaurant

  constructor() { }

  ngOnInit() {
  }

}
```

/restaurants.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';
import {trigger, state, style, transition, animate} from '@angular/animations';

@Component({
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html',
  animations: [
    trigger('toggleSearch', [
      state('hidden',  style({opacity: 0, "max-height": "0px"})),
      state('visible', style({opacity: 1, "max-height": "70px", "margin-top": "20px"})),
    transition('* => *', animate('250ms 0s ease-in-out'))
        ])
      ]
})
export class RestaurantsComponent implements OnInit {

searchBarState = 'hidden'          //controla a visibilidade
```

```
// movido para a classe de servico restaurants.service.ts
// restaurants : Restaurant[] = [
//                    {
//                              id: "bread-bakery",
//                              name: "Bread & Bakery",
//                              category: "Bakery",
//                              deliveryEstimate: "25m",
//                              rating: 4.9,
//                              imagePath: "assets/img/restaurants/breadbakery.png"
//                    },
//                    {
//                              id: "burger-house",
//                              name: "Burger House",
//                              category: "Hamburgers",
//                              deliveryEstimate: "100m",
//                              rating: 3.5,
//                              imagePath: "assets/img/restaurants/burgerhouse.png"
//                    }
// ]
//
//  restaurants : Restaurant[]
//
//  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas
//                    this.restaurants = this.restaurantsService.listarRestaurants();
//  }
  restaurants : Restaurant[] //este array sera carregado a partir do metodo subscribe(...)
  constructor(private restaurantsService : RestaurantsService ) {
  }

  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas
                    this.restaurantsService.listarRestaurants()
                              .subscribe(restaurants => this.restaurants = restaurants);
  }

  toggleSearch(){
    this.searchBarState = this.searchBarState == 'hidden' ? 'visible' : 'hidden'
  }

}
```

/restaurants.component.html

```
<!-- INíCIO do CONTEÚDO -->
<section class="content-header">
      <h1>
        Todos os Restaurantes
        <a (click)="toggleSearch()" class="search-link pull-right"><i class="fa fa-search"></i></a>
      </h1>
      <!-- barra de busca -->
      <div class="row">
        <div class="col-xs-12" [@toggleSearch]="searchBarState">
```

```
          <form>
            <div class="form-group">
              <input type="text" class="form-control" placeholder="O que você está procurando?"/>
            </div>
          </form>
        </div>
      </div>
    </section>
    <section class="content">
      <div class="row">
        <div *ngFor="let rest of restaurants" class="col-sm-6 col-xs-12">
          <mt-restaurant  [restaurant]="rest" ></mt-restaurant>
          <!-- perceba que precisaremos de @Input na propriedade restaurant pois o componente parent neste caso restaurants.component.html ira passar os valores atraves do ngFor -->
        </div>
      </div>
    </section>
<!-- FIM  do CONTEÚDO -->
```

/styles.css

```
/* You can add global styles to this file, and also import other style files */

body {
  color: #555;
}

.box-header {
  color: inherit;
}

/*
 * Component: Place Info Box
 * -------------------
 */
.place-info-box {
  color: #555;
  display: block;
  min-height: 120px;
  background: #fff;
  width: 100%;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.1);
  border-radius: 2px;
  margin-bottom: 15px;
}
.place-info-box small {
  font-size: 14px;
}
.place-info-box .progress {
```

```
  background: rgba(0, 0, 0, 0.2);
  margin: 5px -10px 5px -10px;
  height: 2px;
}
.place-info-box .progress,
.place-info-box .progress .progress-bar {
  border-radius: 0;
}
.place-info .progress .progress-bar {
  background: #fff;
}
.place-info-box-icon {
  border-top-left-radius: 2px;
  border-top-right-radius: 0;
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 2px;
  display: block;
  float: left;
  /*height: 120px;*/
  width: 120px;
  text-align: center;
  background: rgba(0, 0, 0, 0.2);
}
.place-info-box-icon > img {
  max-width: 100%;
  /*border-right: 1px solid rgba(0, 0, 0, 0.1);*/
}
.place-info-box-content {
  padding: 5px 10px;
  margin-left: 120px;
}
.place-info-box-detail {
  display: block;
  font-size: 14px;
  margin-bottom: 5px;
  width: 100%;
}

.place-info-box-text {
  display: block;
  font-size: 18px;
  font-weight: bold;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  margin-bottom: 10px;
  text-transform: uppercase;
  width: 80%;
```

```
  float: left
}
.place-info-box-text {
}

.place-info-box-star {
  display: block;
  font-size: 18px;
  font-weight: bold;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  margin-bottom: 10px;
  width: 20%;
  float: right;
}
.place-info-box-more {
  display: block;
}
.progress-description {
  margin: 0;
}

/* Detail For restaurant */

.box-img-detail {
    width: 180px;
    height: 180px;
}

.box-body dl {
  font-size: 18px;
}

/*
 * Component: Menu Item
 * --------------------
 */
.menu-item-info-box {
  color: #555;
  display: block;
  min-height: 120px;
  background: #fff;
  width: 100%;
  box-shadow: 0 1px 1px rgba(0, 0, 0, 0.1);
  border-radius: 2px;
  margin-bottom: 15px;
}
```

```
.menu-item-info-box small {
  font-size: 14px;
}
.menu-item-info-box .progress {
  background: rgba(0, 0, 0, 0.2);
  margin: 5px -10px 5px -10px;
  height: 2px;
}
.menu-item-info-box .progress,
.menu-item-info-box .progress .progress-bar {
  border-radius: 0;
}
.menu-item-info .progress .progress-bar {
  background: #fff;
}
.menu-item-info-box-icon {
  border-top-left-radius: 2px;
  border-top-right-radius: 0;
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 2px;
  display: block;
  float: left;
  /*height: 120px;*/
  width: 120px;
  text-align: center;
  background: rgba(0, 0, 0, 0.2);
}
.menu-item-info-box-icon > img {
  max-width: 100%;
  /*border-right: 1px solid rgba(0, 0, 0, 0.1);*/
}
.menu-item-info-box-price {
  font-weight: bold;
  margin-right: 20px;
  font-size: 16px;
}

.menu-item-info-box-content {
  padding: 5px 10px;
  margin-left: 120px;
}

.menu-item-info-box-content a {
  cursor: pointer;
}

.menu-item-info-box-detail {
  display: block;
```

```
  font-size: 14px;
  margin-bottom: 15px;
  width: 100%;
}

.menu-item-info-box-text {
  display: block;
  font-size: 18px;
  font-weight: bold;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  margin-bottom: 10px;
  text-transform: uppercase;
  width: 80%;
}
.menu-item-info-box-text {
}

.menu-item-info-box-more {
  display: block;
}

/**
 buttons
*/

.danger {
  color: #dd4b39;
  display: inline;
}

/*Jumbotrons*/

.jumbotron {
  background-color: #fff;
  text-align:center;
}
.jumbotron h2 {
  font-size: 40px;
}

.welcome-jumbotron h1 {
  margin-bottom: 60px;
}

.welcome-jumbotron p {
  margin-bottom: 60px;
```

```
  font-size: 30px;
}

/* inputs */

.fancy-radio {
  position: absolute;
  opacity: 0;
}

/*Review*/
.review-score{
  font-size: 30px;
  font-weight: bold;
}

/*rstaurant detail*/

.detail-footer{
  padding:0px
}

.detail-footer > a{
  text-align: center;
  font-size: 16px;
  color: #555;
  width:50%;
  padding-top: 10px;
  padding-bottom: 10px;
}

.detail-footer > a:hover{
  outline: none;
  text-decoration: none;
}

.detail-active {
  border-bottom: 2px solid #dd4b39;
}

.has-error-block {
  color: #dd4b39;
}

/* Barra de busca de restaurantes */

.search-link{
  cursor: pointer;
```

```
  color: #555;
}

.search-link:hover{
  color: #dd4b39;
}
```

Seção 12

Vídeo 104

Observable ValueChanges de Reactive Forms

* Criando um auto-complete na pesquisa de restaurantes.

/restaurants.component.html

```
<!-- INíCIO do CONTEÚDO -->
    <section class="content-header">
      <h1>
        Todos os Restaurantes
        <a class="search-link pull-right"><i class="fa fa-search"></i></a>
      </h1>

      <!-- barra de busca -->
      <div class="row">
        <div class="col-xs-12"><!-- classe do bootstrap que exibe 12 colunas -->
          <form [formGroup]="searchForm">
            <div class="form-group">
              <input type="text" formControlName="searchControl" class="form-control" placeholder="O que você está procurando?"/>
            </div>
          </form>
        </div>
      </div>
    </section>

    <section class="content">
      <div class="row">
          <div *ngFor="let rest of restaurants" class="col-sm-6 col-xs-12">
            <mt-restaurant  [restaurant]="rest" ></mt-restaurant>
```

```
            <!-- perceba que precisaremos de @Input na propriedade restaurant pois o componente parent neste caso restaurants.component.html ira passar os valores atraves do ngFor -->
        </div>
                                    </div>
    </section>
<!-- FIM  do CONTEÚDO -->
```

/restaurants.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';
import {FormBuilder, FormControl, FormGroup} from '@angular/forms';
import 'rxjs/add/operator/switchMap';
@Component({
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html'
})
export class RestaurantsComponent implements OnInit {

// movido para a classe de servico restaurants.service.ts
// restaurants : Restaurant[] = [
//                  {
//                              id: "bread-bakery",
//                              name: "Bread & Bakery",
//                              category: "Bakery",
//                              deliveryEstimate: "25m",
//                              rating: 4.9,
//                              imagePath: "assets/img/restaurants/breadbakery.png"
//                  },
//                  {
//                              id: "burger-house",
//                              name: "Burger House",
//                              category: "Hamburgers",
//                              deliveryEstimate: "100m",
//                              rating: 3.5,
//                              imagePath: "assets/img/restaurants/burgerhouse.png"
//                  }
// ]
//
//  restaurants : Restaurant[]
//
//  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas
```

```
//                this.restaurants = this.restaurantsService.listarRestaurants();
//  }

 restaurants : Restaurant[] //este array sera carregado a partir do metodo subscribe(...)

  searchForm: FormGroup  //propriedade para representar o formulario
  searchControl: FormControl //propriedade para ouvir os valores digitados no formulario

  constructor(private restaurantsService : RestaurantsService, private fb: FormBuilder ) {
  }





 ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas


    this.searchControl = this.fb.control(''); //começa com o texto vazio
    this.searchForm = this.fb.group({ searchControl: this.searchControl });

   // this.searchControl.valueChanges.subscribe(searchTearm => console.log(searchTearm));//imprime na console
            this.searchControl.valueChanges.switchMap(dadoPesquisado=>
            this.restaurantsService.listarRestaurants(dadoPesquisado))
            .subscribe(restaurants => this.restaurants = restaurants);
                                //Através da propriedade searchControl do tipo FormControl
                                //utilzaremos o atributo valueChanges, que eh um Observable,
                                //onde faremos um subscribe e atraves do qual ouviremos aquilo
                                //que o usuario digitara no campo de entrada.
                                //Aqui, estamos nos inscrevendo no atributo valueChanges que
                                //eh um Observable.

   this.restaurantsService.listarRestaurants() .subscribe(restaurants => this.restaurants = restaurants);
  }

}
```

/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'
import {MEAT_API} from '../app.api'
import {Http} from '@angular/http'
import {Injectable} from '@angular/core'
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/operator/map' //importando um metodo especifico neste caso o map que permite extair um json
import 'rxjs/add/operator/catch' //importando um metodo especifico neste caso o catch para tratar erros
import {ErrorHandler} from '../app-error-handler'
import {MenuItem} from '../restaurant-detail/menu-item/menu-item.model'
```

```
@Injectable() //injetaremos o objeto Http
export class RestaurantsService {

//              rests : Restaurant[] = [
//                          {
//                                      id: "bread-bakery",
//                                      name: "Bread & Bakery",
//                                      category: "Bakery",
//                                      deliveryEstimate: "25m",
//                                      rating: 4.9,
//                                      imagePath: "assets/img/restaurants/breadbakery.png"
//                          },
//                          {
//                                      id: "burger-house",
//                                      name: "Burger House",
//                                      category: "Hamburgers",
//                                      deliveryEstimate: "100m",
//                                      rating: 3.5,
//                                      imagePath: "assets/img/restaurants/burgerhouse.png"
//                          }
//              ]
//              constructor(){
//
//              }
//              listarRestaurants() : Restaurant[] {//metodo que sera chamado no componente restaurants.component
//                          return this.rests;
//              }

            constructor(private http: Http){}

            listarRestaurants(valorPesquisado?: string) : Observable<Restaurant[]> { //valorPesquisado? => parametro opcional
                        return this.http.get(`${MEAT_API}/restaurants`, {params: {q: valorPesquisado}})//q ==> parametro generico para qualquer
                                    .map(response => response.json()) //propriedade de Restaurant exemplo: nome, catogory etc...
                                    .catch(ErrorHandler.handleError) //retorna um Observable<Restaurant[]>
            }

            listarRestaurantsA() : Observable<Restaurant[]> {
                        return this.http.get(`${MEAT_API}/restaurants`) //retorna um Observable<Response>
                                    .map(response => response.json()) //retorna um Observable<Restaurant[]>
                                    .catch(ErrorHandler.handleError)
            }

            listarRestaurantsZ() : Observable<Restaurant[]> {
                        return this.http.get(`${MEAT_API}/abcdef`) //provocando um erro atraves de uma pagina que nao existe
                                    .map(response => response.json()) //retorna um Observable<Restaurant[]>
                                    .catch(ErrorHandler.handleError)
            }
```

```
	restarurantById(id: string) : Observable<Restaurant>{
		return this.http.get(`${MEAT_API}/restaurants/${id}`)
			.map(response => response.json()) //retorna um Observable<Restaurant>
			.catch(ErrorHandler.handleError)
	}

	reviewsOfRestaurants(id: string) : Observable<any> {//perceba que nao temos um objeto do tipo Review

	//por isso o uso do "any" em Observable<any>
		return this.http.get(`${MEAT_API}/restaurants/${id}/reviews`)
			.map(response => response.json()) //retorna um Observable<any>
			.catch(ErrorHandler.handleError)
	}

	menuOfRestaurants(id: string) : Observable<MenuItem[]> {
		console.log(`Id de um componente parente:${id}`)


		return this.http.get(`${MEAT_API}/restaurants/${id}/menu`)
			.map(response => response.json())
			.catch(ErrorHandler.handleError)
	}

}
```

/restaurant/restaurant.model.ts

```
export interface Restaurant {
   id: string
   name: string
   category: string
   deliveryEstimate: string
   rating: number
   imagePath: string
   hours?: string //opcional
   about?: string //opcional

}
```

Seção 12

Vídeo 105

Operadores debounceTime e distinctUntilChanged

Melhorando a busca do auto-complete, evitando muitos acessos à base de dados.

/restaurants/restaurants.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';
import {FormBuilder, FormControl, FormGroup} from '@angular/forms';
import 'rxjs/add/operator/switchMap';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/debounceTime';
import 'rxjs/add/operator/distinctUntilChanged';

@Component({
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html'
})
export class RestaurantsComponent implements OnInit {

// movido para a classe de servico restaurants.service.ts
// restaurants : Restaurant[] = [
//		{
//			id: "bread-bakery",
//			name: "Bread & Bakery",
//			category: "Bakery",
//			deliveryEstimate: "25m",
//			rating: 4.9,
//			imagePath: "assets/img/restaurants/breadbakery.png"
//		},
//		{
//			id: "burger-house",
//			name: "Burger House",
//			category: "Hamburgers",
//			deliveryEstimate: "100m",
//			rating: 3.5,
//			imagePath: "assets/img/restaurants/burgerhouse.png"
//		}
// ]
//
//  restaurants : Restaurant[]
//
//  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas
//		this.restaurants = this.restaurantsService.listarRestaurants();
//  }

  restaurants : Restaurant[] //este array sera carregado a partir do metodo subscribe(...)
```

```
  searchForm: FormGroup  //propriedade para representar o formulario
  searchControl: FormControl //propriedade para ouvir os valores digitados no formulario

  constructor(private restaurantsService : RestaurantsService, private fb: FormBuilder ) {
  }

  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas


    this.searchControl = this.fb.control(''); //começa com o texto vazio
    this.searchForm = this.fb.group({ searchControl: this.searchControl });

   // this.searchControl.valueChanges.subscribe(searchTearm => console.log(searchTearm));//imprime na console
    this.searchControl.valueChanges
        .debounceTime(500)//tempo entre dois eventos - no caso digitacao depois de 500ms um novo acesso a base eh feita
        .distinctUntilChanged()//agora estamos dizendo que as buscas devem ser feitas quando o dado informado for diferente um dos outros
        .do(dadoPesquisado => console.log(`q=${dadoPesquisado}`))//apenas para imprimir na console - pode ser removido
        .switchMap(dadoPesquisado => this.restaurantsService.listarRestaurants(dadoPesquisado))
        .subscribe(restaurants => this.restaurants = restaurants);
                //Através da propriedade searchControl do tipo FormControl
                //utilzaremos o atributo valueChanges, que eh um Observable,
                //onde faremos um subscribe e atraves do qual ouviremos aquilo
                //que o usuario digitara no campo de entrada.
                //Aqui, estamos nos inscrevendo no atributo valueChanges que
                //eh um Observable.

    this.restaurantsService.listarRestaurants()
                                .subscribe(restaurants => this.restaurants = restaurants);
  }

}
```

/restaurants/restautants.componente.html

```
<!-- INíCIO do CONTEÚDO -->
    <section class="content-header">
      <h1>
        Todos os Restaurantes
        <a class="search-link pull-right">
          <small>"{{iptSearch.value}}"</small>
        <i class="fa fa-search"></i></a>
      </h1>

      <!-- barra de busca -->
      <div class="row">
        <div class="col-xs-12"><!-- classe do bootstrap que exibe 12 colunas -->
          <form [formGroup]="searchForm">
```

```
            <div class="form-group">
              <input type="text" #iptSearch formControlName="searchControl" class="form-control" placeholder="O que você está procurando?"/>
            </div>
          </form>
        </div>
      </div>
    </section>


    <section class="content">
      <div class="row">
        <div *ngFor="let rest of restaurants" class="col-sm-6 col-xs-12">
          <mt-restaurant  [restaurant]="rest" ></mt-restaurant>
          <!-- perceba que precisaremos de @Input na propriedade restaurant pois o componente parent neste caso restaurants.component.html ira passar os valores atraves do ngFor -->
        </div>
                                </div>
    </section>
<!-- FIM  do CONTEÚDO →
```

Seção 12

Vídeo 106

Prevenindo a Interrupção do Observable ValueChanges

Evitando que o erro quebre o nosso Observable para isso faremos um tratamento com o operador catch onde iremos capturar o erro e retornaremos um outro Observable com um array vazio. Por exemplo o servidor de banco de dados não responda mais a requisições.

Observação: o Observable do Rxjs é diferente do Observable do Http, pois o do http após a resposta que vem através do objeto response este Observable é destruído enquanto o do Rxjs a stream permanece aberta.

/restaurants/restaurants.component.ts

```
import { Observable } from 'rxjs/Observable';
import { Component, OnInit, Input } from '@angular/core';
import { Restaurant } from './restaurant/restaurant.model';
import { RestaurantsService } from './restaurants.service';
import {FormBuilder, FormControl, FormGroup} from '@angular/forms';
import 'rxjs/add/operator/switchMap';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/debounceTime';
import 'rxjs/add/operator/distinctUntilChanged';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/from';

@Component({
```

```
  selector: 'mt-restaurants',
  templateUrl: './restaurants.component.html'
})
export class RestaurantsComponent implements OnInit {

// movido para a classe de servico restaurants.service.ts
// restaurants : Restaurant[] = [
//                {
//                                id: "bread-bakery",
//                                name: "Bread & Bakery",
//                                category: "Bakery",
//                                deliveryEstimate: "25m",
//                                rating: 4.9,
//                                imagePath: "assets/img/restaurants/breadbakery.png"
//                },
//                {
//                                id: "burger-house",
//                                name: "Burger House",
//                                category: "Hamburgers",
//                                deliveryEstimate: "100m",
//                                rating: 3.5,
//                                imagePath: "assets/img/restaurants/burgerhouse.png"
//                }
// ]
//
//  restaurants : Restaurant[]
//
//  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas
//                this.restaurants = this.restaurantsService.listarRestaurants();
//  }

  restaurants : Restaurant[] //este array sera carregado a partir do metodo subscribe(...)

  searchForm: FormGroup  //propriedade para representar o formulario
  searchControl: FormControl //propriedade para ouvir os valores digitados no formulario

  constructor(private restaurantsService : RestaurantsService, private fb: FormBuilder ) {
  }

  ngOnInit() {//metodo de ciclo de vida que eh executado depois que todas as dependencias forem resolvidas

    this.searchControl = this.fb.control(''); //começa com o texto vazio
    this.searchForm = this.fb.group({ searchControl: this.searchControl });

  // this.searchControl.valueChanges.subscribe(searchTearm => console.log(searchTearm));//imprime na console
    this.searchControl.valueChanges
        .debounceTime(500)//tempo entre dois eventos - no caso digitacao depois de 500ms um novo acesso a base eh feita
        .distinctUntilChanged()//agora estamos dizendo que as buscas devem ser feitas quando o dado informado for diferente um dos outros
```

```
        .do(dadoPesquisado => console.log(`q=${dadoPesquisado}`))//apenas para imprimir na console - pode ser removido
        .switchMap(dadoPesquisado => this.restaurantsService.listarRestaurants(dadoPesquisado)//this.restaurantsService.listarRestaurants(dadoPesquisado) este
                                                                    //Observable pode quebrar caso o banco de dados
                                                                    //fique sem acesso por isso o tratamento
        .catch(error=>Observable.from([]))//retornando um Observable apos um erro com array vazio
        .subscribe(restaurants => this.restaurants = restaurants);
                //Através da propriedade searchControl do tipo FormControl
                //utilzaremos o atributo valueChanges, que eh um Observable,
                //onde faremos um subscribe e atraves do qual ouviremos aquilo
                //que o usuario digitara no campo de entrada.
                //Aqui, estamos nos inscrevendo no atributo valueChanges que
                //eh um Observable.

    this.restaurantsService.listarRestaurants()
                                .subscribe(restaurants => this.restaurants = restaurants);
  }

}
```

/restaurants/restaurants.component.html

```
<!-- INíCIO do CONTEÚDO -->
    <section class="content-header">
      <h1>
        Todos os Restaurantes
        <a class="search-link pull-right">
          <small>"{{iptSearch.value}}"</small>
        <i class="fa fa-search"></i></a>
      </h1>

      <!-- barra de busca -->
      <div class="row">
        <div class="col-xs-12"><!-- classe do bootstrap que exibe 12 colunas -->
          <form [formGroup]="searchForm">
            <div class="form-group">
              <input type="text" #iptSearch formControlName="searchControl" class="form-control" placeholder="O que você está procurando?"/>
            </div>
          </form>
        </div>
      </div>
    </section>

    <section class="content">
      <div class="row">
        <div *ngFor="let rest of restaurants" class="col-sm-6 col-xs-12">
          <mt-restaurant  [restaurant]="rest" ></mt-restaurant>
```

```
            <!-- perceba que precisaremos de @Input na propriedade restaurant pois o componente parent neste caso restaurants.component.html ira passar os valores atraves do ngFor -->
          </div>
          <div *ngIf="restaurants?.length === 0" class="col-sm-6 col-xs-12">
            <p class="lead text-center">Não foram encontrados itens para a busca</p>
          </div>
        </div>
      </section>
<!-- FIM  do CONTEÚDO -->
```

Seção 13

Vídeo 107

Angular 4.3 apresenta um novo módulo para acesso Http chamado HttpClient. Este novo módulo apresenta tipagem nos métodos de serviço e HttpInterceptors que é capaz de modificar o request antes mesmo de ser processado.

HttpClient x Http

Principais mudanças:

*** Mudanças nos Imports:**

Http (na versão antiga)

```
import {HttpModule} from '@angular/http'
@NgModule ({
  imports: [HttpModule]
})
export class AppModule {}
```

HttpClient (na versão nova)

```
import {HttpClientModule} from '@angular/common/http'
@NgModule ({
  imports: [HttpClientModule]
})
export class AppModule {}
```

Mudança no uso:

Http (na versão antiga)

```
import {Http} from '@angular/http'
@Injectable ()
export class MyService {
   constructor (private http: Http) {}
}
```

HttpClient (na versão nova)

```
import {HttpClient} from '@angular/common/http'
@Injectable ()
export class MyService {
   constructor (private http: HttpClient) {}  //não injetamos Http, mas HttpClient
}
```

* O novo serviço possui os mesmos métodos para as requisições Http. São eles: Get, Post, Delete, Put … entre outros,

* Não é mais necessário utilizar o operador **map(…)** para retornar o corpo da resposta, isto agora é feito automaticamente retornando o **body**,

* As respostas agora são tipadas.

Usando Http:

Http (na versão antiga)

```
public restaurantById(id: string): Observable<Restaurant> {
  return this.http.get(`restaurants/${id}`).map(response => response.json());
}
```

HttpClient (na versão nova)

```
public restaurantById(id: string): Observable<Restaurant> {
  return this.http.get<Restaurant>(`restaurants/${id}`); //não usamos o operador map(…) e usamos tipagem
}
```

Como acessar mais informações da Resposta

```
public restaurants(): Observable<Restaurant[]> {
  return this.http.get<Restaurant[]>(`restaurants`,{observe: 'response'})
                    .subscribe(resp => {
                         resp.headers.get('X-PageSize') //custom header
                        this.rests = resp.body //Restaurant[]
})
```

Seção 13

Vídeo 108

Refactoring para o novo módulo HttpClient

Alterando a Angular para a versão 4.3

1) Alterando o package.json

Upgrade para Angular 4.3

Dependências dos pacotes que devem ficar em package.json:

FONTE: https://github.com/cod3rcursos/meat-app-starter

```
"dependencies": {
   "@angular/animations": "4.3.3",
   "@angular/common": "4.3.3",
   "@angular/compiler": "4.3.3",
   "@angular/core": "4.3.3",
   "@angular/forms": "4.3.3",
   "@angular/platform-browser": "4.3.3",
   "@angular/platform-browser-dynamic": "4.3.3",
   "@angular/router": "4.3.3",
   "admin-lte": "2.3.11",
   "core-js": "2.4.1",
   "font-awesome": "4.7.0",
   "intl": "1.2.5",
   "jquery": "3.1.1",
   "reflect-metadata": "0.1.10",
```

```
    "rxjs": "5.4.2",
    "ts-helpers": "1.1.2",
    "web-animations-js": "2.2.5",
    "zone.js": "0.8.16"
  },
  "devDependencies": {
    "@angular/cli": "1.2.7",
    "@angular/compiler-cli": "4.3.3",
    "@types/jasmine": "2.5.53",
    "@types/express": "4.0.37",
    "@types/jsonwebtoken": "7.2.3",
    "@types/node": "7.0.5",
    "codelyzer": "3.1.2",
    "jasmine-core": "2.7.0",
    "jasmine-spec-reporter": "4.1.1",
    "json-server": "0.12.0",
    "jsonwebtoken": "7.4.1",
    "karma": "1.7.0",
    "karma-chrome-launcher": "2.2.0",
    "karma-cli": "1.0.1",
    "karma-jasmine": "1.1.0",
    "karma-remap-istanbul": "0.6.0",
    "protractor": "5.1.2",
    "ts-node": "3.3.0",
    "tslint": "5.5.0",
    "typescript": "2.4.2",
    "webdriver-manager": "12.0.6"
  }
```

2) Remova o arquivo package-lock.json, pois ele será criado novamente automaticamente com o comando npm install.

3) Remova o conteúdo da pasta node_modules

4) > npm install

5) > ng build

6) Consertar os erros de **imports** devido à atualização do Angular 4.0 para 4.3

7) Altere os servicos abaixo:

/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
//import { HttpModule } from '@angular/http';
import { HttpClientModule } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                        //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                        //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim
```

```
//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router'//Import necessario para fazer o carregamento
                                         //em background dos modulos Lazy

//import {LocationStrategy, HashLocationStrategy} from '@angular/common'



@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    //HttpModule, //==> exemplo de Core Module. Veja o github do Angular //Angular 4.0
    HttpClientModule, //Angular 4.3
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
    RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
    //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                              //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular nao
                  //sendo necessario portanto fazer imports novamente neste modulo.


    SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou nao.Neste caso
                                 //estamos importando. Veja o Modulo OrderModule onde nao importamos os
                                 //serviços apenas os componentes declarados no SharedModule.
    //OrderModule => este modulo nao foi inserido aqui, pois nao queremos
```

```
    //que o mesmo seja carregado na inicializacao do projeto, porem faremos
    //este carregamento como lazy loading no app.routes.ts

    //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
  ],

   providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],
   //providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}], utilizado para evitar problemas de navegação ao
   //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url
   //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
   //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url

  //transferido para CoreModule
   //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

/restaurants/restaurants.service.ts

```
import {Restaurant} from './restaurant/restaurant.model'

import {MEAT_API} from '../app.api'

import {HttpClient,HttpParams} from '@angular/common/http'

import {Injectable} from '@angular/core'

import {Observable} from 'rxjs/Observable'

import 'rxjs/add/operator/map' //importando um metodo especifico neste caso o map que permite extair um json

import 'rxjs/add/operator/catch' //importando um metodo especifico neste caso o catch para tratar erros

import {ErrorHandler} from '../app-error-handler'

import {MenuItem} from '../restaurant-detail/menu-item/menu-item.model'

@Injectable() //injetaremos o objeto Http
export class RestaurantsService {


//                rests : Restaurant[] = [
//                                {
//                                                id: "bread-bakery",
```

```
//                                  name: "Bread & Bakery",
//                                  category: "Bakery",
//                                  deliveryEstimate: "25m",
//                                  rating: 4.9,
//                                  imagePath: "assets/img/restaurants/breadbakery.png"
//                          },
//                          {
//                                  id: "burger-house",
//                                  name: "Burger House",
//                                  category: "Hamburgers",
//                                  deliveryEstimate: "100m",
//                                  rating: 3.5,
//                                  imagePath: "assets/img/restaurants/burgerhouse.png"
//                          }
//              ]
//              constructor(){
//
//              }
//              listarRestaurants() : Restaurant[] {//metodo que sera chamado no componente restaurants.component
//                          return this.rests;
//              }



                constructor(private http: HttpClient){} //Angular 4.3

                listarRestaurants(valorPesquisado?: string) : Observable<Restaurant[]> { //valorPesquisado? => parametro opcional


                            //HttpParams => eh um objeto imutavel, logo quando setamos um parametro no mesmo seja com set(...)
                            //ou build(...) na realidade nao estamos setando no objeto original, mas uma copia do mesmo por isso
                            //params.set(...) nao funciona. Veja a forma correta abaixo.
                            let params: HttpParams = undefined

                            if(valorPesquisado){
                              params = new HttpParams().set('q', valorPesquisado) //retorna uma outra instancia de HttpParams
                              //params = new HttpParams().append('q', valorPesquisado)
                            }

                            //if(valorPesquisado){ //nao funciona de acordo com a forma abaixo
                            //  params = new HttpParams() //objeto imutavel
                            //  params.set('q', valorPesquisado) //retorna uma outra instancia de HttpParams
                            //}


                            return this.http.get<Restaurant[]>(`${MEAT_API}/restaurants`, {params})//q ==> parametro generico para qualquer
                                                        //.map(response => response.json()) //propriedade de Restaurant exemplo: nome, catogory etc...
                                                        //.catch(ErrorHandler.handleError) //retorna um Observable<Restaurant[]>
                }

                listarRestaurantsA() : Observable<Restaurant[]> {
```

```
        return this.http.get<Restaurant[]>(`${MEAT_API}/restaurants`) //retorna um Observable<Response>
                                                //.map(response => response.json()) //retorna um Observable<Restaurant[]>
                                                //.catch(ErrorHandler.handleError)
        }

        listarRestaurantsZ() : Observable<Restaurant[]> {
        return this.http.get<Restaurant[]>(`${MEAT_API}/abcdef`) //provocando um erro atraves de uma pagina que nao existe
                                                //.map(response => response.json()) //retorna um Observable<Restaurant[]>
                                                //.catch(ErrorHandler.handleError)
        }

        restarurantById(id: string) : Observable<Restaurant>{
        return this.http.get<Restaurant>(`${MEAT_API}/restaurants/${id}`) //Aqui, estamos retornando um Observable<Restaurant> por isso fazemos get<Restaurant>
                                                //.map(response => response.json()) //retorna um Observable<Restaurant>
                                                //.catch(ErrorHandler.handleError)
        }

        reviewsOfRestaurants(id: string) : Observable<any>{//perceba que nao temos um objeto do tipo Review
                                                //por isso o uso do "any" em Observable<any>
        return this.http.get(`${MEAT_API}/restaurants/${id}/reviews`)
                                                //.map(response => response.json()) //retorna um Observable<any>
                                                //.catch(ErrorHandler.handleError)
        }

        menuOfRestaurants(id: string) : Observable<MenuItem[]>{
        console.log(`Id de um componente parente:${id}`)

        return this.http.get<MenuItem[]>(`${MEAT_API}/restaurants/${id}/menu`)
                                                //.map(response => response.json())
                                                //.catch(ErrorHandler.handleError)
        }

//constructor(private http: Http){} //Angular 4.0

//    listarRestaurants(valorPesquisado?: string) : Observable<Restaurant[]> { //valorPesquisado? => parametro opcional
//                    return this.http.get(`${MEAT_API}/restaurants`, {params: {q: valorPesquisado}})//q ==> parametro generico para qualquer
//                            .map(response => response.json()) //propriedade de Restaurant exemplo: nome, catogory etc...
//                            .catch(ErrorHandler.handleError) //retorna um Observable<Restaurant[]>
//          }

//          listarRestaurantsA() : Observable<Restaurant[]> {
//                    return this.http.get(`${MEAT_API}/restaurants`) //retorna um Observable<Response>
//                            .map(response => response.json()) //retorna um Observable<Restaurant[]>
//                            .catch(ErrorHandler.handleError)
//          }

 //         listarRestaurantsZ() : Observable<Restaurant[]> {
 //         return this.http.get(`${MEAT_API}/abcdef`) //provocando um erro atraves de uma pagina que nao existe
```

```
//                      .map(response => response.json()) //retorna um Observable<Restaurant[]>
//                      .catch(ErrorHandler.handleError)
//          }

//          restarurantById(id: string) : Observable<Restaurant>{
//                      return this.http.get(`${MEAT_API}/restaurants/${id}`)
//                              .map(response => response.json()) //retorna um Observable<Restaurant>
//                              .catch(ErrorHandler.handleError)
//          }

//          reviewsOfRestaurants(id: string) : Observable<any> {//perceba que nao temos um objeto do tipo Review
//
            //por isso o uso do "any" em Observable<any>
//                      return this.http.get(`${MEAT_API}/restaurants/${id}/reviews`)
//                              .map(response => response.json()) //retorna um Observable<any>
//                              .catch(ErrorHandler.handleError)
//          }

//          menuOfRestaurants(id: string) : Observable<MenuItem[]> {
//                      console.log(`Id de um componente parente:${id}`)
//
//                      return this.http.get(`${MEAT_API}/restaurants/${id}/menu`)
//                              .map(response => response.json())
//                              .catch(ErrorHandler.handleError)
//          }

}
```

/order/order.service.ts

```
import {Injectable} from '@angular/core';
import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';
import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';
import { Order, OrderItem } from './order.model';
//import { Http } from '@angular/http'; // Angular 4.0
import { HttpClient } from '@angular/common/http'; // Angular 4.3
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';
import {MEAT_API} from '../app.api';
//import { Headers, RequestOptions } from '@angular/http'; // Angular 4.0
//import { Headers, RequestOptions } from '@angular/common/http' // Angular 4.3 ja esta embutido no HttpClient

@Injectable() //injetando o servico ShoppingCartService
export class OrderService {

 // constructor(private cartService: ShoppingCartService, private http: Http){
 //}
```

```
constructor(private cartService: ShoppingCartService, private http: HttpClient){

}

cartItems() : CartItem[]{
  return this.cartService.items
}

increaseQty(item : CartItem){
  return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
}

decreaseQty(item : CartItem){
  return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
}

remove(item : CartItem){
    this.cartService.removeItem(item)
 }

 itemsValue(): number {  //valor total da compra dos items
     return this.cartService.total()
 }

 clear(){
   this.cartService.clear()
 }



 // Angular 4.3
checkOrder(order: Order): Observable<string>{
  //quando o nosso objeto Order for enviando para o servidor
  //o mesmo produzira um id do tipo string para o objeto Order,
  //o qual representa a nossa compra.
  //Neste metodo utilizaremso POST uma vez que estamos inserindo
  //dados na nossa base de dado json.
  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
  //JSON.stringify(order) =>cria a representacao textual do objeto Order
  //headers.append("Content-Type","application/json") => indica o formato do dado
  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
  //que fora salvo no nosso arquivo db.json
  //Para resumir quando usamos um objeto Http no servico estamos de fato
  //configurando o nosso objeto Observable
```

```
  //const headers = new Headers()
  //headers.append("Content-Type","application/json")//nome do header, valor
  //return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
  return this.http.post<Order>(`${MEAT_API}/orders`,order) //t<Order> este eh o tipo que iremos receber no post
  //.map(response=> response.json())
  .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                    //para retornar apenas o ID da compra
}

// Angular 4.0
//checkOrder(order: Order): Observable<string>{
//                  //quando o nosso objeto Order for enviando para o servidor
//                  //o mesmo produzira um id do tipo string para o objeto Order,
//                  //o qual representa a nossa compra.
//                  //Neste metodo utilizaremso POST uma vez que estamos inserindo
//                  //dados na nossa base de dado json.
//                  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
//                  //JSON.stringify(order) =>cria a representacao textual do objeto Order
//                  //headers.append("Content-Type","application/json") => indica o formato do dado
//                  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
//                  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
//                  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
//                  //que fora salvo no nosso arquivo db.json
//                  //Para resumir quando usamos um objeto Http no servico estamos de fato
//                  //configurando o nosso objeto Observable
//   const headers = new Headers()
//   headers.append("Content-Type","application/json")//nome do header, valor

//   return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
//                .map(response=> response.json())
//                .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
//                                  //para retornar apenas o ID da compra
// }
}
```

/order/order.model.ts

```
class Order{
                  constructor( //perceba que este construtor tem os mesmos atributos que
                        //surgem no debugg da console do browser ao se clicar no
                        //botao "Concluir Pedido" localizado no template order.component.html
          public address: string,
                public number: number,
          public optionalAddress: string,
          public paymentOption: string,
          public orderItems: OrderItem[] = [],
          public id?: string //parametro opcional
          ){}
}
```

```
class OrderItem{
		constructor(public quantity: number, public menuId: string ){}
}


export {Order, OrderItem}
```

/app-error-handler.ts

```
import {HttpErrorResponse} from '@angular/common/http' // Angular 4.3
import {Observable} from 'rxjs/Observable'
import 'rxjs/add/observable/throw'
export class ErrorHandler { //criando uma classe de erro
static handleError(error : HttpErrorResponse | any){
		let errorMessage: string
		if(error instanceof HttpErrorResponse){
			const body = error.error
			errorMessage = `${error.url} : ${error.status} - ${error.statusText || ''} ${body}`
		}else{
			errorMessage = error.message ? error.message : error.toString()
		}
		console.log(">>>==================> " + errorMessage)
		return Observable.throw(errorMessage);


		}

// Angular 4.0
//import {Response} from '@angular/http' // Angular 4.0
//import {Observable} from 'rxjs/Observable'
//import 'rxjs/add/observable/throw'
//export class ErrorHandler { //criando uma classe de erro
//static handleError(error : Response | any){ // Angular 4.0
//		let errorMessage: string
//		if(error instanceof Response){
//			errorMessage = `Erro [ ${error.status} ] ao acessar a URL [ ${error.url} ] - [ ${error.statusText} ]`
//		}else{
//			errorMessage = error.toString()
//		}
//		console.log(">>>==================> " + errorMessage)
//		return Observable.throw(errorMessage);


//		}

}
```

Seção 14

Vídeo 109

Personalizando o Backend

Nesta seção vamos personalizar a inicialização do json-server, logo não utilizaremos “json-server db.json” uma vez que utilizaremos o script contido no server.js para inicializar o json-server e a chamada no browser será através do protocolo https. Nesta seção também criaremos uma Aplicação em Express.

backend/server.js

```
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares) //aqui temos uma array de funções de callbacks

// Add custom routes before JSON Server router
//com o get estamos configurando uma rota com apenas uma função callback
server.get('/echo', (req, res) => {   //server.use('/echo', (req, res) => ….) com o método use será permitido qualquer verbo HTTP
  res.jsonp(req.query)
})

// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)  //Por padrão uma Aplicação em Express não faz o parse do body = extrair os dados do request,
server.use((req, res, next) => {       //logo para fazer o parse utilizamos a  middleware “jsonServer.bodyParser” transformando em
  if (req.method === 'POST') {       //um objeto JSON.
    req.body.createdAt = Date.now()
  }
  // Continue to JSON Server router
  next()
})

// Use default router
server.use(router)
server.listen(3001, () => {  //aqui estamos subindo a aplicação na porta 3001
  console.log('JSON Server is running')
})
```

backend/keys/cert.pem

-----BEGIN CERTIFICATE-----

MIIDtTCCAp2gAwIBAgIJAKDr6YbZ+6KXMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV

BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX

aWRnaXRzIFB0eSBMdGQwHhcNMTcwODA4MDI0NzA3WhcNMTcwOTA3MDI0NzA3WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl28gI9ABf
dN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+FGA9vXy4L
2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS0CRRYZGX
w08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrgsGgiH0Hn
jOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIwd3KhL6e5
aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABo4GnMIGkMB0GA1UdDgQWBBQebMNu
xfSChegKLYv2Dp7aVPYIRjB1BgNVHSMEbjBsgBQebMNuxfSChegKLYv2Dp7aVPYI
RqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNV
BAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAKDr6YbZ+6KXMAwGA1UdEwQF
MAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAIt5OHCBsYuIvB1UsI9rFkRunQ9eCORD
gk/dg6mtQaCUKqJRoQq+Zqi+y/z+Jxl1pgzudaOPwX/wLDM31PGWZbwzABR1uIR8
aQ/sDLw0yUdGvtUnrZOK4CVNQgl+zXNTmpmEx1JAT7xvXb5OPahdi/AYqVW6SWET
QCiRCNsqv3xqyStAURi8B0BVVJUTNbn9+CkBU7FDZpBjWCKUsMfm/4dm/u3eTxIS
oOf70F25k5Ws5I11usvlQY6/I17aGXU29voKws0v7iClFwqtOt8nntGi5I6OQrd5
2PvOp/kA2TPxi83nP0/WT0pzvM/ykzbxh4oKmCIzp6wXx8JaKIr4Xr0=
-----END CERTIFICATE-----

backend/keys/key.pem

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl
28gI9ABfdN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+F
GA9vXy4L2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS
0CRRYZGXw08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrg
sGgiH0HnjOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIw
d3KhL6e5aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABAoIBABvTp31tkzLLy4qo
yYbWsJ4w7vw0womd0U5RRfwKa6lalDHvVFaBrsUbSIDhagIpdNQdDBWrNuEO/Cwt
x12AIIP1meN/gXvPdKds3cYkL2g/HGE25UMI8FghuI6evnmb+aLHcsHbxsf45ze+
lERyzF3AUxQ/UaNmgIqEPIw34GYqiQB4ih6H1tQrKWVuvMtNc7o1kYOuTjR581h9
gXDKxGPkFyn2Y6xw1FDevAjzLvb4fJ/PLcTtEzbpX0+Jwpw/sPO5Vu9aj5Y/WpcF
k94ypnd8hR8J7kvjUmoPJwxHZ8wU75OCq+9KvIq8AU8LARVv9cK26drmrGEH5YEu
Dgy7uAECgYEA+l9yJZj1tzqYUoy8P70+jz5Ixrk771JForKBYv/rQL8gtpCOdVOy
2SrjgbvRKsf6mklTmJy3ljLNVuoFxLaiBauOH7zlC+twucdZhWOdKfj92WFr78d6
Txp1E2jUYijw+SVnmOm71IFqF9A1+vGnX+MV9awCEbzck5Km1IodsXkCgYEAx47f
atR/jRxmDKYhvU+i27fP8BxessBIW+jgP4qE+YaFNdUG0p2zjnco1G66DzAntiFB
pjYFzZFe6kFUIsHNeag3i0HQqqfTsYeOuFo5wA2yHxqhGj//aBlvjl2nScm4nNxL
e+IiT5lnYixsMeXp0wsXDjVRL+qI0ddCxMSamKMCgYEAuyOGNTX8SyUf32vT9Sxl
Szn5uZeQhdQPuokEDtRMqctyDyNDZ8zUvF/DRx/0Oq7Y0GOIeWSCEGEOfC1VfHAp
8GUHSQ+uf9dOHX40mnAXc9zpGIcp30JRWDjlU0minXAOlZrwsSnbO/4zJ3N4NYZU
XWZl7ofP0CRDFoah7fECOrkCgYAWeZPV/0rCRMGFiSxhmtd3sXL80uUjcp5zo32m
Cl7Bj5HvJyfWjqZLVEZH7pSL41uFmqc/JfRvW4D6FxN3uqX+nqBq0aXDopUn+GFt
h9PbOm+mKySbxTtpqPu0Z/3QRKTDHdhugmBk4J0RJHmnAQLdnvEgy4ZyRBLWt0yw
xoe2ywKBgHuDdsU4CW/FW3vNqMe1HLuIBeqHCWg5XSgl/MHLiflJ+tQxbKLTXtyU
z/sNijftjwWrs+NrzJvTWnlXSyLGFkKK/TotkHBDsEY5z+Rhu0Ndm20btkD4j7Hf
AeCSGnwPKKLLtNh6pVngBDiUbh+GkN+awwETQCDLgxqB94LJN02L
-----END RSA PRIVATE KEY-----

No terminal faça:

>node backend/dist/server //sem extensão *.js - inicializando o json-server a partir do script contido em server.js

No browser faça:

https://localhost:3001

// json-server db.json //db.json => arquivo *.json que esta simulando nosso banco de dados

// json-server -p ***4000*** db.json //rodando o json-server em uma ***porta específica***.

Seção 14

Vídeo 110

Utilizando HTTPS e Typescript (Backend com Https e Type)

Nesta seção utilizaremos o **nodemon** que é um utilitário que irá monitorar aquivos específicos e em caso de mudanças destes o servidor será reiniciado.

1) Crie o projeto: >ng new my-app-cli - -spec=false

2) Crie uma pasta de nome backend na raiz do projeto, será nesta pasta que criaremos nossos arquivos server.ts e tsconfig.json

3) Crie uma pasta de nome keys em backend.

3.1) Na pasta backend/keys/, crie os arquivos cert.pem e key.pem com os respectivos conteúdos abaixo:

backend/keys/cert.pem

-----BEGIN CERTIFICATE-----

MIIDtTCCAp2gAwIBAgIJAKDr6YbZ+6KXMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV

BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX

aWRnaXRzIFB0eSBMdGQwHhcNMTcwODA4MDI0NzA3WhcNMTcwOTA3MDI0NzA3WjBF

MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50

ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB

CgKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl28gI9ABf

dN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+FGA9vXy4L

2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS0CRRYZGX

w08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrgsGgiH0Hn

jOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIwd3KhL6e5

aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABo4GnMIGkMB0GA1UdDgQWBBQebMNu

xfSChegKLYv2Dp7aVPYIRjB1BgNVHSMEbjBsgBQebMNuxfSChegKLYv2Dp7aVPYI

RqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNV

BAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAKDr6YbZ+6KXMAwGA1UdEwQF

MAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAIt5OHCBsYuIvB1UsI9rFkRunQ9eCORD

gk/dg6mtQaCUKqJRoQq+Zqi+y/z+Jxl1pgzudaOPwX/wLDM31PGWZbwzABR1uIR8

aQ/sDLw0yUdGvtUnrZOK4CVNQgl+zXNTmpmEx1JAT7xvXb5OPahdi/AYqVW6SWET

QCiRCNsqv3xqyStAURi8B0BVVJUTNbn9+CkBU7FDZpBjWCKUsMfm/4dm/u3eTxIS

oOf70F25k5Ws5I11usvlQY6/I17aGXU29voKws0v7iClFwqtOt8nntGi5I6OQrd5

2PvOp/kA2TPxi83nP0/WT0pzvM/ykzbxh4oKmCIzp6wXx8JaKIr4Xr0=

-----END CERTIFICATE-----

backend/keys/key.pem

-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl

28gI9ABfdN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+F

GA9vXy4L2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS

0CRRYZGXw08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrg

sGgiH0HnjOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIw

d3KhL6e5aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABAoIBABvTp31tkzLLy4qo

yYbWsJ4w7vw0womd0U5RRfwKa6lalDHvVFaBrsUbSIDhagIpdNQdDBWrNuEO/Cwt

x12AIIP1meN/gXvPdKds3cYkL2g/HGE25UMI8FghuI6evnmb+aLHcsHbxsf45ze+

lERyzF3AUxQ/UaNmgIqEPIw34GYqiQB4ih6H1tQrKWVuvMtNc7o1kYOuTjR581h9

gXDKxGPkFyn2Y6xw1FDevAjzLvb4fJ/PLcTtEzbpX0+Jwpw/sPO5Vu9aj5Y/WpcF

k94ypnd8hR8J7kvjUmoPJwxHZ8wU75OCq+9KvIq8AU8LARVv9cK26drmrGEH5YEu

Dgy7uAECgYEA+l9yJZj1tzqYUoy8P70+jz5Ixrk771JForKBYv/rQL8gtpCOdVOy

2SrjgbvRKsf6mklTmJy3ljLNVuoFxLaiBauOH7zlC+twucdZhWOdKfj92WFr78d6

Txp1E2jUYijw+SVnmOm71IFqF9A1+vGnX+MV9awCEbzck5Km1IodsXkCgYEAx47f

atR/jRxmDKYhvU+i27fP8BxessBIW+jgP4qE+YaFNdUG0p2zjnco1G66DzAntiFB

pjYFzZFe6kFUIsHNeag3i0HQqqfTsYeOuFo5wA2yHxqhGj//aBlvjl2nScm4nNxL

e+IiT5lnYixsMeXp0wsXDjVRL+qI0ddCxMSamKMCgYEAuyOGNTX8SyUf32vT9Sxl

Szn5uZeQhdQPuokEDtRMqctyDyNDZ8zUvF/DRx/0Oq7Y0GOIeWSCEGEOfC1VfHAp

8GUHSQ+uf9dOHX40mnAXc9zpGIcp30JRWDjlU0minXAOlZrwsSnbO/4zJ3N4NYZU

XWZl7ofP0CRDFoah7fECOrkCgYAWeZPV/0rCRMGFiSxhmtd3sXL80uUjcp5zo32m

Cl7Bj5HvJyfWjqZLVEZH7pSL41uFmqc/JfRvW4D6FxN3uqX+nqBq0aXDopUn+GFt

h9PbOm+mKySbxTtpqPu0Z/3QRKTDHdhugmBk4J0RJHmnAQLdnvEgy4ZyRBLWt0yw

xoe2ywKBgHuDdsU4CW/FW3vNqMe1HLuIBeqHCWg5XSgl/MHLiflJ+tQxbKLTXtyU

z/sNijftjwWrs+NrzJvTWnlXSyLGFkKK/TotkHBDsEY5z+Rhu0Ndm20btkD4j7Hf

AeCSGnwPKKLLtNh6pVngBDiUbh+GkN+awwETQCDLgxqB94LJN02L

-----END RSA PRIVATE KEY-----

4) Remova o arquivo tsconfig.json original caso haja e crie o arquivo tsconfig.json em backend por linha de comando:

> tsc - -init

4.1) Habilite as seguintes entradas abaixo:

```
{
 "compilerOptions": {
  /* Basic Options */
 "compileOnSave": true, //compila o arquivo de ts para js ao salvar o arquivo ts. Algumas IDEs não possuem suporte a compileOnSave.
  "target": "es5",                    /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' or 'ESNEXT'. */
  "module": "commonjs",               /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
  // "lib": [],                       /* Specify library files to be included in the compilation. */
  "allowJs": true,                    /* Allow javascript files to be compiled. */
  // "checkJs": true,                 /* Report errors in .js files. */
  // "jsx": "preserve",               /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
  // "declaration": true,             /* Generates corresponding '.d.ts' file. */
  // "declarationMap": true,          /* Generates a sourcemap for each corresponding '.d.ts' file. */
  // "sourceMap": true,               /* Generates corresponding '.map' file. */
  // "outFile": "./",                 /* Concatenate and emit output to single file. */
  "outDir": "dist",                   /* Redirect output structure to the directory. */
 // "outDir": "./",     caso não seja configurada uma pasta para armazenar os arquivos *.js os mesmo serão salvos no mesmo nivel do arquivo *.ts comrrenspondente  */
  // "rootDir": "./",                 /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
  // "composite": true,               /* Enable project compilation */
```

```
    // "removeComments": true,                /* Do not emit comments to output. */
    // "noEmit": true,                        /* Do not emit outputs. */
    // "importHelpers": true,                 /* Import emit helpers from 'tslib'. */
    // "downlevelIteration": true,            /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or 'ES3'. */
    // "isolatedModules": true,               /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */

    /* Strict Type-Checking Options */
    "strict": true,                           /* Enable all strict type-checking options. */
    // "noImplicitAny": true,                 /* Raise error on expressions and declarations with an implied 'any' type. */
    // "strictNullChecks": true,              /* Enable strict null checks. */
    // "strictFunctionTypes": true,           /* Enable strict checking of function types. */
    // "strictBindCallApply": true,           /* Enable strict 'bind', 'call', and 'apply' methods on functions. */
    // "strictPropertyInitialization": true,  /* Enable strict checking of property initialization in classes. */
    // "noImplicitThis": true,                /* Raise error on 'this' expressions with an implied 'any' type. */
    // "alwaysStrict": true,                  /* Parse in strict mode and emit "use strict" for each source file. */

    /* Additional Checks */
    // "noUnusedLocals": true,                /* Report errors on unused locals. */
    // "noUnusedParameters": true,            /* Report errors on unused parameters. */
    // "noImplicitReturns": true,             /* Report error when not all code paths in function return a value. */
    // "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */

    /* Module Resolution Options */
    // "moduleResolution": "node",            /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
    // "baseUrl": "./",                       /* Base directory to resolve non-absolute module names. */
    // "paths": {},                           /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
    // "rootDirs": [],                        /* List of root folders whose combined content represents the structure of the project at runtime. */
    // "typeRoots": [],                       /* List of folders to include type definitions from. */
    // "types": [],                           /* Type declaration files to be included in compilation. */
    // "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, just typechecking. */
    "esModuleInterop": true                   /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for all imports. Implies 'allowSyntheticDefaultImports'. */
    // "preserveSymlinks": true,              /* Do not resolve the real path of symlinks. */

    /* Source Map Options */
    // "sourceRoot": "",                      /* Specify the location where debugger should locate TypeScript files instead of source locations. */
    // "mapRoot": "",                         /* Specify the location where debugger should locate map files instead of generated locations. */
    // "inlineSourceMap": true,               /* Emit a single file with source maps instead of having a separate file. */
    // "inlineSources": true,                 /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--sourceMap' to be set. */

    /* Experimental Options */
    "experimentalDecorators": true,           /* Enables experimental support for ES7 decorators. */
    "emitDecoratorMetadata": true,            /* Enables experimental support for emitting type metadata for decorators. */
  }
}
```

5) Crie o arquivo server.ts em backend com o código abaixo e salve:

```
import * as jsonServer from 'json-server' //npm install json-server @types/json-server -D
import {Express} from 'express' // linha original - apresenta erro quando instalo //npm install json-server @types/json-server -D
//import {Express} from 'express'  // bilbioteca utilizada para criarmos uma Aplicação em Express
```

```
import * as fs from 'fs'    //fs => File System => modulo que permite ler um arquivo do disco
import * as https from 'https' //importando o modulo Https

const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

// Use default router
server.use(router)

const options = {    //options é um objeto JSON que possui duas propriedades "cert" e "key"
   cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado
   key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave
}
let port = 3001; //porta onde esta aplicação esta respondendo ou seja o json-server.
https.createServer(options, server).listen(port, () => {  // https é um objeto que permite criar um servidor - perceba que o servidor esta sendo criado com o protocolo em
                                                          // https, porém não é válido pois não é reconhecido pelo browser. Embora não seja válido o Certificado fornecerá
                                                          // a chave publica para o browser fazer a criptografia dos dados que forem enviados.
   console.log('JSON Server is running on https://localhost:'+port)
})
```

5.1) Caso o arquivo server.js, criado a partir do server.ts, não seja exibido rode o comando a seguir na pasta onde esta o arquivo server.ts: >tsc -w

Obs.: No arquivo tsconfig.json segundo a configuração abaixo os arquivos *.ts deveriam ser compilados automaticamente, gerando os seus respectivos *.js quando o arquivo *.ts for salvo, mas algumas IDES não possuem suporte a "compileOnSave", veja o arquivo tsconfig.ts, por isso devemos rodar o comando "tsc -w" no diretório onde esta o arquivo *.ts para gerar o arquivo *.js.

tsconfig.json

```
{
  "compileOnSave": true, //compila o arquivo de ts para js ao salvar o arquivo ts
  "compilerOptions": {
    "outDir": "dist"
  }
}
```

6) Crie o arquivo db.json com o conteúdo abaixo na raiz do projetos

db.json

```
{
  "restaurants": [
    {
```

"id": "bread-bakery",
"name": "Bread & Bakery",
"category": "Bakery",
"deliveryEstimate": "25m",
"rating": 4.9,
"imagePath": "assets/img/restaurants/breadbakery.png",
"about": "A Bread & Brakery tem 40 anos de mercado. Fazemos os melhores doces e pães. Compre e confira.",
"hours": "Funciona de segunda à sexta, de 8h às 23h"
},
{
"id": "burger-house",
"name": "Burger House",
"category": "Hamburgers",
"deliveryEstimate": "100m",
"rating": 3.5,
"imagePath": "assets/img/restaurants/burgerhouse.png",
"about": "40 anos se especializando em trash food.",
"hours": "Funciona todos os dias, de 10h às 22h"
},
{
"id": "coffee-corner",
"name": "Coffee Corner",
"category": "Coffee Shop",
"deliveryEstimate": "30-40m",
"rating": 4.8,
"imagePath": "assets/img/restaurants/coffeecorner.png",
"about": "A Coffe Corner foi eleita a melhor cafeteria da cidade.",
"hours": "Funciona de segunda à sábado, de 6h às 22h"
},
{
"id": "green-food",
"name": "Green Food",
"category": "Saudável",
"deliveryEstimate": "75m",
"rating": 4.1,
"imagePath": "assets/img/restaurants/greenfood.png",
"about": "Comida saudável é no Green Food. Compramos barato, vendemos caro. ;)",
"hours": "Somente em horário de almoço, das 11h às 15h"
},
{
"id": "ice-cream",
"name": "Ice Cream",
"category": "Ice Creams",
"deliveryEstimate": "40-65m",
"rating": 4.5,
"imagePath": "assets/img/restaurants/icy.png",
"about": "A Ice Cream é uma sorveteria inovadora.",
"hours": "Funciona todos os dias, de 10h às 1h"

```
  },
  {
    "id": "tasty-treats",
    "name": "Tasty Treats",
    "category": "Doces",
    "deliveryEstimate": "20m",
    "rating": 4.4,
    "imagePath": "assets/img/restaurants/tasty.png",
    "about": "A doceria com mais tradição da cidade",
    "hours": "Funciona de segunda à sábado, de 8h às 23h"
  }
],
"menu": [
  {
    "id": "cup-cake",
    "imagePath": "assets/img/foods/cupcake.png",
    "name": "Cup Cake",
    "description": "Cup Cake recheado de Doce de Leite",
    "price": 8.7,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "donut",
    "imagePath": "assets/img/foods/donut.png",
    "name": "Donut",
    "description": "Coberto com chantilly",
    "price": 2.5,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "bread",
    "imagePath": "assets/img/foods/bread.png",
    "name": "Pão Artesanal Italiano",
    "description": "Pão artesanal com queijos italianos",
    "price": 15.9,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "burger",
    "imagePath": "assets/img/foods/burger.png",
    "name": "Classic Burger",
    "description": "O clássico. Não tem como errar.",
    "price": 18.5,
    "restaurantId": "burger-house"
  },
  {
    "id": "fries",
    "imagePath": "assets/img/foods/fries.png",
```

"name": "Batatas Fritas",
"description": "Batatas fritas crocantes",
"price": 5.5,
"restaurantId": "burger-house"
},
{
"id": "coke.png",
"imagePath": "assets/img/foods/coke.png",
"name": "Refrigerante",
"description": "O refri mais gelado da cidade.",
"price": 4.5,
"restaurantId": "burger-house"
},
{
"id": "capp",
"imagePath": "assets/img/foods/capp.png",
"name": "Cappuccino com Chantilly",
"description": "Tradicional cappuccino com chantilly",
"price": 9.9,
"restaurantId": "coffee-corner"
},
{
"id": "espreso",
"imagePath": "assets/img/foods/espreso.png",
"name": "Super Espreso",
"description": "Café espresso duplo. ",
"price": 12.5,
"restaurantId": "coffee-corner"
},
{
"id": "starb",
"imagePath": "assets/img/foods/starb.png",
"name": "Starbucks Copycat",
"description": "O mais pedido da casa. O verdadeiro café americano pura água.",
"price": 15.6,
"restaurantId": "coffee-corner"
},
{
"id": "zlasagna",
"imagePath": "assets/img/foods/zlasagna.png",
"name": "Lasagna de Zucchini",
"description": "Lasagna deliciosa que se salva por causa do queijo",
"price": 14.9,
"restaurantId": "green-food"
},
{
"id": "greensandwich",
"imagePath": "assets/img/foods/greensandwich.png",

"name": "Sanduíche de Frango",
"description": "Frango grelhado e salada",
"price": 10.5,
"restaurantId": "green-food"
},
{
"id": "ceasar",
"imagePath": "assets/img/foods/ceasar.png",
"name": "Salada Ceasar",
"description": "Clássica salada ceasar.",
"price": 33.9,
"restaurantId": "green-food"
},
{
"id": "sunday",
"imagePath": "assets/img/foods/sunday.png",
"name": "Mega Sunday",
"description": "Sunday com várias bolas de sorvete a sua escolha",
"price": 21.5,
"restaurantId": "ice-cream"
},
{
"id": "chocpop",
"imagePath": "assets/img/foods/chocpop.png",
"name": "Picolé de Brigadeiro",
"description": "Picolé de brigadeiro recheado com côco.",
"price": 8.5,
"restaurantId": "ice-cream"
},
{
"id": "icestd.png",
"imagePath": "assets/img/foods/icestd.png",
"name": "Sorvete Simples",
"description": "Clássico sorvete de rua onde todos os sabores são idênticos.",
"price": 1.5,
"restaurantId": "ice-cream"
},
{
"id": "cup",
"imagePath": "assets/img/foods/cup.png",
"name": "Cup Cake de Choc. Branco",
"description": "Cup Cake de chocolate branco",
"price": 5.5,
"restaurantId": "tasty-treats"
},
{
"id": "straw",
"imagePath": "assets/img/foods/straw.png",

"name": "Bolo de Morango",
"description": "Bolo recheado e coberto com morangos",
"price": 40.5,
"restaurantId": "tasty-treats"
},
{
"id": "cake",
"imagePath": "assets/img/foods/cake.png",
"name": "Fatia de Bolo",
"description": "Fatia de bolo de morango",
"price": 5.9,
"restaurantId": "tasty-treats"
}
],
"reviews": [
{
"name": "Julia",
"date": "2017-01-23T18:25:43",
"rating": 4.5,
"comments": "Tudo muito bom, entrega no tempo certo",
"restaurantId": "bread-bakery"
},
{
"name": "Aline",
"date": "2016-12-12T18:25:43",
"rating": 3,
"comments": "NÃO recomendo. O recheio é só um creme, não tem camarão e nem carne",
"restaurantId": "bread-bakery"
},
{
"name": "Renata",
"date": "2016-12-12T18:25:43",
"rating": 2,
"comments": "Não foi muito agradável ao nosso paladar, e eu e meu esposo no outro dia passamos super mal. Pode até ter sido outra coisa, mas...",
"restaurantId": "bread-bakery"
},
{
"name": "Renata",
"date": "2016-12-12T18:25:43",
"rating": 5,
"comments": "Nunca deixou na mão, comida sempre deliciosa.",
"restaurantId": "bread-bakery"
},
{
"name": "Daniela",
"date": "2017-01-23T18:25:43",
"rating": 4,
"comments": "Sanduíche muito bom! Já pedi diversas vezes sempre bom!",

"restaurantId": "burger-house"
},
{
"name": "Laila",
"date": "2017-02-02T18:25:43",
"rating": 5,
"comments": "Melhor hambúrguer que já provei. Vai ser difícil voltar para mc e bk",
"restaurantId": "burger-house"
},
{
"name": "Suzana",
"date": "2016-12-12T18:25:43",
"rating": 3.5,
"comments": "Meu sanduíche chegou frio e todo desmontado =\\",
"restaurantId": "burger-house"
},
{
"name": "Eric",
"date": "2017-01-23T18:25:43",
"rating": 4.5,
"comments": "Excelente opção! Recomendo!",
"restaurantId": "coffee-corner"
},
{
"name": "Amanda",
"date": "2017-02-02T18:25:43",
"rating": 2.5,
"comments": "Demorou muito e o café veio gelado",
"restaurantId": "coffee-corner"
},
{
"name": "Carla",
"date": "2016-12-12T18:25:43",
"rating": 4.8,
"comments": "Tudo perfeito. Entrega, embalagem, comida. Resultado? Melhor custo benefício",
"restaurantId": "coffee-corner"
},
{
"name": "Rodrigo",
"date": "2016-12-12T18:25:43",
"rating": 4,
"comments": "Muito rápida a entrega, mas é caro.",
"restaurantId": "coffee-corner"
},
{
"name": "André",
"date": "2016-12-12T18:25:43",
"rating": 4.5,

"comments": "Excelente como sempre!",

"restaurantId": "coffee-corner"

},

{

"name": "João",

"date": "2017-01-23T18:25:43",

"rating": 5,

"comments": "Excelente. Fiz o pedido com outro endereço e quando vi o erro liguei avisando. O motoqueiro cortou a cidade todinha e a comida ainda chegou quentinha (e deliciosa). Estou sempre pedindo e recomendando",

"restaurantId": "green-food"

},

{

"name": "Marcelo",

"date": "2017-02-02T18:25:43",

"rating": 4.5,

"comments": "Chegou muito rápido, antes mesmo da previsão",

"restaurantId": "green-food"

},

{

"name": "Fatinha",

"date": "2017-01-23T18:25:43",

"rating": 2,

"comments": "Ñ gostei dos sabores q pedi e achei o preço muito elevado",

"restaurantId": "ice-cream"

},

{

"name": "Lucas",

"date": "2017-02-02T18:25:43",

"rating": 2.5,

"comments": "Esperar 1:45hr pra receber o pedido é INACEITÁVEL",

"restaurantId": "ice-cream"

},

{

"name": "Jacira",

"date": "2017-02-01T18:25:43",

"rating": 4.5,

"comments": "Veio OK!",

"restaurantId": "ice-cream"

},

{

"name": "Ana",

"date": "2017-01-23T18:25:43",

"rating": 5,

"comments": "Rápido e sempre uma delicia!! Ja pedi 3x e sempre veio rapido e tudo certinho.",

"restaurantId": "tasty-treats"

},

{

"name": "Larissa",

"date": "2017-02-02T18:25:43",

"rating": 2.5,
"comments": "A taxa de entrega poderia ser mais barata.",
"restaurantId": "tasty-treats"
},
{
"name": "Jarbas",
"date": "2016-12-15T18:25:43",
"rating": 4.7,
"comments": "Excelente e chegou muito rápido!",
"restaurantId": "tasty-treats"
},
{
"name": "Rafaela",
"date": "2017-01-12T18:25:43",
"rating": 5,
"comments": "Chegou rápido e perfeito!",
"restaurantId": "tasty-treats"
},
{
"name": "Julia",
"date": "2016-12-12T18:25:43",
"rating": 1.5,
"comments": "Sempre gostei muito dos pedidos que eu fiz, mas dessa vez foi péssimo e decepcionou muito. Demorou 30min a mais pra chegar do que o tempo máximo esperado.",
"restaurantId": "tasty-treats"
},
{
"name": "Marília",
"date": "2016-12-20T18:25:43",
"rating": 4.5,
"comments": "Taxa de entrega bastante elevada e precisa colocar no app se a pessoa quer porção adicional de algo (calda, crocante etc)",
"restaurantId": "tasty-treats"
}
],
"orders": []
}

7) Crie o arquivo server.ts em backend e gere o arquivo server.js

```
import * as jsonServer from 'json-server' //npm install json-server @types/json-server -D
//import {Express} from 'express' //no projeto original apresenta erro quando instalo //npm install json-server @types/json-server -D

import * as fs from 'fs'   //fs => File System
import * as https from 'https'

const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')  //db.json => arquivo que simula nosso servidor de banco de dados
const middlewares = jsonServer.defaults()
```

```
// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

// Use default router
server.use(router)

const options = {  //options => objeto JSON com duas propriedades
   cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado
   key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave
}
let port = 3010;
https.createServer(options, server).listen(port, () => {    //criando o servidor
   console.log('JSON Server is running on https://localhost:'+port)
})
```

8) Rode o comando a seguir no terminal para disparar o json-server:

> node backend/dist/server //não use a extensão *.js

Obs.: O comando acima deve ser rodado fora da pasta “backend”

9) Entre com o endereço a seguir no browser:**https://localhost:3001**

9.1) No caso de alterarmos a porta no arquivo server.ts, o servidor ainda responderá às chamadas na porta 3001, pois não existe nenhum processo monitorando a alteração do arquivo server.ts e reiniciando o servidor para obter as novas configurações. Com a finalidade de resolvermos este impasse utilizaremos um utilitário chamado **nodemon** que irá monitorar as mudanças no(s) arquivo(s) além de reiniciar o servidor json-server. Um outro ponto que vale ressaltar é que esta aplicação é considerada uma Aplicação Node.

10) Instalando o serviço nodemon.

>npm install nodemon -g //-g => instalando de forma global

11) Inicializando o nodemon para monitorar determinados arquivos e reiniciar o servidor json-server.

>nodemon - - watch [pasta_monitorada/] caminho_da_pasta_monitorada_ate_ao_arquivo_monitorado/arquivo_monitorado.js

>nodemon - - watch backend backend/dist/server.js //neste caso não remova a extensão *.js

12) Altere a porta no aquivo server.js para 3020 e faça uma chamada ao servidor com o endereço a seguir: https://localhost:3020

Obs.: Repare no terminal que o **nodemon** detectou as mudanças no aquivo server.ts e reiniciou automaticamente o servidor json-server.

Obs.: A partir deste momento qualquer alteração no arquivo server.ts fará com que o nodemon reinicie o servidor json-server, obtendo assim as novas alterações.

Seção 14

Vídeo 111

Implementando a Rota de Login

Nesta seção utilizaremos o **nodemon** que é um utilitário que irá monitorar aquivos específicos e em caso de mudanças destes o servidor será reiniciado.

1) Crie o projeto: >ng new my-app-cli - -spec=false

2) Crie uma pasta de nome backend na raiz do projeto, será nesta pasta que criaremos nossos arquivos server.ts e tsconfig.json

3) Crie uma pasta de nome keys em backend.

3.1) Na pasta backend/keys/, crie os arquivos cert.pem e key.pem com os respectivos conteúdos abaixo:

backend/keys/cert.pem

-----BEGIN CERTIFICATE-----

MIIDtTCCAp2gAwIBAgIJAKDr6YbZ+6KXMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV

BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX

aWRnaXRzIFB0eSBMdGQwHhcNMTcwODA4MDI0NzA3WhcNMTcwOTA3MDI0NzA3WjBF

MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50

ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB

CgKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl28gI9ABf

dN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+FGA9vXy4L

2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS0CRRYZGX

w08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrgsGgiH0Hn

jOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIwd3KhL6e5

aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABo4GnMIGkMB0GA1UdDgQWBBQebMNu

xfSChegKLYv2Dp7aVPYIRjB1BgNVHSMEbjBsgBQebMNuxfSChegKLYv2Dp7aVPYI

RqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNV

BAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAKDr6YbZ+6KXMAwGA1UdEwQF

MAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAIt5OHCBsYuIvB1UsI9rFkRunQ9eCORD

gk/dg6mtQaCUKqJRoQq+Zqi+y/z+Jxl1pgzudaOPwX/wLDM31PGWZbwzABR1uIR8

aQ/sDLw0yUdGvtUnrZOK4CVNQgl+zXNTmpmEx1JAT7xvXb5OPahdi/AYqVW6SWET

QCiRCNsqv3xqyStAURi8B0BVVJUTNbn9+CkBU7FDZpBjWCKUsMfm/4dm/u3eTxIS

oOf70F25k5Ws5I11usvlQY6/I17aGXU29voKws0v7iClFwqtOt8nntGi5I6OQrd5

2PvOp/kA2TPxi83nP0/WT0pzvM/ykzbxh4oKmCIzp6wXx8JaKIr4Xr0=

-----END CERTIFICATE-----

backend/keys/key.pem

-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl

28gI9ABfdN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+F

GA9vXy4L2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS
0CRRYZGXw08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrg
sGgiH0HnjOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIw
d3KhL6e5aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABAoIBABvTp31tkzLLy4qo
yYbWsJ4w7vw0womd0U5RRfwKa6lalDHvVFaBrsUbSIDhagIpdNQdDBWrNuEO/Cwt
x12AIIP1meN/gXvPdKds3cYkL2g/HGE25UMI8FghuI6evnmb+aLHcsHbxsf45ze+
lERyzF3AUxQ/UaNmgIqEPIw34GYqiQB4ih6H1tQrKWVuvMtNc7o1kYOuTjR581h9
gXDKxGPkFyn2Y6xw1FDevAjzLvb4fJ/PLcTtEzbpX0+Jwpw/sPO5Vu9aj5Y/WpcF
k94ypnd8hR8J7kvjUmoPJwxHZ8wU75OCq+9KvIq8AU8LARVv9cK26drmrGEH5YEu
Dgy7uAECgYEA+l9yJZj1tzqYUoy8P70+jz5Ixrk771JForKBYv/rQL8gtpCOdVOy
2SrjgbvRKsf6mklTmJy3ljLNVuoFxLaiBauOH7zlC+twucdZhWOdKfj92WFr78d6
Txp1E2jUYijw+SVnmOm71IFqF9A1+vGnX+MV9awCEbzck5Km1IodsXkCgYEAx47f
atR/jRxmDKYhvU+i27fP8BxessBIW+jgP4qE+YaFNdUG0p2zjnco1G66DzAntiFB
pjYFzZFe6kFUIsHNeag3i0HQqqfTsYeOuFo5wA2yHxqhGj//aBlvjl2nScm4nNxL
e+IiT5lnYixsMeXp0wsXDjVRL+qI0ddCxMSamKMCgYEAuyOGNTX8SyUf32vT9Sxl
Szn5uZeQhdQPuokEDtRMqctyDyNDZ8zUvF/DRx/0Oq7Y0GOIeWSCEGEOfC1VfHAp
8GUHSQ+uf9dOHX40mnAXc9zpGIcp30JRWDjlU0minXAOlZrwsSnbO/4zJ3N4NYZU
XWZl7ofP0CRDFoah7fECOrkCgYAWeZPV/0rCRMGFiSxhmtd3sXL80uUjcp5zo32m
Cl7Bj5HvJyfWjqZLVEZH7pSL41uFmqc/JfRvW4D6FxN3uqX+nqBq0aXDopUn+GFt
h9PbOm+mKySbxTtpqPu0Z/3QRKTDHdhugmBk4J0RJHmnAQLdnvEgy4ZyRBLWt0yw
xoe2ywKBgHuDdsU4CW/FW3vNqMe1HLuIBeqHCWg5XSgl/MHLiflJ+tQxbKLTXtyU
z/sNijftjwWrs+NrzJvTWnlXSyLGFkKK/TotkHBDsEY5z+Rhu0Ndm20btkD4j7Hf
AeCSGnwPKKLLtNh6pVngBDiUbh+GkN+awwETQCDLgxqB94LJN02L
-----END RSA PRIVATE KEY-----

4) Remova o arquivo tsconfig.json original caso haja e crie o arquivo tsconfig.json em backend por linha de comando:

> tsc - -init

4.1) Habilite as seguintes entradas abaixo:

```
{
 "compilerOptions": {
  /* Basic Options */
 "compileOnSave": true, //compila o arquivo de ts para js ao salvar o arquivo ts. Algumas IDEs não possuem suporte a compileOnSave.
  "target": "es5",                    /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' or 'ESNEXT'. */
  "module": "commonjs",               /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
  // "lib": [],                       /* Specify library files to be included in the compilation. */
  "allowJs": true,                    /* Allow javascript files to be compiled. */
  // "checkJs": true,                 /* Report errors in .js files. */
  // "jsx": "preserve",               /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
  // "declaration": true,             /* Generates corresponding '.d.ts' file. */
  // "declarationMap": true,          /* Generates a sourcemap for each corresponding '.d.ts' file. */
  // "sourceMap": true,               /* Generates corresponding '.map' file. */
  // "outFile": "./",                 /* Concatenate and emit output to single file. */
  "outDir": "dist",                   /* Redirect output structure to the directory. */
 // "outDir": "./",     caso não seja configurada uma pasta para armazenar os arquivos *.js os mesmo serão salvos no mesmo nivel do arquivo *.ts comrrenspondente  */
  // "rootDir": "./",                 /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
  // "composite": true,               /* Enable project compilation */
  // "removeComments": true,          /* Do not emit comments to output. */
  // "noEmit": true,                  /* Do not emit outputs. */
  // "importHelpers": true,           /* Import emit helpers from 'tslib'. */
  // "downlevelIteration": true,      /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or 'ES3'. */
  // "isolatedModules": true,         /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */
```

```
  /* Strict Type-Checking Options */
  "strict": true,                       /* Enable all strict type-checking options. */
  // "noImplicitAny": true,             /* Raise error on expressions and declarations with an implied 'any' type. */
  // "strictNullChecks": true,          /* Enable strict null checks. */
  // "strictFunctionTypes": true,       /* Enable strict checking of function types. */
  // "strictBindCallApply": true,       /* Enable strict 'bind', 'call', and 'apply' methods on functions. */
  // "strictPropertyInitialization": true,  /* Enable strict checking of property initialization in classes. */
  // "noImplicitThis": true,            /* Raise error on 'this' expressions with an implied 'any' type. */
  // "alwaysStrict": true,              /* Parse in strict mode and emit "use strict" for each source file. */

  /* Additional Checks */
  // "noUnusedLocals": true,            /* Report errors on unused locals. */
  // "noUnusedParameters": true,        /* Report errors on unused parameters. */
  // "noImplicitReturns": true,         /* Report error when not all code paths in function return a value. */
  // "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */

  /* Module Resolution Options */
  // "moduleResolution": "node",        /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
  // "baseUrl": "./",                   /* Base directory to resolve non-absolute module names. */
  // "paths": {},                       /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
  // "rootDirs": [],                    /* List of root folders whose combined content represents the structure of the project at runtime. */
  // "typeRoots": [],                   /* List of folders to include type definitions from. */
  // "types": [],                       /* Type declaration files to be included in compilation. */
  // "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, just typechecking. */
  "esModuleInterop": true      /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for all imports. Implies 'allowSyntheticDefaultImports'. */
  // "preserveSymlinks": true,          /* Do not resolve the real path of symlinks. */

  /* Source Map Options */
  // "sourceRoot": "",                  /* Specify the location where debugger should locate TypeScript files instead of source locations. */
  // "mapRoot": "",                     /* Specify the location where debugger should locate map files instead of generated locations. */
  // "inlineSourceMap": true,           /* Emit a single file with source maps instead of having a separate file. */
  // "inlineSources": true,             /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--sourceMap' to be set. */

  /* Experimental Options */
  "experimentalDecorators": true,      /* Enables experimental support for ES7 decorators. */
  "emitDecoratorMetadata": true,       /* Enables experimental support for emitting type metadata for decorators. */
 }
}
```

5) Crie o arquivo server.ts em backend com o código abaixo e salve:

```
import * as jsonServer from 'json-server' //npm install json-server @types/json-server -D
import {Express} from 'express' // linha original - apresenta erro quando instalo //npm install json-server @types/json-server -D
//import {Express} from 'express'  // bilbioteca utilizada para criarmos uma Aplicação em Express
```

```
import * as fs from 'fs'    //fs => File System => modulo que permite ler um arquivo do disco
import * as https from 'https' //importando o modulo Httpsimport { handleAuthentication } from './auth';

const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

//1)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//      response.json({message: 'ok'}) //enviando a mensagem ok para o postman
//})

//2)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//    response.send({message: 'Alright'}) //enviando a mensagem ok para o postman
//})

//3)
//middleware que pode ser associado a uma rota neste caso a url: /login - esta eh uma rota personalizada para
//fazer o login do usuario que sera constituido de email e senha atraves de um objeto JSON criado no postman
server.post('/login', handleAuthentication) //devemos sempre combinar um "rota" ==> "/login" com um "middleware" ==> "handleAuthentication"

// Use default router
server.use(router)

const options = {
    cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado
    key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave
}

let port = 3023;

https.createServer(options, server).listen(port, () => {    //criando o servidor
    console.log('JSON Server is running on https://localhost:'+port)
})

//Enviar atraves do Postman
//{
//                "email" : "juliana@gmail.com",
//                "password" : "juliana23"
```

//}

5.1) Caso o arquivo server.js, criado a partir do server.ts, não seja exibido rode o comando a seguir na pasta onde esta o arquivo server.ts: >tsc -w

Obs.: No arquivo tsconfig.json segundo a configuração abaixo os arquivos *.ts deveriam ser compilados automaticamente, gerando os seus respectivos *.js quando o arquivo *.ts for salvo, mas algumas IDES não possuem suporte a "compileOnSave", veja o arquivo tsconfig.ts, por isso devemos rodar o comando "tsc -w" no diretório onde esta o arquivo *.ts para gerar o arquivo *.js.

tsconfig.json

```
{
  "compileOnSave": true, //compila o arquivo de ts para js ao salvar o arquivo ts
  "compilerOptions": {
    "outDir": "dist"
  }
}
```

6) Crie o arquivo auth.ts com o conteúdo abaixo na pasta bakend.

auth.ts

```
import {Request, Response} from 'express'
import { User, users } from './users';

//codigo que ira processar o post que fora feito para: https://localhost:3023/login
//handleAuthentication => eh uma constante que recebe o resultado de uma funcao
export const handleAuthentication = (req : Request, resp : Response) => {

   const user : User = req.body;
   //const user: User = req.body //a partir do postman enviaremos um objeto do tipo User com propriedades localizadas
                                 //no corpo do objeto Request: email e password

   if(isValid(user)){//valida se o usuario eh valido ou nao
      const dbUser: User = users[user.email];//users["juliana@gmail.com"] retornara a linha completa

      resp.json( { name: dbUser.name, email: dbUser.email } )//retornando os dados do usuario no corpo do objeto Response
   }else{
      resp.status(403).json({message: 'Dados invalidos!'}) //um requisicao pelo postman sem nada no body teria este resultado
   }

}

function isValid(user: User) : boolean {

   if(!user){//se o usuario NAO existir
     return false;
   }
   const dbUser : User= users[user.email] //mesmo que haja erro de compilacao rode a aplicacao
   return dbUser !== undefined && dbUser.matches(user);
}
```

//Erros:

//400 = erro de má formacao no request.

//401 = nao autorizado - recurso protegido / segundo a especificacao http devemos enviar um novo header informando

//qual metodo de autenticacao foi utilizado.

//403 = informacao invalida.

7) Crie o arquivo users.ts com o conteúdo na pasta backend.

users.ts

```
export class User {

    constructor(public email: string, public name: string, public password: string ){
    }

    //perceba que o metodo esta dentro da classe User
    matches(another: User): boolean { //compara o User que vem do body com um elemento da const - cada linha eh um elemento
        return another !== undefined && another.email === this.email && another.password == this.password;
    }

}



export const users = { //objeto JSON com tres propriedades sendo utilizado para simular nosso banco de dados
    "juliana@gmail.com" : new User('juliana@gmail.com', 'juliana', 'juliana23'),
    "fabio@gmail.com": new User('fabio@gmail.com',   'fabio',   'fabio23'),
    "amanda@gmail.com" : new User('amanda@gmail.com',  'amanda',  'amanda23')
}

//O acesso a cada elemento da constante "users" deve ser feito com base na variavel de declaracao do objeto User: users["juliana@gmail.com"]

//////////// No Postman faca:
//{
//          "email" : "juliana@gmail.com",
//          "password" : "juliana23"
//}
```

8) Crie o arquivo db.json com o conteúdo abaixo na raiz do projetos.

db.json

{
 "restaurants": [
  {
   "id": "bread-bakery",
   "name": "Bread & Bakery",
   "category": "Bakery",
   "deliveryEstimate": "25m",
   "rating": 4.9,
   "imagePath": "assets/img/restaurants/breadbakery.png",
   "about": "A Bread & Brakery tem 40 anos de mercado. Fazemos os melhores doces e pães. Compre e confira.",
   "hours": "Funciona de segunda à sexta, de 8h às 23h"
  },
  {
   "id": "burger-house",
   "name": "Burger House",
   "category": "Hamburgers",
   "deliveryEstimate": "100m",
   "rating": 3.5,
   "imagePath": "assets/img/restaurants/burgerhouse.png",
   "about": "40 anos se especializando em trash food.",
   "hours": "Funciona todos os dias, de 10h às 22h"
  },
  {
   "id": "coffee-corner",
   "name": "Coffee Corner",
   "category": "Coffee Shop",
   "deliveryEstimate": "30-40m",
   "rating": 4.8,
   "imagePath": "assets/img/restaurants/coffeecorner.png",
   "about": "A Coffe Corner foi eleita a melhor cafeteria da cidade.",
   "hours": "Funciona de segunda à sábado, de 6h às 22h"
  },
  {
   "id": "green-food",
   "name": "Green Food",
   "category": "Saudável",
   "deliveryEstimate": "75m",
   "rating": 4.1,
   "imagePath": "assets/img/restaurants/greenfood.png",
   "about": "Comida saudável é no Green Food. Compramos barato, vendemos caro. ;)",
   "hours": "Somente em horário de almoço, das 11h às 15h"
  },
  {
   "id": "ice-cream",
   "name": "Ice Cream",
   "category": "Ice Creams",
   "deliveryEstimate": "40-65m",

"rating": 4.5,
"imagePath": "assets/img/restaurants/icy.png",
"about": "A Ice Cream é uma sorveteria inovadora.",
"hours": "Funciona todos os dias, de 10h às 1h"
},
{
"id": "tasty-treats",
"name": "Tasty Treats",
"category": "Doces",
"deliveryEstimate": "20m",
"rating": 4.4,
"imagePath": "assets/img/restaurants/tasty.png",
"about": "A doceria com mais tradição da cidade",
"hours": "Funciona de segunda à sábado, de 8h às 23h"
}
],
"menu": [
{
"id": "cup-cake",
"imagePath": "assets/img/foods/cupcake.png",
"name": "Cup Cake",
"description": "Cup Cake recheado de Doce de Leite",
"price": 8.7,
"restaurantId": "bread-bakery"
},
{
"id": "donut",
"imagePath": "assets/img/foods/donut.png",
"name": "Donut",
"description": "Coberto com chantilly",
"price": 2.5,
"restaurantId": "bread-bakery"
},
{
"id": "bread",
"imagePath": "assets/img/foods/bread.png",
"name": "Pão Artesanal Italiano",
"description": "Pão artesanal com queijos italianos",
"price": 15.9,
"restaurantId": "bread-bakery"
},
{
"id": "burger",
"imagePath": "assets/img/foods/burger.png",
"name": "Classic Burger",
"description": "O clássico. Não tem como errar.",
"price": 18.5,
"restaurantId": "burger-house"

},
{
"id": "fries",
"imagePath": "assets/img/foods/fries.png",
"name": "Batatas Fritas",
"description": "Batatas fritas crocantes",
"price": 5.5,
"restaurantId": "burger-house"
},
{
"id": "coke.png",
"imagePath": "assets/img/foods/coke.png",
"name": "Refrigerante",
"description": "O refri mais gelado da cidade.",
"price": 4.5,
"restaurantId": "burger-house"
},
{
"id": "capp",
"imagePath": "assets/img/foods/capp.png",
"name": "Cappuccino com Chantilly",
"description": "Tradicional cappuccino com chantilly",
"price": 9.9,
"restaurantId": "coffee-corner"
},
{
"id": "espreso",
"imagePath": "assets/img/foods/espreso.png",
"name": "Super Espreso",
"description": "Café espresso duplo. ",
"price": 12.5,
"restaurantId": "coffee-corner"
},
{
"id": "starb",
"imagePath": "assets/img/foods/starb.png",
"name": "Starbucks Copycat",
"description": "O mais pedido da casa. O verdadeiro café americano pura água.",
"price": 15.6,
"restaurantId": "coffee-corner"
},
{
"id": "zlasagna",
"imagePath": "assets/img/foods/zlasagna.png",
"name": "Lasagna de Zucchini",
"description": "Lasagna deliciosa que se salva por causa do queijo",
"price": 14.9,
"restaurantId": "green-food"

},
{
"id": "greensandwich",
"imagePath": "assets/img/foods/greensandwich.png",
"name": "Sanduíche de Frango",
"description": "Frango grelhado e salada",
"price": 10.5,
"restaurantId": "green-food"
},
{
"id": "ceasar",
"imagePath": "assets/img/foods/ceasar.png",
"name": "Salada Ceasar",
"description": "Clássica salada ceasar.",
"price": 33.9,
"restaurantId": "green-food"
},
{
"id": "sunday",
"imagePath": "assets/img/foods/sunday.png",
"name": "Mega Sunday",
"description": "Sunday com várias bolas de sorvete a sua escolha",
"price": 21.5,
"restaurantId": "ice-cream"
},
{
"id": "chocpop",
"imagePath": "assets/img/foods/chocpop.png",
"name": "Picolé de Brigadeiro",
"description": "Picolé de brigadeiro recheado com côco.",
"price": 8.5,
"restaurantId": "ice-cream"
},
{
"id": "icestd.png",
"imagePath": "assets/img/foods/icestd.png",
"name": "Sorvete Simples",
"description": "Clássico sorvete de rua onde todos os sabores são idênticos.",
"price": 1.5,
"restaurantId": "ice-cream"
},
{
"id": "cup",
"imagePath": "assets/img/foods/cup.png",
"name": "Cup Cake de Choc. Branco",
"description": "Cup Cake de chocolate branco",
"price": 5.5,
"restaurantId": "tasty-treats"

```
  },
  {
    "id": "straw",
    "imagePath": "assets/img/foods/straw.png",
    "name": "Bolo de Morango",
    "description": "Bolo recheado e coberto com morangos",
    "price": 40.5,
    "restaurantId": "tasty-treats"
  },
  {
    "id": "cake",
    "imagePath": "assets/img/foods/cake.png",
    "name": "Fatia de Bolo",
    "description": "Fatia de bolo de morango",
    "price": 5.9,
    "restaurantId": "tasty-treats"
  }
],
"reviews": [
  {
    "name": "Julia",
    "date": "2017-01-23T18:25:43",
    "rating": 4.5,
    "comments": "Tudo muito bom, entrega no tempo certo",
    "restaurantId": "bread-bakery"
  },
  {
    "name": "Aline",
    "date": "2016-12-12T18:25:43",
    "rating": 3,
    "comments": "NÃO recomendo. O recheio é só um creme, não tem camarão e nem carne",
    "restaurantId": "bread-bakery"
  },
  {
    "name": "Renata",
    "date": "2016-12-12T18:25:43",
    "rating": 2,
    "comments": "Não foi muito agradável ao nosso paladar, e eu e meu esposo no outro dia passamos super mal. Pode até ter sido outra coisa, mas...",
    "restaurantId": "bread-bakery"
  },
  {
    "name": "Renata",
    "date": "2016-12-12T18:25:43",
    "rating": 5,
    "comments": "Nunca deixou na mão, comida sempre deliciosa.",
    "restaurantId": "bread-bakery"
  },
  {
```

"name": "Daniela",
"date": "2017-01-23T18:25:43",
"rating": 4,
"comments": "Sanduíche muito bom! Já pedi diversas vezes sempre bom!",
"restaurantId": "burger-house"
},
{
"name": "Laila",
"date": "2017-02-02T18:25:43",
"rating": 5,
"comments": "Melhor hambúrguer que já provei. Vai ser difícil voltar para mc e bk",
"restaurantId": "burger-house"
},
{
"name": "Suzana",
"date": "2016-12-12T18:25:43",
"rating": 3.5,
"comments": "Meu sanduíche chegou frio e todo desmontado =\\",
"restaurantId": "burger-house"
},
{
"name": "Eric",
"date": "2017-01-23T18:25:43",
"rating": 4.5,
"comments": "Excelente opção! Recomendo!",
"restaurantId": "coffee-corner"
},
{
"name": "Amanda",
"date": "2017-02-02T18:25:43",
"rating": 2.5,
"comments": "Demorou muito e o café veio gelado",
"restaurantId": "coffee-corner"
},
{
"name": "Carla",
"date": "2016-12-12T18:25:43",
"rating": 4.8,
"comments": "Tudo perfeito. Entrega, embalagem, comida. Resultado? Melhor custo beneficio",
"restaurantId": "coffee-corner"
},
{
"name": "Rodrigo",
"date": "2016-12-12T18:25:43",
"rating": 4,
"comments": "Muito rápida a entrega, mas é caro.",
"restaurantId": "coffee-corner"
},

{
"name": "André",
"date": "2016-12-12T18:25:43",
"rating": 4.5,
"comments": "Excelente como sempre!",
"restaurantId": "coffee-corner"
},
{
"name": "João",
"date": "2017-01-23T18:25:43",
"rating": 5,
"comments": "Excelente. Fiz o pedido com outro endereço e quando vi o erro liguei avisando. O motoqueiro cortou a cidade todinha e a comida ainda chegou quentinha (e deliciosa). Estou sempre pedindo e recomendando",
"restaurantId": "green-food"
},
{
"name": "Marcelo",
"date": "2017-02-02T18:25:43",
"rating": 4.5,
"comments": "Chegou muito rápido, antes mesmo da previsão",
"restaurantId": "green-food"
},
{
"name": "Fatinha",
"date": "2017-01-23T18:25:43",
"rating": 2,
"comments": "Ñ gostei dos sabores q pedi e achei o preço muito elevado",
"restaurantId": "ice-cream"
},
{
"name": "Lucas",
"date": "2017-02-02T18:25:43",
"rating": 2.5,
"comments": "Esperar 1:45hr pra receber o pedido é INACEITÁVEL",
"restaurantId": "ice-cream"
},
{
"name": "Jacira",
"date": "2017-02-01T18:25:43",
"rating": 4.5,
"comments": "Veio OK!",
"restaurantId": "ice-cream"
},
{
"name": "Ana",
"date": "2017-01-23T18:25:43",
"rating": 5,
"comments": "Rápido e sempre uma delicia!! Ja pedi 3x e sempre veio rapido e tudo certinho.",
"restaurantId": "tasty-treats"

```
    },
    {
      "name": "Larissa",
      "date": "2017-02-02T18:25:43",
      "rating": 2.5,
      "comments": "A taxa de entrega poderia ser mais barata.",
      "restaurantId": "tasty-treats"
    },
    {
      "name": "Jarbas",
      "date": "2016-12-15T18:25:43",
      "rating": 4.7,
      "comments": "Excelente e chegou muito rápido!",
      "restaurantId": "tasty-treats"
    },
    {
      "name": "Rafaela",
      "date": "2017-01-12T18:25:43",
      "rating": 5,
      "comments": "Chegou rápido e perfeito!",
      "restaurantId": "tasty-treats"
    },
    {
      "name": "Julia",
      "date": "2016-12-12T18:25:43",
      "rating": 1.5,
      "comments": "Sempre gostei muito dos pedidos que eu fiz, mas dessa vez foi péssimo e decepcionou muito. Demorou 30min a mais pra chegar do que o tempo máximo esperado.",
      "restaurantId": "tasty-treats"
    },
    {
      "name": "Marília",
      "date": "2016-12-20T18:25:43",
      "rating": 4.5,
      "comments": "Taxa de entrega bastante elevada e precisa colocar no app se a pessoa quer porção adicional de algo (calda, crocante etc)",
      "restaurantId": "tasty-treats"
    }
  ],
  "orders": []
}
```


9) Crie o arquivo server.ts em backend e gere o arquivo server.js

```
import * as jsonServer from 'json-server' //npm install json-server @types/json-server -D
//import {Express} from 'express' //no projeto original apresenta erro quando instalo //npm install json-server @types/json-server -D

import * as fs from 'fs'   //fs => File System
import * as https from 'https'

const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')  //db.json => arquivo que simula nosso servidor de banco de dados
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

// Use default router
server.use(router)

const options = {  //options => objeto JSON com duas propriedades
   cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado
   key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave
}
let port = 3001; //porta onde esta aplicação esta respondendo
https.createServer(options, server).listen(port, () => { //criando o servidor - perceba que o servidor esta sendo criado com o protocolo em https. //https.createServer
   console.log('JSON Server is running on https://localhost:'+port) //rodar com a url que esta sendo exibida no terminal
})
```

8) Rode o comando a seguir no terminal para disparar o json-server:

> node backend/dist/server //não use a extensão *.js

Obs.: O comando acima deve ser rodado fora da pasta “backend”

9) Entre com o endereço a seguir no browser:**https://localhost:3000**

9.1) No caso de alterarmos a porta no arquivo server.ts, o servidor ainda responderá às chamadas na porta 3001, pois não existe nenhum processo monitorando a alteração do arquivo server.ts e reiniciando o servidor para obter as novas configurações. Com a finalidade de resolvermos este impasse utilizaremos um utilitário chamado **nodemon** que irá monitorar as mudanças no(s) arquivo(s) além de reiniciar o servidor json-server. Um outro ponto que vale ressaltar é que esta aplicação é considerada uma Aplicação Node.

10) Instalando o serviço nodemon.

>npm install nodemon -g //-g => instalando de forma global

11) Inicializando o nodemon para monitorar determinados arquivos e reiniciar o servidor json-server.

>nodemon - - watch [pasta_monitorada/] caminho_da_pasta_monitorada_ate_ao_arquivo_monitorado/arquivo_monitorado.js

>nodemon - - watch backend backend/dist/server.js //neste caso não remova a extensão *.js

12) Altere a porta no aquivo server.js para 3020 e faça uma chamada ao servidor com o endereço a seguir: https://localhost:3020

Obs.: Repare no terminal que o **nodemon** detectou as mudanças no aquivo server.ts e reiniciou automaticamente o servidor json-server.

Obs.: A partir deste momento qualquer alteração no arquivo server.ts fará com que o nodemon reinicie o servidor json-server, obtendo assim as novas alterações.

EXTRA:

https://pt.stackoverflow.com/questions/64507/o-que-s%C3%A3o-middlewares-em-nodejs

Middleware é todo o tipo de função que está entre um pedido HTTP e a resposta final que o servidor envia de volta para o cliente.

Por exemplo, usando Express.js, um pedido GET simples teria esta código:

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

Se quiseres fazer um log do tipo de request e do url de um podes usar este middleware simples: request

```
app.use(function(req, res, next) {
    console.log(req.method, req.url);
    next();
});
```

A função middleware tem 3 parâmetros, pedido, resposta e callback. Podes ter ***n*** middleware a processar um pedido HTTP, encadeados. Quando um middleware acaba de processar coloca-se no final do código next(); invocando assim a callback e o código continua a correr para o proximo middleware ou resposta final.

O middleware é portanto uma funcionalidade, funções que executam processos intermédios. Os exemplos mais comuns são interagir com a BD, ir buscar ficheiros estáticos, tratar de erros ou redirecionamentos.

Um exepmplo de encadeamento de "middlewares" poderia ser assim:

```
var express = require('express'), // chamar a app e dar configurações minimas
app = express.createServer(),
port = 1337;

function middleHandler(req, res, next) {
// tratar erros
  var err = req.body.foo == null;
  if (!err) next();
  else res.send("a variavel foo não tem dados!");
}

app.use(function (req, res, next) {
// escrever na BD
 console.log(JSON.stringify(req.body));
 next();
});

app.use(function (req, res, next) {
 // outros processos
 next();
});

app.get('/', middleHandler, function (req, res) {
 console.log(ultimo passo);
 res.send("Obrigado pelo seu registo!");
});

app.listen(port); // iniciar o servidor
console.log('start server');
```

Repara que neste exemplo em cima o tem um middleware como argumento. Podes usar middleware encadeados também dessa maneira.
Aqui fica um exemplo real:

```
app.get('/', githubEvents, twitter, getLatestBlog, function(req, res){
   res.render('index', {
   title: 'MooTools',
   site: 'mootools',
   lastBlogPost: res.locals.lastBlogPost,
   tweetFeed: res.locals.twitter
   });
});
```

Sugestão de leitura:

Documentação Express.js sobre e tipos comums de middleware (em Inglês)

Quando você cria uma rota em sua aplicação web você libera uma área de sua aplicação para que usuários acessem via browser ou outras aplicações acessem através de algum framework http client. Quando essa rota é acessada, dois objetos principais surgem no callback dessa função, eles são:

request: ele é responsável por carregar dados da requisição que esta sendo realizada, geralmente vem com dados do cliente e algums parâmetros de input, como querystrings, parâmetros de rotas e body de um formulário. Em resumo, este objeto contém diversos dados do cliente.

response: este objeto permite que o servidor envie uma resposta para o cliente que realizou uma requisição. Aqui você pode enviar um html, json, dados via header, redirecionar a resposta para uma outra requisição, em geral este é um objeto focado em dar uma resposta para o cliente.

Os **middlewares** são funções que podem tratar os inputs e outputs das rotas antes e ou depois que uma rota é processada, ou seja, você pode criar um middleware que intercepta e verifica se uma requisição esta enviando um header específico e que caso o mesmo não esteja enviando o header ela retorne uma tela de erro para o usuário, negando a requisição de acessar uma determinada rota da aplicação, neste caso você criou e injetou um middleware que trata uma pré-requisição. Você também pode criar um middleware que no final de cada resposta de uma rota, retorne também um header com metadados da resposta, por exemplo, headers de paginação de dados. Neste caso estamos criando um middleware de pós-requisição. Não há limites ao injetar middlewares, você pode criar e configurar N middlewares em sua aplicação, porém é sempre bom entender o que são cada middlewares e principalmente a ordem que cada middleware é injetado afeta no processamento de uma rota, ou seja, caso você injete middlewares em uma ordem errada, como efeito colateral sua aplicação pode responder ou até mesmo não processar corretamente suas rotas, por isso é extremamente importante entender o que cada middleware faz e em que ordem injetá-los.

Seção 14

Vídeo 112

<u>Introdução a JWT</u>

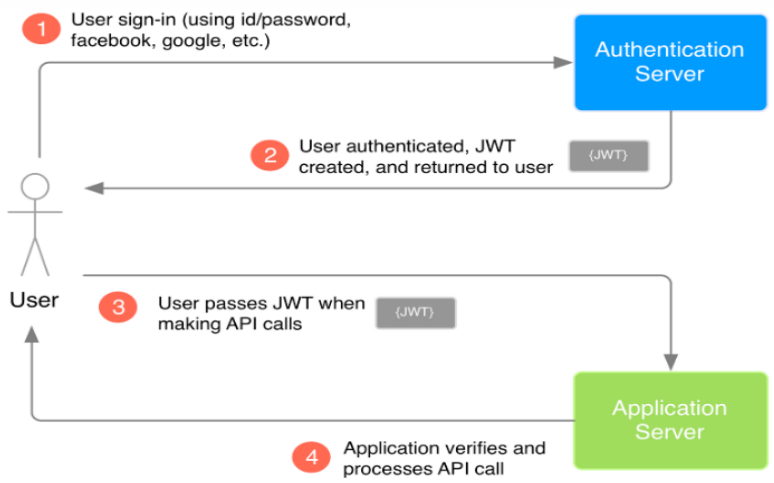
O que é um Token?

É um "dado" que serve para identificar o portador o qual pode ser um usuário ou até mesmo uma outra aplicação. Portanto, aquele que tem em sua posse um token terá o direito de acessar ou fazer algo em um sistema.

O que é um Token JWT?

Um Token JWT (JSON Web Token) é um padrão de formato de token utilizado para aplicações web que precisam identificar ou dar permissão a uma das partes.

IMAGEM EXTRA:



How an application uses JWT to verify the authenticity of a user.

Token JWT (JSON Web Token) **NÃO** tem a finalidade de resolver a segurança de uma aplicação.

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

EXTRA: O Token é enviado no hearder de um objeto request e response.

Token JWT resolvendo problemas…

Numa aplicação web tradicional a identificação do cliente, ou seja, do browser e feita através de cookies no qual em algum momento o browser enviará os dados do cliente para o back-end, servidor, e este gerará um id (set-cookie:sessionid=123) que é inserido no cookie que por sua vez identificará exclusivamente o cliente.

(browser) ========== user = John ==========> (servidor)

(browser) <========== set-cookie: sessionid = 123========== (servidor) //no header do response

(browser) ========== sessionid = 123==========> (servidor) //no header da request

(browser) <========== Hello, John!========== (servidor)

O cookie virá no ***header da resposta http***, sempre que o cliente realizar uma nova chamada http, o browser enviará automaticamente o cookie previamente criado contendo o sessionid do cliente para o servidor e desta forma o servidor será capaz de identificar o cliente que esta associado ao id do cookie, assim funciona a especificação HTTP.

(browser) ========== sessionid = 123==========> (servidor)

(browser) <========== Hello, John!========== (servidor)

Os cookies não são exclusivos para identificação do cliente, mas são fundamentais para identificação do browser devido a natureza do ***Protocolo Http que é Stateless***.

Identificação em uma Aplicação Mobile

No caso de aplicações que executam nativamente em um ***Smartphone*** mesmo utilizando o Protocolo HTTP ***existe uma grande chance de não se utilizar cookies para identificar a aplicação e o usuário***, isso por que esta funcionalidade já é implementada nativamente nos navegadores, mas não por outras plataformas então se escolhe um padrão diferente de identificação, o que leva uma aplicação a lidar com diferentes formatos de identificação para atender a diferentes tipos de clientes então para resolver este problema utilizamos o Token JWT onde se estabelece um formato padrão para todos, evitando assim que a aplicação tenha que lidar com múltiplos formatos.

No exemplo abaixo ilustramos a identificação de uma Aplicação Mobile:

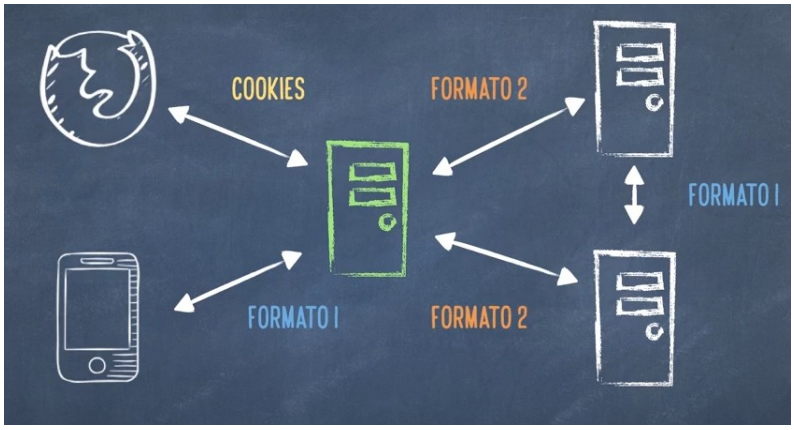
(IOS ou Android(nativo)) ========== user = John ==========> (servidor)

(IOS ou Android(nativo)) <========== {name: John, access:123}========== (servidor)

No exemplo abaixo ilustramos um formato genérico de token:

(IOS ou Android(nativo)) ========== access = 123 ==========> (servidor)

(IOS ou Android(nativo)) <========== {message: “Ok, John”}========== (servidor)



Diferentes aplicações acessando o back-end com formatos de identificação distintos para atender a diversos tipos de clientes.

Em um cenário com JWT o browser poderá enviar a identificação e receberá o token de volta que deve ser mandado em requisições seguintes trabalhando com diferentes aplicações.

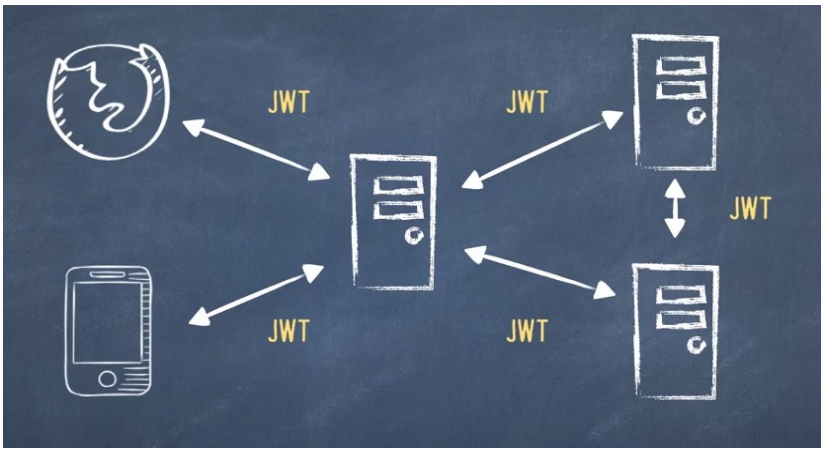
Token(s) Padronizado(s) com JWT:

(browser, aplicação ou mobile) ========== user = John ==========> (servidor)

(browser, aplicação ou mobile) <========== access-token: 123.456.789 ========== (servidor)

(browser, aplicação ou mobile) ========= autorization = bearer: 123.456.789 =======> (servidor)

(browser, aplicação ou mobile) <========== Hello, John!========== (servidor)



Servidor, back-end, atendendo a diferentes aplicações clientes com Token padronizado JWT.

Seção 14

Vídeo 113

Estrutura de um Token JWT:

Um token JWT é dividido em três partes: header, payload e signature.

*** Header:** escrito no formato JSON, logo pode ser considerado um objeto do tipo JSON, é onde vão as informações sobre o token.

```
{
  "alg" : "HS256"  //algoritmo de criptografia utilizado para assinar o token. Outros exemplos: HMAC, SHA256 ou RSA.
  "typ" : "JWT"    //tipo do token
}
```

Obs.: os campos que carregam as informações como alg, typ entre outros são chamados de ***Claims***.

EXTRA:

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.

```
{
  "ativo" : true,
  "profile" : "admin"
}
```

*** PayLoad:** também no formato JSON, logo pode ser considerado um objeto JSON, carrega informações que são de interesse da aplicação. Estes claims podem ser ***reservados***, os quais são definidos pela especificação JWT e também ***públicos*** ou ***privados***. A aplicação pode adicionar novos claims neste caso são chamados de privados, pois interessam apenas aqueles que produzem e consome o token.

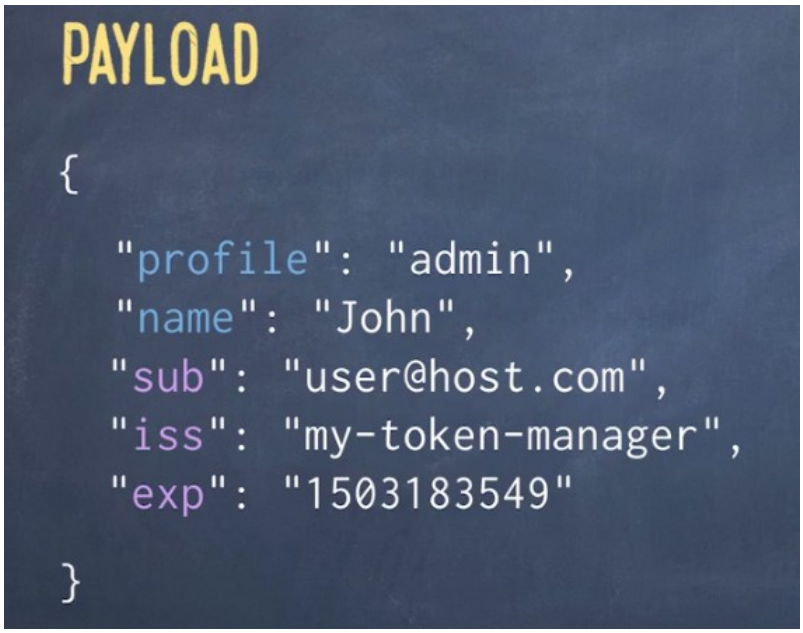
{

“sub” : “user@host.com” //subject => representa o principal interessado, ou seja, algo que identifique o usuário.

“iss” : “my-token-manager” //issuer => aquele que gera o token

“exp” : “1530984702” //expire => tempo de expiração do token em segundos

}



profile e name são exemplos de private claims.

EXTRA:

**** 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.

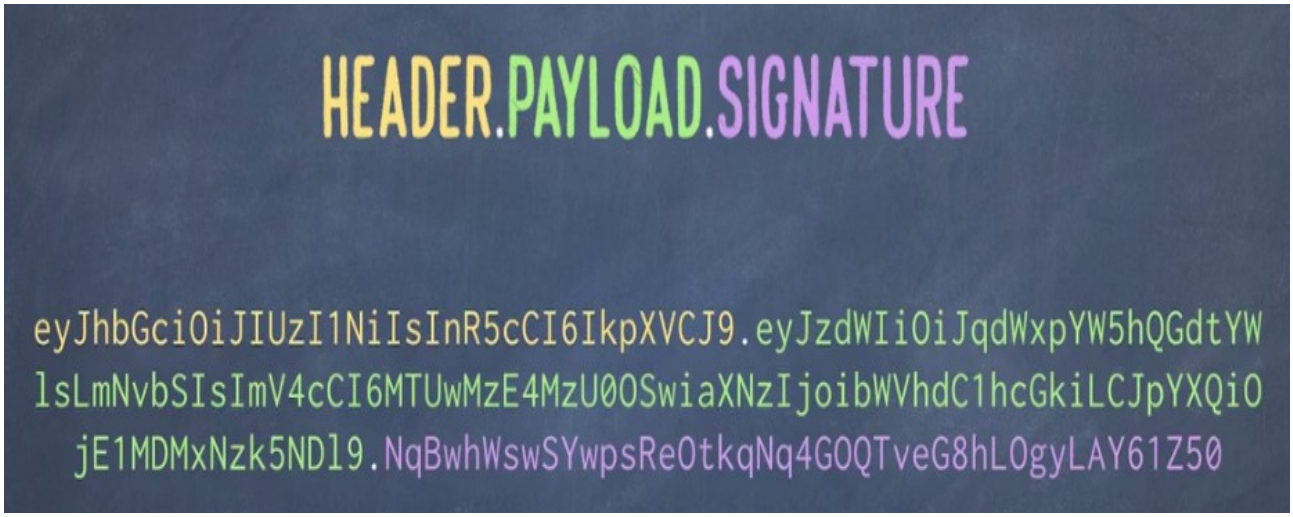
Apenas quem está de posse da chave pode criar, alterar e validar o token.

Como é formado o token...

Para formar o token o header e o payload são codificados em "base64 url" e unidos por um ponto.
A assinatura é opcional, porém se feita será a terceira parte do token, unida também por um ponto.

Exemplo:

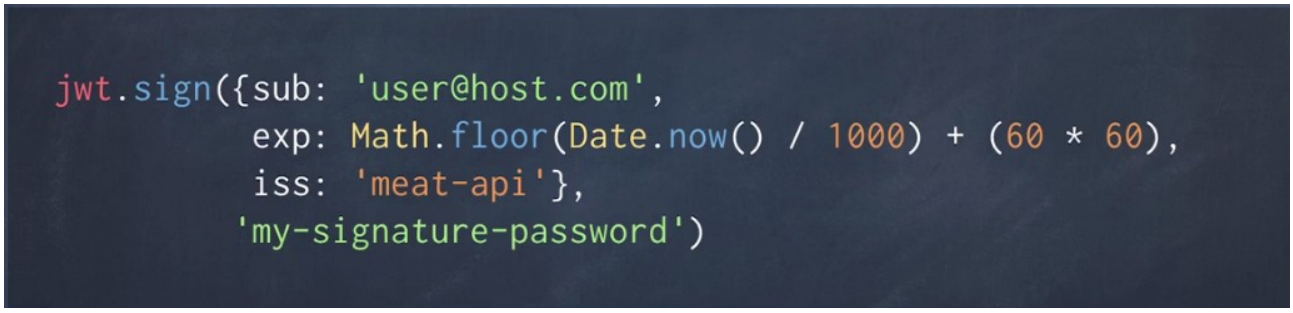
Header.PayLoad.Signature



Como o token é gerado...

Existe uma biblioteca fornecida pela API Auth Zeton capaz de gerar o token.

```
jwt.sign( //sign é a função capaz de criar o corpo de um token com um objeto JSON e assinar o token com uma senha.
            { //Payload =  corpo do token
                sub: 'user@host.com',  // portador do token – neste caso um usuario
                exp: Math.floor(Date.now() / 1000) + (60 + 60),  // tempo de expiração do token
                iss: 'meat-api'          // o emissor do token – no caso nossa aplicação meat-api
            },
            'my-signature-password'      // senha usada para a assinatura do token
        )
```

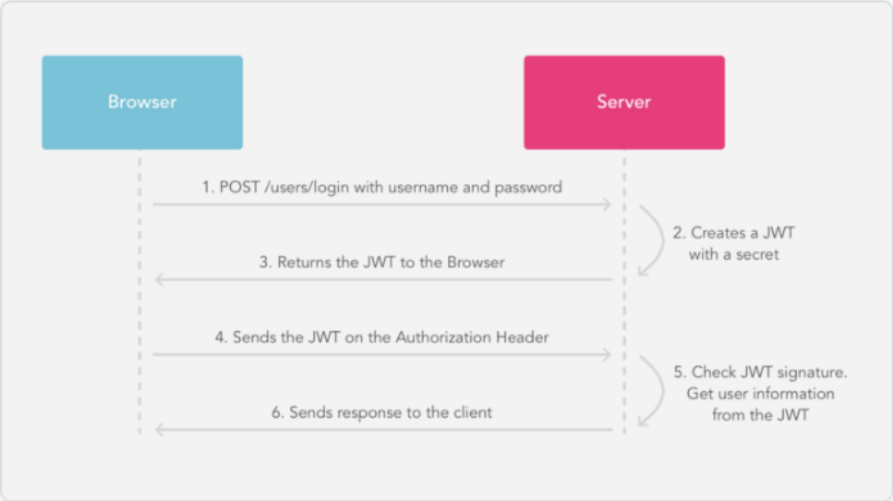


EXTRA:

Usando o token

Ao fazer login em um serviço de autenticação um token JWT é criado e retornado para o cliente. 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>



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.

Seção 14

Vídeo 114

Adicionando Token JWT a Rota de Login

1) Para criar nossa pequena aplicação com login importe as bibliotecas abaixo:

npm install json-server @types/json-server -D

npm install --save @types/json-server

npm install express

npm install --save @types/express

npm install jsonwebtoken

npm install --save @types/jsonwebtoken

2) **package.json**

```
{
 "name": "meat",
 "version": "1.0.0",
 "license": "MIT",
 "angular-cli": {},
 "scripts": {
   "ng": "ng",
   "start": "ng serve",
   "lint": "tslint \"src/**/*.ts\" --project src/tsconfig.json --type-check && tslint \"e2e/**/*.ts\" --project e2e/tsconfig.json --type-check",
   "test": "ng test",
   "pree2e": "webdriver-manager update --standalone false --gecko false",
   "e2e": "protractor"
 },
 "private": true,
 "dependencies": {
   "@angular/animations": "4.3.3",
   "@angular/common": "4.3.3",
   "@angular/compiler": "4.3.3",
   "@angular/core": "4.3.3",
   "@angular/forms": "4.3.3",
   "@angular/platform-browser": "4.3.3",
   "@angular/platform-browser-dynamic": "4.3.3",
   "@angular/router": "4.3.3",
   "admin-lte": "2.3.11",
   "core-js": "2.4.1",
   "express": "^4.17.1",
   "font-awesome": "4.7.0",
   "intl": "1.2.5",
   "jquery": "3.1.1",
   "reflect-metadata": "0.1.10",
   "rxjs": "5.4.2",
   "ts-helpers": "1.1.2",
   "web-animations-js": "2.2.5",
   "zone.js": "0.8.16"
```

```
  },
  "devDependencies": {
    "@angular/cli": "1.2.7",
    "@angular/compiler-cli": "4.3.3",
    "@types/express": "^4.0.37",
    "@types/jasmine": "2.5.53",
    "@types/json-server": "^0.14.0",
    "@types/jsonwebtoken": "^7.2.3",
    "@types/node": "7.0.5",
    "codelyzer": "3.1.2",
    "jasmine-core": "2.7.0",
    "jasmine-spec-reporter": "4.1.1",
    "json-server": "^0.12.0",
    "jsonwebtoken": "^7.4.1",
    "karma": "1.7.0",
    "karma-chrome-launcher": "2.2.0",
    "karma-cli": "1.0.1",
    "karma-jasmine": "1.1.0",
    "karma-remap-istanbul": "0.6.0",
    "protractor": "5.1.2",
    "ts-node": "3.3.0",
    "tslint": "5.5.0",
    "typescript": "2.4.2",
    "webdriver-manager": "12.0.6"
  }
}
```

3) backend/tsconfig.json

```
{
  "compileOnSave": true, /* compila o arquivo de ts para js ao salvar o arquivo ts */
  "compilerOptions": {
    /* Basic Options */
    "target": "es5",                    /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' or 'ESNEXT'. */
    "module": "commonjs",               /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
    // "lib": [],                       /* Specify library files to be included in the compilation. */
    "allowJs": true,                    /* Allow javascript files to be compiled. */
    // "checkJs": true,                 /* Report errors in .js files. */
    // "jsx": "preserve",               /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
```

```
// "declaration": true,                /* Generates corresponding '.d.ts' file. */
// "declarationMap": true,             /* Generates a sourcemap for each corresponding '.d.ts' file. */
// "sourceMap": true,                  /* Generates corresponding '.map' file. */
// "outFile": "./",                    /* Concatenate and emit output to single file. */
 "outDir": "dist",                    /* Redirect output structure to the directory. */
// "rootDir": "./",                    /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
// "composite": true,                  /* Enable project compilation */
// "removeComments": true,             /* Do not emit comments to output. */
// "noEmit": true,                     /* Do not emit outputs. */
// "importHelpers": true,              /* Import emit helpers from 'tslib'. */
// "downlevelIteration": true,         /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or 'ES3'. */
// "isolatedModules": true,            /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */

/* Strict Type-Checking Options */
"strict": true,                        /* Enable all strict type-checking options. */
// "noImplicitAny": true,              /* Raise error on expressions and declarations with an implied 'any' type. */
// "strictNullChecks": true,           /* Enable strict null checks. */
// "strictFunctionTypes": true,        /* Enable strict checking of function types. */
// "strictBindCallApply": true,        /* Enable strict 'bind', 'call', and 'apply' methods on functions. */
// "strictPropertyInitialization": true,  /* Enable strict checking of property initialization in classes. */
// "noImplicitThis": true,             /* Raise error on 'this' expressions with an implied 'any' type. */
// "alwaysStrict": true,               /* Parse in strict mode and emit "use strict" for each source file. */

/* Additional Checks */
// "noUnusedLocals": true,             /* Report errors on unused locals. */
// "noUnusedParameters": true,         /* Report errors on unused parameters. */
// "noImplicitReturns": true,          /* Report error when not all code paths in function return a value. */
// "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */

/* Module Resolution Options */
// "moduleResolution": "node",         /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
// "baseUrl": "./",                    /* Base directory to resolve non-absolute module names. */
// "paths": {},                        /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
// "rootDirs": [],                     /* List of root folders whose combined content represents the structure of the project at runtime. */
// "typeRoots": [],                    /* List of folders to include type definitions from. */
// "types": [],                        /* Type declaration files to be included in compilation. */
// "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, just typechecking. */
```

```
    "esModuleInterop": true,                /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for all imports. Implies
'allowSyntheticDefaultImports'. */
    // "preserveSymlinks": true,            /* Do not resolve the real path of symlinks. */

    /* Source Map Options */
    // "sourceRoot": "",                    /* Specify the location where debugger should locate TypeScript files instead of source locations. */
    // "mapRoot": "",                       /* Specify the location where debugger should locate map files instead of generated locations. */
    // "inlineSourceMap": true,             /* Emit a single file with source maps instead of having a separate file. */
    // "inlineSources": true,               /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--sourceMap' to be set. */

    /* Experimental Options */
    "experimentalDecorators": true,        /* Enables experimental support for ES7 decorators. */
    "emitDecoratorMetadata": true,         /* Enables experimental support for emitting type metadata for decorators. */
  }
}
```

4) backend/keys/cert.pem


```
-----BEGIN CERTIFICATE-----
MIIDtTCCAp2gAwIBAgIJAKDr6YbZ+6KXMA0GCSqGSIb3DQEBBQUAMEUxCzAJBgNV
BAYTAkFVMRMwEQYDVQQIEwpTb21lLVN0YXRlMSEwHwYDVQQKExhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTcwODA4MDI0NzA3WhcNMTcwOTA3MDI0NzA3WjBF
MQswCQYDVQQGEwJBVTETMBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl28gI9ABf
dN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+FGA9vXy4L
2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS0CRRYZGX
w08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrgsGgiH0Hn
jOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIwd3KhL6e5
aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABo4GnMIGkMB0GA1UdDgQWBBQebMNu
xfSChegKLYv2Dp7aVPYIRjB1BgNVHSMEbjBsgBQebMNuxfSChegKLYv2Dp7aVPYI
RqFJpEcwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNV
BAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZIIJAKDr6YbZ+6KXMAwGA1UdEwQF
MAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAIt5OHCBsYuIvB1UsI9rFkRunQ9eCORD
gk/dg6mtQaCUKqJRoQq+Zqi+y/z+Jxl1pgzudaOPwX/wLDM31PGWZbwzABR1uIR8
```

aQ/sDLw0yUdGvtUnrZOK4CVNQgl+zXNTmpmEx1JAT7xvXb5OPahdi/AYqVW6SWET

QCiRCNsqv3xqyStAURi8B0BVVJUTNbn9+CkBU7FDZpBjWCKUsMfm/4dm/u3eTxIS

oOf70F25k5Ws5I11usvlQY6/I17aGXU29voKws0v7iClFwqtOt8nntGi5I6OQrd5

2PvOp/kA2TPxi83nP0/WT0pzvM/ykzbxh4oKmCIzp6wXx8JaKIr4Xr0=

-----END CERTIFICATE-----

5) backend/keys/key.pem

-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAwyvtLiqJp2yqsoeRFex6b2KNgruU/yINe6/mEDrzZB29tnPl

28gI9ABfdN/cJxvO1mhVjcn81FM2OUJJ8+5pUUNyqQZGAjevF7Z/pOAm6nrd0S+F

GA9vXy4L2E9DTB82UnDPKLoZhwFK+gWW4KHGlyRoBALKmFUmKjaMuYu+MWb+ucIS

0CRRYZGXw08lNz5Sn46DzNT8czdqEjMAvZ9GEieKlwzjMtalDKb91CqRCr/IUJrg

sGgiH0HnjOouphZvM52hltxCbtrMXA/92FHIHK/gsRDpT9k7E39m6RIjtvTdQHIw

d3KhL6e5aa7kFkfcpcGGHyXd63VbJE+2+RHYCwIDAQABAoIBABvTp31tkzLLy4qo

yYbWsJ4w7vw0womd0U5RRfwKa6lalDHvVFaBrsUbSIDhagIpdNQdDBWrNuEO/Cwt

x12AIIP1meN/gXvPdKds3cYkL2g/HGE25UMI8FghuI6evnmb+aLHcsHbxsf45ze+

lERyzF3AUxQ/UaNmgIqEPIw34GYqiQB4ih6H1tQrKWVuvMtNc7o1kYOuTjR581h9

gXDKxGPkFyn2Y6xw1FDevAjzLvb4fJ/PLcTtEzbpX0+Jwpw/sPO5Vu9aj5Y/WpcF

k94ypnd8hR8J7kvjUmoPJwxHZ8wU75OCq+9KvIq8AU8LARVv9cK26drmrGEH5YEu

Dgy7uAECgYEA+l9yJZj1tzqYUoy8P70+jz5Ixrk771JForKBYv/rQL8gtpCOdVOy

2SrjgbvRKsf6mklTmJy3ljLNVuoFxLaiBauOH7zlC+twucdZhWOdKfj92WFr78d6

Txp1E2jUYijw+SVnmOm71IFqF9A1+vGnX+MV9awCEbzck5Km1IodsXkCgYEAx47f

atR/jRxmDKYhvU+i27fP8BxessBIW+jgP4qE+YaFNdUG0p2zjnco1G66DzAntiFB

pjYFzZFe6kFUIsHNeag3i0HQqqfTsYeOuFo5wA2yHxqhGj//aBlvjl2nScm4nNxL

e+IiT5lnYixsMeXp0wsXDjVRL+qI0ddCxMSamKMCgYEAuyOGNTX8SyUf32vT9Sxl

Szn5uZeQhdQPuokEDtRMqctyDyNDZ8zUvF/DRx/0Oq7Y0GOIeWSCEGEOfC1VfHAp

8GUHSQ+uf9dOHX40mnAXc9zpGIcp30JRWDjlU0minXAOlZrwsSnbO/4zJ3N4NYZU

XWZl7ofP0CRDFoah7fECOrkCgYAWeZPV/0rCRMGFiSxhmtd3sXL80uUjcp5zo32m

Cl7Bj5HvJyfWjqZLVEZH7pSL41uFmqc/JfRvW4D6FxN3uqX+nqBq0aXDopUn+GFt

h9PbOm+mKySbxTtpqPu0Z/3QRKTDHdhugmBk4J0RJHmnAQLdnvEgy4ZyRBLWt0yw

xoe2ywKBgHuDdsU4CW/FW3vNqMe1HLuIBeqHCWg5XSgl/MHLiflJ+tQxbKLTXtyU

z/sNijftjwWrs+NrzJvTWnlXSyLGFkKK/TotkHBDsEY5z+Rhu0Ndm20btkD4j7Hf

AeCSGnwPKKLLtNh6pVngBDiUbh+GkN+awwETQCDLgxqB94LJN02L

-----END RSA PRIVATE KEY-----

6) backend/auth.ts

```
import {Request, Response} from 'express' //npm install express //npm install --save @types/express
import { User, users } from './users';
import * as jwt from 'jsonwebtoken';//importando a biblioteca que criara e assinara o token //npm install jsonwebtoken //npm install --save @types/jsonwebtoken

//codigo que ira processar o post que fora feito para: https://localhost:3023/login
//handleAuthentication => eh uma constante que recebe o resultado de uma funcao
export const handleAuthentication = (req : Request, resp : Response) => {

   const user : User = req.body;
   //const user: User = req.body //a partir do postman enviaremos um objeto do tipo User com propriedades localizadas
                                  //no corpo do objeto Request: email e password

   if(isValid(user)){//valida se o usuario eh valido ou nao
      const dbUser: User = users[user.email];//users["juliana@gmail.com"] retornara a linha completa

      const token = jwt.sign({sub: dbUser.email, iss:'meat-api'},'meat-api-password') //token com corpo e password - neste exercicio nao iremos compartilhar
                                                                                        //o nosso password com o client neste caso um usuario, mas poderia ser
                                                                                        //tambem uma outra aplicacao.
                                                                                        //sub: dbUser.email => portador o token, ou seja, quem recebera o token
                                                                                        //iss: 'meat-api' => emissor do token, quem esta enviando o token

      resp.json( { name: dbUser.name, email: dbUser.email, accessToken: token } ) //retornando os dados do usuario no corpo do objeto Response + o token criado
   }else{
      resp.status(403).json({message: 'Dados invalidos!'}) //um requisicao pelo postman sem nada no body teria este resultado
   }

}

function isValid(user: User) : boolean {

   if(!user){//se o usuario NAO existir
     return false;
   }
   const dbUser : User= users[user.email] //mesmo que haja erro de compilacao rode a aplicacao
   return dbUser !== undefined && dbUser.matches(user);
}

//Erros:
//400 = erro de má formacao no request.
//401 = nao autorizado - recurso protegido / segundo a especificacao http devemos enviar um novo header informando
//qual metodo de autenticacao foi utilizado.
//403 = informacao invalida.
```

7) backend/users.ts

```
export class User {
   constructor(public email: string, public name: string, public password: string ){
   }
   //perceba que o metodo esta dentro da classe User
   matches(another: User): boolean { //compara o User que vem do body com um elemento da const - cada linha eh um elemento
      return another !== undefined && another.email === this.email && another.password == this.password;
   }
```

```
}
export const users = { //objeto JSON com tres propriedades sendo utilizado para simular nosso banco de dados
    "juliana@gmail.com" : new User('juliana@gmail.com', 'juliana', 'juliana23'),
    "fabio@gmail.com": new User('fabio@gmail.com',   'fabio',   'fabio23'),
    "amanda@gmail.com" : new User('amanda@gmail.com',  'amanda',  'amanda23')
}
//O acesso a cada elemento da constante "users" deve ser feito com base na variavel de declaracao do objeto User: users["juliana@gmail.com"]
//////////// No Postman faca:
//{
//                    "email" : "juliana@gmail.com",
//                    "password" : "juliana23"
//}
```

8) backend/server.ts

```
import * as jsonServer from 'json-server'  /* npm install json-server @types/json-server -D */
import {Request, Response} from 'express'//import {Express} from 'express'  /* original apresenta erro quando instalo //npm install json-server @types/json-server -D */


import * as fs from 'fs'
import * as https from 'https'
import { handleAuthentication } from './auth';

const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

//1)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//      response.json({message: 'ok'}) //enviando a mensagem ok para o postman
//})

//2)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//    response.send({message: 'Alright'}) //enviando a mensagem ok para o postman
//})
//3)
//middleware que pode ser associado a uma rota neste caso a url: /login - esta eh uma rota personalizada para
//fazer o login do usuario que sera constituido de email e senha atraves de um objeto JSON criado no postman
server.post('/login', handleAuthentication)

// Use default router
server.use(router)

const options = {
```

```
    cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado

    key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave

}

let port = 3023;

https.createServer(options, server).listen(port, () => {    //criando o servidor

    console.log('JSON Server is running on https://localhost:'+port)

})


//Enviar atraves do Postman

//{

//                "email" : "juliana@gmail.com",

//                "password" : "juliana23"

//}
```

Obs.: Rode a aplicação no Postman com a url abaixo:

POST: https://localhost:3023/login

Body:

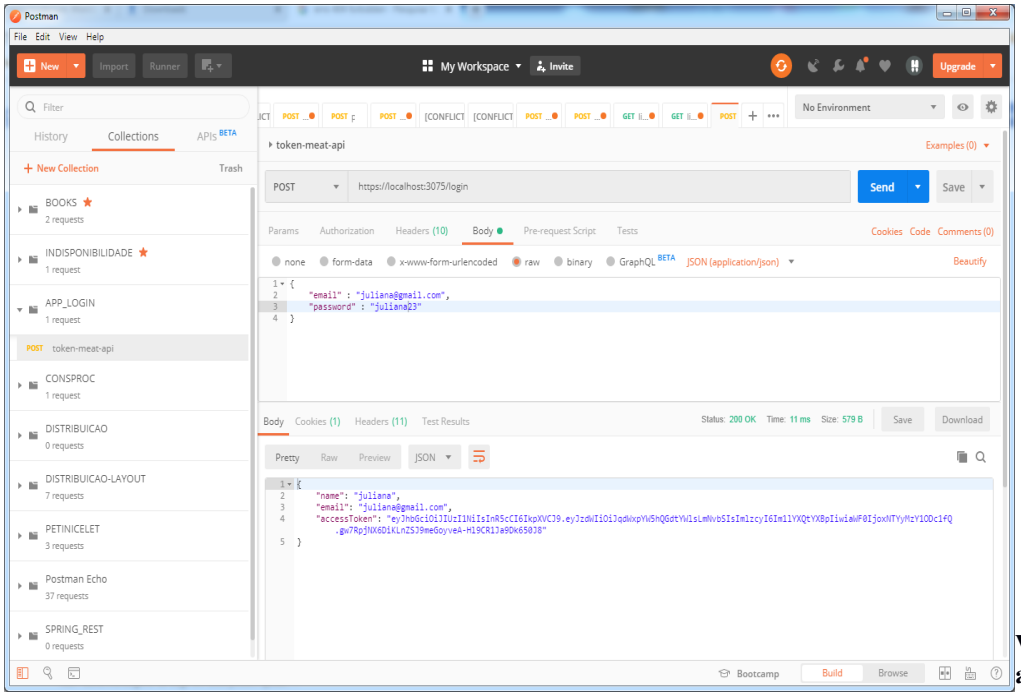
```
    {
                "email" : "juliana@gmail.com",
                "password" : "juliana23"
    }
```

Resultado obtido no Postman.

Body:


```
{
    "name": "juliana",
    "email": "juliana@gmail.com",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqdWxpYW5hQGdtYWlsLmNvbSIsImlzcyI6Im1lYXQtYXBpIiwiaWF0IjoxNTYyMTk4OTA0fQ.cJzoH-v_nE7EOEYrU_JI3CTbplZhjZyEgLMNRvycja8"
}
```


postman:



Viste o site abaixo para decodificar o token e obter as informações do Token:

https://jwt.io

https://jwt.io > Debugger

Seção 14

Vídeo 115

Middleware de Autorização

/package.json

```
{
  "name": "meat",
  "version": "1.0.0",
  "license": "MIT",
  "angular-cli": {},
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "lint": "tslint \"src/**/*.ts\" --project src/tsconfig.json --type-check && tslint \"e2e/**/*.ts\" --project e2e/tsconfig.json --type-check",
    "test": "ng test",
    "pree2e": "webdriver-manager update --standalone false --gecko false",
    "e2e": "protractor"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "4.3.3",
    "@angular/common": "4.3.3",
    "@angular/compiler": "4.3.3",
    "@angular/core": "4.3.3",
    "@angular/forms": "4.3.3",
    "@angular/platform-browser": "4.3.3",
    "@angular/platform-browser-dynamic": "4.3.3",
    "@angular/router": "4.3.3",
    "admin-lte": "2.3.11",
    "core-js": "2.4.1",
    "express": "^4.17.1",
    "font-awesome": "4.7.0",
    "intl": "1.2.5",
    "jquery": "3.1.1",
    "reflect-metadata": "0.1.10",
    "rxjs": "5.4.2",
    "ts-helpers": "1.1.2",
    "web-animations-js": "2.2.5",
    "zone.js": "0.8.16"
  },
  "devDependencies": {
```

```
    "@angular/cli": "1.2.7",
    "@angular/compiler-cli": "4.3.3",
    "@types/express": "^4.0.37",
    "@types/jasmine": "2.5.53",
    "@types/json-server": "^0.14.0",
    "@types/jsonwebtoken": "^7.2.3",
    "@types/node": "7.0.5",
    "codelyzer": "3.1.2",
    "jasmine-core": "2.7.0",
    "jasmine-spec-reporter": "4.1.1",
    "json-server": "^0.12.0",
    "jsonwebtoken": "^7.4.1",
    "karma": "1.7.0",
    "karma-chrome-launcher": "2.2.0",
    "karma-cli": "1.0.1",
    "karma-jasmine": "1.1.0",
    "karma-remap-istanbul": "0.6.0",
    "protractor": "5.1.2",
    "ts-node": "3.3.0",
    "tslint": "5.5.0",
    "typescript": "2.4.2",
    "webdriver-manager": "12.0.6"
  }
}
```

/tsconfig.json

```
{
  "compileOnSave": true, /* compila o arquivo de ts para js ao salvar o arquivo ts */
  "compilerOptions": {
    /* Basic Options */
    "target": "es5",                          /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017','ES2018' or 'ESNEXT'. */
    "module": "commonjs",                     /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', or 'ESNext'. */
    // "lib": [],                             /* Specify library files to be included in the compilation. */
    "allowJs": true,                          /* Allow javascript files to be compiled. */
    // "checkJs": true,                       /* Report errors in .js files. */
    // "jsx": "preserve",                     /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
    // "declaration": true,                   /* Generates corresponding '.d.ts' file. */
    // "declarationMap": true,                /* Generates a sourcemap for each corresponding '.d.ts' file. */
    // "sourceMap": true,                     /* Generates corresponding '.map' file. */
    // "outFile": "./",                       /* Concatenate and emit output to single file. */
     "outDir": "dist",                        /* Redirect output structure to the directory. */
    // "rootDir": "./",                       /* Specify the root directory of input files. Use to control the output directory structure with --outDir. */
    // "composite": true,                     /* Enable project compilation */
    // "removeComments": true,                /* Do not emit comments to output. */
    // "noEmit": true,                        /* Do not emit outputs. */
    // "importHelpers": true,                 /* Import emit helpers from 'tslib'. */
```

```
    // "downlevelIteration": true,           /* Provide full support for iterables in 'for-of', spread, and destructuring when targeting 'ES5' or 'ES3'. */
    // "isolatedModules": true,              /* Transpile each file as a separate module (similar to 'ts.transpileModule'). */

    /* Strict Type-Checking Options */
    "strict": true,                       /* Enable all strict type-checking options. */
    // "noImplicitAny": true,                /* Raise error on expressions and declarations with an implied 'any' type. */
    // "strictNullChecks": true,             /* Enable strict null checks. */
    // "strictFunctionTypes": true,          /* Enable strict checking of function types. */
    // "strictBindCallApply": true,          /* Enable strict 'bind', 'call', and 'apply' methods on functions. */
    // "strictPropertyInitialization": true,  /* Enable strict checking of property initialization in classes. */
    // "noImplicitThis": true,               /* Raise error on 'this' expressions with an implied 'any' type. */
    // "alwaysStrict": true,                 /* Parse in strict mode and emit "use strict" for each source file. */

    /* Additional Checks */
    // "noUnusedLocals": true,               /* Report errors on unused locals. */
    // "noUnusedParameters": true,           /* Report errors on unused parameters. */
    // "noImplicitReturns": true,            /* Report error when not all code paths in function return a value. */
    // "noFallthroughCasesInSwitch": true,    /* Report errors for fallthrough cases in switch statement. */

    /* Module Resolution Options */
    // "moduleResolution": "node",           /* Specify module resolution strategy: 'node' (Node.js) or 'classic' (TypeScript pre-1.6). */
    // "baseUrl": "./",                      /* Base directory to resolve non-absolute module names. */
    // "paths": {},                          /* A series of entries which re-map imports to lookup locations relative to the 'baseUrl'. */
    // "rootDirs": [],                       /* List of root folders whose combined content represents the structure of the project at runtime. */
    // "typeRoots": [],                      /* List of folders to include type definitions from. */
    // "types": [],                          /* Type declaration files to be included in compilation. */
    // "allowSyntheticDefaultImports": true,  /* Allow default imports from modules with no default export. This does not affect code emit, just typechecking. */
    "esModuleInterop": true,                 /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for all imports. Implies 'allowSyntheticDefaultImports'.
*/
    // "preserveSymlinks": true,             /* Do not resolve the real path of symlinks. */

    /* Source Map Options */
    // "sourceRoot": "",                     /* Specify the location where debugger should locate TypeScript files instead of source locations. */
    // "mapRoot": "",                        /* Specify the location where debugger should locate map files instead of generated locations. */
    // "inlineSourceMap": true,              /* Emit a single file with source maps instead of having a separate file. */
    // "inlineSources": true,                /* Emit the source alongside the sourcemaps within a single file; requires '--inlineSourceMap' or '--sourceMap' to be set. */

    /* Experimental Options */
    "experimentalDecorators": true,        /* Enables experimental support for ES7 decorators. */
    "emitDecoratorMetadata": true,         /* Enables experimental support for emitting type metadata for decorators. */
  }
}
```

/backend/server.ts

```
import * as jsonServer from 'json-server'  /* npm install json-server @types/json-server -D */
import {Request, Response} from 'express'//import {Express} from 'express'  /* original apresenta erro quando instalo //npm install json-server @types/json-server -D */


import * as fs from 'fs'
import * as https from 'https'
import { handleAuthentication } from './auth';
import { handleAuthorization } from './authz';
```

```
const server = jsonServer.create() //const server: Express = jsonServer.create() - original apresenta erro
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)
// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)

//1)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//      response.json({message: 'ok'}) //enviando a mensagem ok para o postman
//})

//2)
// metodo http por onde passamos nossa rota de login - 1 parametro a url, segundo parametro uma funcao
//server.post('/login', (request, response) =>{
//    response.send({message: 'Alright'}) //enviando a mensagem ok para o postman
//})

//3)
//middleware que pode ser associado a uma rota neste caso a url: /login - esta eh uma rota personalizada para
//fazer o login do usuario que sera constituido de email e senha atraves de um objeto JSON criado no postman
server.post('/login', handleAuthentication)

server.use('/orders', handleAuthorization)//o metodo "use(...)" foi usado para que nossa middleware
                                    //de autorizacao sirva a todos os metodos http

// Use default router
server.use(router)

const options = {
    cert: fs.readFileSync('./backend/keys/cert.pem'), //caminho do certificado
    key: fs.readFileSync('./backend/keys/key.pem') //caminho da chave
}

let port = 3099; //porta onde esta aplicação esta respondendo


https.createServer(options, server).listen(port, () => {    //criando o servidor - perceba que o servidor esta sendo criado com oprotocolo em https. //https.createServer
    console.log('JSON Server is running on https://localhost:'+port)  //rodar com a url que esta sendo exibida no terminal
})

//Enviar atraves do Postman
//{
//                "email" : "juliana@gmail.com",
//                "password" : "juliana23"
//}
```

/backend/users.ts

```
export class User {

    constructor(public email: string, public name: string, public password: string ){
    }

    //perceba que o metodo esta dentro da classe User
    matches(another: User): boolean { //compara o User que vem do body com um elemento da const - cada linha eh um elemento
        return another !== undefined && another.email === this.email && another.password == this.password;
    }
}

//{[key:string]: User} ==> tipagem usada na constante users
//{[key:string]: User} ==> chave do tipo string e valor do tipo User

export const users: {[key:string]: User} = { //objeto JSON com tres propriedades sendo utilizado para simular nosso banco de dados
    "juliana@gmail.com" : new User('juliana@gmail.com', 'juliana', 'juliana23'),
    "fabio@gmail.com": new User('fabio@gmail.com',   'fabio',   'fabio23'),
    "amanda@gmail.com" : new User('amanda@gmail.com',  'amanda',  'amanda23')
}

//O acesso a cada elemento da constante "users" deve ser feito com base na variavel de declaracao do objeto User: users["juliana@gmail.com"]

//////////////// No Postman faca:
//{
//                    "email" : "juliana@gmail.com",
//                    "password" : "juliana23"
//}
```

/backend/auth.ts

```
import {Request, Response} from 'express' //npm install express //npm install --save @types/express
import { User, users } from './users';

import * as jwt from 'jsonwebtoken';//importando a biblioteca que criara e assinara o token //npm install jsonwebtoken //npm install --save @types/jsonwebtoken
import {apiConfig} from './api-config'



//codigo que ira processar o post que fora feito para: https://localhost:3023/login
//handleAuthentication => eh uma constante que recebe o resultado de uma funcao
export const handleAuthentication = (req : Request, resp : Response) => {

    const user : User = req.body;
    //const user: User = req.body //a partir do postman enviaremos um objeto do tipo User com propriedades localizadas
                            //no corpo do objeto Request: email e password

    if(isValid(user)){//valida se o usuario eh valido ou nao
        const dbUser: User = users[user.email];//users["juliana@gmail.com"] retornara a linha completa


                    //jwt.sign ==> cria e assina o token
        const token = jwt.sign({sub: dbUser.email, iss:'meat-api'}, apiConfig.secret) //token com corpo e password - neste exercicio nao iremos compartilhar
                                                                    //o nosso password com o client neste caso um usuario, mas poderia ser
                                                                    //tambem uma outra aplicacao.
```

```
                    //sub: dbUser.email => portador o token, ou seja, quem recebera o token
                    //iss: 'meat-api' => emissor do token, quem esta enviando o token

      resp.json( { name: dbUser.name, email: dbUser.email, accessToken: token } )//retornando os dados do usuario no corpo do objeto Response + o token criado
   }else{
      resp.status(403).json({message: 'Dados invalidos!'}) //um requisicao pelo postman sem nada no body teria este resultado
   }

}


function isValid(user: User) : boolean {

   if(!user){//se o usuario NAO existir
     return false;
   }
   const dbUser : User= users[user.email] //mesmo que haja erro de compilacao rode a aplicacao
   return dbUser !== undefined && dbUser.matches(user);
}

//Erros:
//400 = erro de má formacao no request. / bad request
//401 = nao autorizado - recurso protegido / segundo a especificacao http devemos enviar um novo header informando
//qual metodo de autenticacao foi utilizado.
//403 = informacao invalida. / forbidden
```

/backend/authz.ts

```
import {Request, Response} from 'express'
import * as jwt from 'jsonwebtoken'
import {apiConfig} from './api-config'

export const handleAuthorization = (req: Request, resp: Response, next: any) => { //next ==> eh uma funcao de callback


   const token = extractToken(req) //funcao usada para extrair o token do request
    if(!token) {//se o token nao existir
      resp.setHeader('WWW-Authenticate', 'Bearer token_type="JWT"')//informando que a aplicacao aguarda um token do tipo JWT. Aqui temos (chave, valor)
      resp.status(401).json({message: 'Voce precisa se autenticar!!!'})//se o request for enviado sem o token ou se voce nao esta autenticado
          //o status 403 tambem seria aceitavel
    }else{
      jwt.verify(token, apiConfig.secret, (error, decoded)=>{
          if(decoded) {//se o token tiver sido decodificado entao
             next()   //se passou no if esta tudo ok e a função next() permitira o request passar para retornar o recurso '/orders'
          }else{       //apiConfig.secret ==> password para o token
             resp.status(403).json({message: "Não autorizado!!!"})
          }

      }) //o metodo "verify" verifica e decodifica o token | o metodo verify recebe o token e o password do token
    }  //(error, decoded) ==> eh uma funcao de callback perceba que a mesma esta
      //sendo passada como parametro de uma outra funcao neste caso da funcao "verify"
}
```

```
function extractToken(req: Request): string{ //funcao que retornara o token de acesso
   let token: any = undefined; //let token = undefined; //original


   if(req.headers && req.headers.authorization){//o token vem no hearders do request
    //Autorization: Bearer XXX.XX.XXX ==> assim deve vir o token no request
    //Bearer XXX.XX.XXX ==> sera a partir deste trecho que iremos extrair o valor do token que vira do request


    const partOne = req.headers.authorization.slice(0,req.headers.authorization.indexOf(' '));
    const partTwo = req.headers.authorization.slice(req.headers.authorization.indexOf(' ')+1);
    //const parts: string[] = req.headers.authorization.split(' ') //original nao esta funcionando


            //a partir do .authorization obtemos um array uma vez que
            //estamos dividindo os elementos deste array quando for
            //for encontrado espaço. Veja: "Bearer XXX.XX.XXX" aqui
            //temos um espaço, logo teremos dois elementos.
      if(partOne !== undefined && partOne === 'Bearer' ){
         token = partTwo;
      }


     // if(parseInt.length === 2 && parts[0] === 'Bearer'){  //if original
     //     token = parts[1]
     // }


   }
   return token;
}

//Erros:
//400 = erro de má formacao no request. / bad request
//401 = nao autorizado - recurso protegido / segundo a especificacao http devemos enviar um novo header informando
//qual metodo de autenticacao foi utilizado.
//403 = informacao invalida. / forbidden
```

/backend/api-config.ts

```
//constante que guarda a senha para codificacao e decodificacao do tokwn
export const apiConfig = {
   secret: 'meat-api-password'
}
```

/db.json

```
{
  "restaurants": [
    {
      "id": "bread-bakery",
      "name": "Bread & Bakery",
      "category": "Padaria",
```

"deliveryEstimate": "25m",
"rating": 4.9,
"imagePath": "assets/img/restaurants/breadbakery.png",
"about": "A Bread & Brakery tem 40 anos de mercado. Fazemos os melhores doces e paes. Compre e confira.",
"hours": "Funciona de segunda a sexta, de 8h as 23h"
},
{
"id": "burger-house",
"name": "Burger House",
"category": "Hamburgers",
"deliveryEstimate": "100m",
"rating": 3.5,
"imagePath": "assets/img/restaurants/burgerhouse.png",
"about": "40 anos se especializando em trash food.",
"hours": "Funciona todos os dias, de 10h as 22h"
},
{
"id": "coffee-corner",
"name": "Coffee Corner",
"category": "Cafeteria",
"deliveryEstimate": "30-40m",
"rating": 4.8,
"imagePath": "assets/img/restaurants/coffeecorner.png",
"about": "A Coffe Corner foi eleita a melhor cafeteria da cidade.",
"hours": "Funciona de segunda a sabado, de 6h as 22h"
},
{
"id": "green-food",
"name": "Green Food",
"category": "Saudavel",
"deliveryEstimate": "75m",
"rating": 4.1,
"imagePath": "assets/img/restaurants/greenfood.png",
"about": "Comida saudavel a no Green Food. Compramos barato, vendemos caro. ;)",
"hours": "Somente em horario de almoao, das 11h as 15h"
},
{
"id": "ice-cream",
"name": "Ice Cream",
"category": "Sorvetes",
"deliveryEstimate": "40-65m",
"rating": 4.5,
"imagePath": "assets/img/restaurants/icy.png",
"about": "A Ice Cream a uma sorveteria inovadora.",
"hours": "Funciona todos os dias, de 10h as 1h"
},
{
"id": "tasty-treats",
"name": "Tasty Treats",
"category": "Doces",
"deliveryEstimate": "20m",
"rating": 4.4,

```
    "imagePath": "assets/img/restaurants/tasty.png",
    "about": "A doceria com mais tradiaao da cidade",
    "hours": "Funciona de segunda a sabado, de 8h as 23h"
  }
],
"menu": [
  {
    "id": "cup-cake",
    "imagePath": "assets/img/foods/cupcake.png",
    "name": "Cup Cake",
    "description": "Cup Cake recheado de Doce de Leite",
    "price": 8.7,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "donut",
    "imagePath": "assets/img/foods/donut.png",
    "name": "Donut",
    "description": "Coberto com chantilly",
    "price": 2.5,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "bread",
    "imagePath": "assets/img/foods/bread.png",
    "name": "Pao Artesanal Italiano",
    "description": "Pao artesanal com queijos italianos",
    "price": 15.9,
    "restaurantId": "bread-bakery"
  },
  {
    "id": "burger",
    "imagePath": "assets/img/foods/burger.png",
    "name": "Classic Burger",
    "description": "O classico. Nao tem como errar.",
    "price": 18.5,
    "restaurantId": "burger-house"
  },
  {
    "id": "fries",
    "imagePath": "assets/img/foods/fries.png",
    "name": "Batatas Fritas",
    "description": "Batatas fritas crocantes",
    "price": 5.5,
    "restaurantId": "burger-house"
  },
  {
    "id": "coke.png",
    "imagePath": "assets/img/foods/coke.png",
    "name": "Refrigerante",
    "description": "O refri mais gelado da cidade.",
    "price": 4.5,
```

"restaurantId": "burger-house"
},
{
"id": "capp",
"imagePath": "assets/img/foods/capp.png",
"name": "Cappuccino com Chantilly",
"description": "Tradicional cappuccino com chantilly",
"price": 9.9,
"restaurantId": "coffee-corner"
},
{
"id": "espreso",
"imagePath": "assets/img/foods/espreso.png",
"name": "Super Espreso",
"description": "Cafa espresso duplo. ",
"price": 12.5,
"restaurantId": "coffee-corner"
},
{
"id": "starb",
"imagePath": "assets/img/foods/starb.png",
"name": "Starbucks Copycat",
"description": "O mais pedido da casa. O verdadeiro cafa americano pura agua.",
"price": 15.6,
"restaurantId": "coffee-corner"
},
{
"id": "zlasagna",
"imagePath": "assets/img/foods/zlasagna.png",
"name": "Lasagna de Zucchini",
"description": "Lasagna deliciosa que se salva por causa do queijo",
"price": 14.9,
"restaurantId": "green-food"
},
{
"id": "greensandwich",
"imagePath": "assets/img/foods/greensandwich.png",
"name": "Sanduache de Frango",
"description": "Frango grelhado e salada",
"price": 10.5,
"restaurantId": "green-food"
},
{
"id": "ceasar",
"imagePath": "assets/img/foods/ceasar.png",
"name": "Salada Ceasar",
"description": "Classica salada ceasar.",
"price": 33.9,
"restaurantId": "green-food"
},
{
"id": "sunday",

"imagePath": "assets/img/foods/sunday.png",
"name": "Mega Sunday",
"description": "Sunday com varias bolas de sorvete a sua escolha",
"price": 21.5,
"restaurantId": "ice-cream"
},
{
"id": "chocpop",
"imagePath": "assets/img/foods/chocpop.png",
"name": "Picola de Brigadeiro",
"description": "Picola de brigadeiro recheado com caco.",
"price": 8.5,
"restaurantId": "ice-cream"
},
{
"id": "icestd.png",
"imagePath": "assets/img/foods/icestd.png",
"name": "Sorvete Simples",
"description": "Classico sorvete de rua onde todos os sabores sao idanticos.",
"price": 1.5,
"restaurantId": "ice-cream"
},
{
"id": "cup",
"imagePath": "assets/img/foods/cup.png",
"name": "Cup Cake de Choc. Branco",
"description": "Cup Cake de chocolate branco",
"price": 5.5,
"restaurantId": "tasty-treats"
},
{
"id": "straw",
"imagePath": "assets/img/foods/straw.png",
"name": "Bolo de Morango",
"description": "Bolo recheado e coberto com morangos",
"price": 40.5,
"restaurantId": "tasty-treats"
},
{
"id": "cake",
"imagePath": "assets/img/foods/cake.png",
"name": "Fatia de Bolo",
"description": "Fatia de bolo de morango",
"price": 5.9,
"restaurantId": "tasty-treats"
}
],
"reviews": [
{
"name": "Julia",
"date": "2017-01-23T18:25:43",
"rating": 4.5,

"comments": "Tudo muito bom, entrega no tempo certo",
"restaurantId": "bread-bakery"
},
{
"name": "Aline",
"date": "2016-12-12T18:25:43",
"rating": 3,
"comments": "NaO recomendo. O recheio a sa um creme, nao tem camarao e nem carne",
"restaurantId": "bread-bakery"
},
{
"name": "Renata",
"date": "2016-12-12T18:25:43",
"rating": 2,
"comments": "Nao foi muito agradavel ao nosso paladar, e eu e meu esposo no outro dia passamos super mal. Pode ata ter sido outra coisa, mas...",
"restaurantId": "bread-bakery"
},
{
"name": "Renata",
"date": "2016-12-12T18:25:43",
"rating": 5,
"comments": "Nunca deixou na mao, comida sempre deliciosa.",
"restaurantId": "bread-bakery"
},
{
"name": "Daniela",
"date": "2017-01-23T18:25:43",
"rating": 4,
"comments": "Sanduache muito bom! Ja pedi diversas vezes sempre bom!",
"restaurantId": "burger-house"
},
{
"name": "Laila",
"date": "2017-02-02T18:25:43",
"rating": 5,
"comments": "Melhor hambarguer que ja provei. Vai ser difacil voltar para mc e bk",
"restaurantId": "burger-house"
},
{
"name": "Suzana",
"date": "2016-12-12T18:25:43",
"rating": 3.5,
"comments": "Meu sanduache chegou frio e todo desmontado =\\",
"restaurantId": "burger-house"
},
{
"name": "Eric",
"date": "2017-01-23T18:25:43",
"rating": 4.5,
"comments": "Excelente opaao! Recomendo!",
"restaurantId": "coffee-corner"
},

{
"name": "Amanda",
"date": "2017-02-02T18:25:43",
"rating": 2.5,
"comments": "Demorou muito e o cafa veio gelado",
"restaurantId": "coffee-corner"
},
{
"name": "Carla",
"date": "2016-12-12T18:25:43",
"rating": 4.8,
"comments": "Tudo perfeito. Entrega, embalagem, comida. Resultado? Melhor custo beneficio",
"restaurantId": "coffee-corner"
},
{
"name": "Rodrigo",
"date": "2016-12-12T18:25:43",
"rating": 4,
"comments": "Muito rapida a entrega, mas a caro.",
"restaurantId": "coffee-corner"
},
{
"name": "Andra",
"date": "2016-12-12T18:25:43",
"rating": 4.5,
"comments": "Excelente como sempre!",
"restaurantId": "coffee-corner"
},
{
"name": "Joao",
"date": "2017-01-23T18:25:43",
"rating": 5,
"comments": "Excelente. Fiz o pedido com outro endereao e quando vi o erro liguei avisando. O motoqueiro cortou a cidade todinha e a comida ainda chegou quentinha (e deliciosa). Estou sempre pedindo e recomendando",
"restaurantId": "green-food"
},
{
"name": "Marcelo",
"date": "2017-02-02T18:25:43",
"rating": 4.5,
"comments": "Chegou muito rapido, antes mesmo da previsao",
"restaurantId": "green-food"
},
{
"name": "Fatinha",
"date": "2017-01-23T18:25:43",
"rating": 2,
"comments": "a gostei dos sabores q pedi e achei o preao muito elevado",
"restaurantId": "ice-cream"
},
{
"name": "Lucas",
"date": "2017-02-02T18:25:43",

"rating": 2.5,
"comments": "Esperar 1:45hr pra receber o pedido a INACEITaVEL",
"restaurantId": "ice-cream"
},
{
"name": "Jacira",
"date": "2017-02-01T18:25:43",
"rating": 4.5,
"comments": "Veio OK!",
"restaurantId": "ice-cream"
},
{
"name": "Ana",
"date": "2017-01-23T18:25:43",
"rating": 5,
"comments": "Rapido e sempre uma delicia!! Ja pedi 3x e sempre veio rapido e tudo certinho.",
"restaurantId": "tasty-treats"
},
{
"name": "Larissa",
"date": "2017-02-02T18:25:43",
"rating": 2.5,
"comments": "A taxa de entrega poderia ser mais barata.",
"restaurantId": "tasty-treats"
},
{
"name": "Jarbas",
"date": "2016-12-15T18:25:43",
"rating": 4.7,
"comments": "Excelente e chegou muito rapido!",
"restaurantId": "tasty-treats"
},
{
"name": "Rafaela",
"date": "2017-01-12T18:25:43",
"rating": 5,
"comments": "Chegou rapido e perfeito!",
"restaurantId": "tasty-treats"
},
{
"name": "Julia",
"date": "2016-12-12T18:25:43",
"rating": 1.5,
"comments": "Sempre gostei muito dos pedidos que eu fiz, mas dessa vez foi passimo e decepcionou muito. Demorou 30min a mais pra chegar do que o tempo maximo esperado.",
"restaurantId": "tasty-treats"
},
{
"name": "Maralia",
"date": "2016-12-20T18:25:43",
"rating": 4.5,
"comments": "Taxa de entrega bastante elevada e precisa colocar no app se a pessoa quer poraao adicional de algo (calda, crocante etc)",
"restaurantId": "tasty-treats"

```
    }
  ],
  "orders": [
    {
      "name": "Juliana",
      "email": "juliana@gmail.com",
      "emailConfirmation": "juliana@gmail.com",
      "address": "pontes",
      "number": "200",
      "optionalAddress": "",
      "paymentOption": "MON",
      "orderItems": [
        {
          "quantity": 1,
          "menuId": "cup-cake"
        }
      ],
      "id": 1
    },
    {
      "name": "Amanda",
      "email": "amanda@gmail.com",
      "emailConfirmation": "amanda@gmail.com",
      "address": "pontes",
      "number": "100",
      "optionalAddress": "",
      "paymentOption": "DEB",
      "orderItems": [
        {
          "quantity": 2,
          "menuId": "cup-cake"
        }
      ],
      "id": 2
    },
    {
      "email": "amanda@gmail.com",
      "password": "amanda23",
      "id": 3
    },
    {
      "email": "amanda@gmail.com",
      "password": "amanda23",
      "id": 4
    },
    {
      "email": "juliana@gmail.com",
      "password": "juliana23",
      "id": 5
    }
  ]
}
```

Seção 15

Vídeo 116

Preparando a Pagina de Login

app/security/login/login.componente.html

```
<!-- INÍCIO DO CONTEÚDO -->
<section class="content login-content" style="margin: 10% auto">
  <div class="login-logo">
      <b>MEAT</b> | Login
  </div>
  <div class="col-xs-12 col-sm-offset-3 col-sm-6 login-box-body">
  <p class="login-box-msg">Por favor, efetue o login para continuar</p>

  <form>
    <div class="form-group has-feedback">
        <input type="email" class="form-control" placeholder="Email">
        <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
    </div>
    <div class="form-group has-feedback">
        <input type="password" class="form-control" placeholder="Password">
        <span class="glyphicon glyphicon-lock form-control-feedback"></span>
    </div>
    <div class="row">
    <!-- /.col -->
    <div class="col-xs-offset-8 col-xs-4">
      <button type="submit" class="btn btn-primary btn-block btn-flat">Entrar</button>
    </div>
    <!-- /.col -->
    </div>
  </form>
  </div>
</section>
<!-- FIM DO CONTEÚDO -->
```

app/security/login/login.componente.ts

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

@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
```

```
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
//import { HttpModule } from '@angular/http';
import { HttpClientModule } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                                //de um novo modulo OrderModule
```

```
//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router';
import { LoginComponent } from './security/login/login.component'

//import {LocationStrategy, HashLocationStrategy} from '@angular/common'



@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
    //AboutComponent,
    RestaurantsComponent,
    RestaurantComponent,
    RestaurantDetailComponent,
    MenuComponent,
    ShoppingCartComponent,
    MenuItemComponent,
    ReviewsComponent,
    //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
    //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
    //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
    //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
    OrderSummaryComponent,
    LoginComponent,
    //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
  ],
  imports: [
    BrowserModule,
    //HttpModule, //==> exemplo de Core Module. Veja o github do Angular //Angular 4.0
    HttpClientModule, //Angular 4.3
    //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
    //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
```

```
    RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
    //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                        //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
    //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular nao
                //sendo necessario portanto fazer imports novamente neste modulo.


    SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou nao.Neste caso
                            //estamos importando. Veja o Modulo OrderModule onde nao importamos os
                            //serviços apenas os componentes declarados no SharedModule.
    //OrderModule => este modulo nao foi inserido aqui, pois nao queremos
    //que o mesmo seja carregado na inicializacao do projeto, porém faremos
    //este carregamento como lazy loading no app.routes.ts

    //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
  ],


   providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],
   //providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}], utilizado para evitar problemas de navegação ao
   //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url
   //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
   //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url


  //transferido para CoreModule
   //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
//import {OrderComponent} from './order/order.component'
import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'
import { LoginComponent } from './security/login/login.component';

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        {path: 'login', component: LoginComponent},
```

```
        //{path: 'about', component: AboutComponent},//esta rota nao sera mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule - rota responsavel pelo Lazy Loading
        {path: 'restaurants', component: RestaurantsComponent},
        //{path: 'order', component: OrderComponent},//esta rota nao sera mais usada uma vez que criamos o OrderModule
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                  {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a pagina menu
                        {path: 'menu', component: MenuComponent},         //como padrao quando for especificado
                        {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.
                ]},                                  //Veja: http://localhost:4200/restaurants/green-food
                                        //eh igual a http://localhost:4200/restaurants/green-food/menu
                                  //devido ao redirectTo.
  {path: 'order-summary', component: OrderSummaryComponent},
  {path: 'order', loadChildren: './order/order.module#OrderModule'}//rota para o Modulo OrderModule - rota responsavel pelo Lazy Loading
]
```

app/app.api.ts

```
import {environment} from '../environments/environment'  //importando o objeto environment localizado no arquivo "environment.ts"
//export const MEAT_API = 'http://localhost:3000'  // variável que define a nossa url de back-end da aplicação
export const MEAT_API = environment.api           //o que definira qual ambiente sera chamado sera o build, por exempo:
                                                  // >ng build - -prod / ng build - -dev / ng build - -outra_variavel_criada
```

app/environments/environment.ts

```
// The file contents for the current environment will overwrite these during build.
// The build system defaults to the dev environment which uses `environment.ts`, but if you do
// `ng build --env=prod` then `environment.prod.ts` will be used instead.
// The list of which env maps to which file can be found in `angular-cli.json`.
export const environment = {
  production: false,
  api: 'https://localhost:3080' //==> rodando em https - veja o arquivo backend/dist/server.ts
  // api: 'http://localhost:3080' ==> rodando em http
};
```

Comando necessário para rodar a aplicação "*meat-app-starter-59*"

```
>nodemon --watch backend backend/dist/server.js  //aqui, estamos monitorando o arquivo server.js caso haja alguma modificação no arquivo.
>ng serve //levantando a aplicação na porta default (4200) ==> ng serve  - -port NUMERO_DE_PORTA_DIFERENTE
```

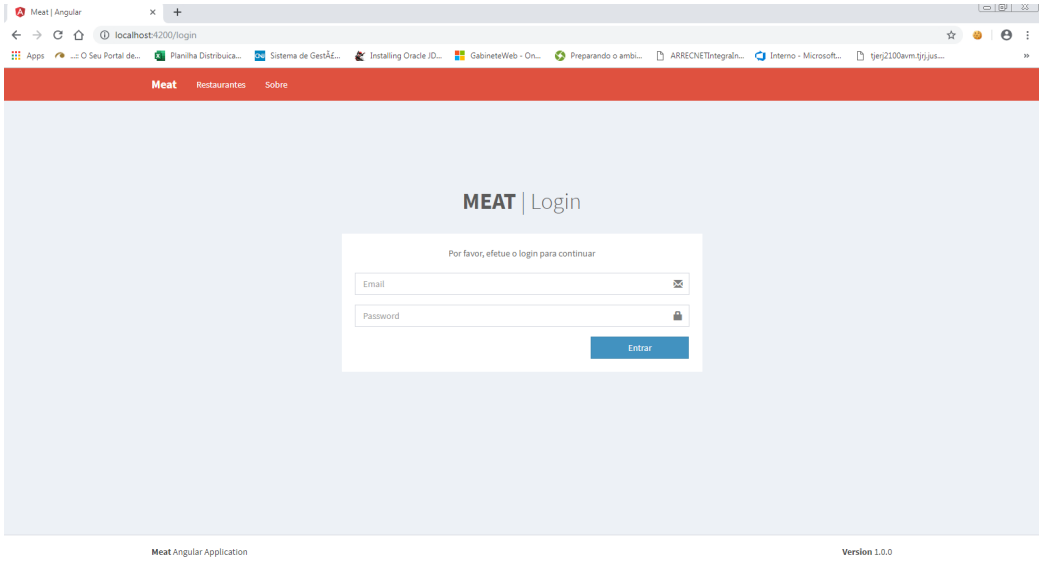
Não necessários:

```
meat-app-starter-59\src\app>tsc -w
meat-app-starter-59>node backend/dist/server  //ao executarmos o arquivo "server.js" sem extensão estaremos inicializando o json-server
                                              // a partir do script contido em server.js
```

No navegador faça:

localhost:4200

localhost:4200/login



localhost:4200/login

Seção 15

Vídeo 117

Criando o Componente de Login

app/security/login/login.componente.html

```
<!-- INÍCIO DO CONTEÚDO -->
<section class="content login-content" style="margin: 10% auto">
 <div class="login-logo">
    <b>MEAT</b> | Login
 </div>
 <div class="col-xs-12 col-sm-offset-3 col-sm-6 login-box-body">
 <p class="login-box-msg">Por favor, efetue o login para continuar</p>

 <form [formGroup]="loginForm"> <!-- Aqui, estamos associando o LoginForm com o form -->

   <mt-input-customizado-container [showTip]="false">
    <input type="text" formControlName="email" class="form-control" placeholder="Email"><!-- type="email" para type="text" para o browser nao tentar validar-->
    <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
   </mt-input-customizado-container>

   <mt-input-customizado-container [showTip]="false"> <!-- property binding -->
    <input type="password" formControlName="password" class="form-control" placeholder="Password">
    <span class="glyphicon glyphicon-lock form-control-feedback"></span>
   </mt-input-customizado-container>

 <div class="row">
 <!-- /.col -->
```

```
    <div class="col-xs-offset-8 col-xs-4">
      <button type="submit" [disabled]="loginForm.invalid" class="btn btn-primary btn-block btn-flat">Entrar</button>
    </div>
    <!-- /.col -->
    </div>
  </form>
  </div>
</section>
<!-- FIM DO CONTEÚDO -->
```

app/security/login/login.componente.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit { //trabalhando com Reactive Form

  loginForm: FormGroup

  constructor(private fb: FormBuilder,) { }

  ngOnInit() {
    this.loginForm = this.fb.group({
      email: this.fb.control('',[Validators.required, Validators.email]),
      password: this.fb.control('',[Validators.required])
    })
  }

}
```

app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
//import { HttpModule } from '@angular/http';
import { HttpClientModule } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
```

```
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                        //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                        //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                        //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                        //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router';
import { LoginComponent } from './security/login/login.component'

//import {LocationStrategy, HashLocationStrategy} from '@angular/common'



@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    HomeComponent,
```

```
  //AboutComponent,
  RestaurantsComponent,
  RestaurantComponent,
  RestaurantDetailComponent,
  MenuComponent,
  ShoppingCartComponent,
  MenuItemComponent,
  ReviewsComponent,
  //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
  //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
  //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
  OrderSummaryComponent,
  LoginComponent,
  //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
 ],
imports: [
  BrowserModule,
  //HttpModule, //==> exemplo de Core Module. Veja o github do Angular //Angular 4.0
  HttpClientModule, //Angular 4.3
  //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
  RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
  //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                        //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
  //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular nao
              //sendo necessario portanto fazer imports novamente neste modulo.


  SharedModule.retornaNossosServicos()//Com esta funcao temos a possibilidade de importar os serviços ou nao.Neste caso
                          //estamos importando. Veja o Modulo OrderModule onde nao importamos os
                          //serviços apenas os componentes declarados no SharedModule.
  //OrderModule => este modulo nao foi inserido aqui, pois nao queremos
  //que o mesmo seja carregado na inicializacao do projeto, porem faremos
  //este carregamento como lazy loading no app.routes.ts

  //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule
 ],

 providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],
 //providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}], utilizado para evitar problemas de navegação ao
 //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url
 //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
 //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url

//transferido para CoreModule
```

```
  //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo

  bootstrap: [AppComponent]

})

export class AppModule { }
```

app/app.routes.ts

```
import {Routes} from '@angular/router'

import {HomeComponent} from './home/home.component'

//import {AboutComponent} from './about/about.component'

import {RestaurantsComponent} from './restaurants/restaurants.component'

import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'

import {MenuComponent} from './restaurant-detail/menu/menu.component'

import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'

//import {OrderComponent} from './order/order.component'

 import {OrderSummaryComponent } from './order-summary/order-summary.component'

import {AboutModule} from './about/about.module'

import { LoginComponent } from './security/login/login.component';


export const ROUTES: Routes = [


        {path: '', component: HomeComponent},

        {path: 'login', component: LoginComponent},

        //{path: 'about', component: AboutComponent},//esta rota nao sera mais usada uma vez que criamos o AboutModule

        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule - rota responsavel pelo Lazy Loading

        {path: 'restaurants', component: RestaurantsComponent},

        //{path: 'order', component: OrderComponent},//esta rota nao sera mais usada uma vez que criamos o OrderModule

        {path: 'restaurants/:id', component: RestaurantDetailComponent,

                children: [  // rotas de navegacao filhas

                    {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a pagina menu

                            {path: 'menu', component: MenuComponent},          //como padrao quando for especificado

                            {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.

                ]},                                    //Veja: http://localhost:4200/restaurants/green-food

                                        //eh igual a http://localhost:4200/restaurants/green-food/menu

                                 //devido ao redirectTo.

    {path: 'order-summary', component: OrderSummaryComponent},

    {path: 'order', loadChildren: './order/order.module#OrderModule'}//rota para o Modulo OrderModule - rota responsavel pelo Lazy Loading

]
```

app/app.api.ts

```
import {environment} from '../environments/environment'  //importando o objeto environment localizado no arquivo "environment.ts"

//export const MEAT_API = 'http://localhost:3000'  // variável que define a nossa url de back-end da aplicação

export const MEAT_API = environment.api            //o que definira qual ambiente sera chamado sera o build, por exempo:
                                                   // >ng build - -prod / ng build - -dev / ng build - -outra_variavel_criada
```

app/environments/environment.ts

```
// The file contents for the current environment will overwrite these during build.
// The build system defaults to the dev environment which uses `environment.ts`, but if you do
// `ng build --env=prod` then `environment.prod.ts` will be used instead.
// The list of which env maps to which file can be found in `angular-cli.json`.
export const environment = {
  production: false,
  api: 'https://localhost:3080' //==> rodando em https - veja o arquivo backend/dist/server.ts
  // api: 'http://localhost:3080' ==> rodando em http
};
```

app/shared/input/input.component.ts

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

//import {NgModel} from '@angular/forms';

import {FormControlName} from '@angular/forms';

@Component({
  selector: 'mt-input-customizado-container', //selector: 'mt-input-customizado-container',
  templateUrl: './input-customizado.component.html'
})
export class InputCustomizadoComponent implements OnInit, AfterContentInit {

 meuInputGenerico : any
 @Input() label : string
 @Input() errorMessage : string
 @Input() showTip: boolean = true  //atributo criado na aula: 117. Criando o Componente de login.mp4

// @ContentChild(NgModel) model: NgModel //==> Usado em Template Forms
 @ContentChild(FormControlName) control: FormControlName //==> Usado em Reactive Forms
  constructor() { }

  ngOnInit() {
  }

  ngAfterContentInit(){
    //this.meuInputGenerico = this.model || this.control  //==> tentar pegar uma das duas
                this.meuInputGenerico = this.control  //this.meuInputGenerico = this.model
                if(this.meuInputGenerico === undefined){
                            throw new Error('Esse componente precisa ser usado com uma diretiva FormControlName')
    //throw new Error('Esse componente precisa ser usado com uma diretiva NgModel')
    //throw new Error('Esse componente precisa ser usado com uma diretiva NgModel ou FormControlName')
                 }
  }

  hasSuccess() : boolean {
```

```
        return this.meuInputGenerico.valid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }

  hasError() : boolean {
        return this.meuInputGenerico.invalid && (this.meuInputGenerico.dirty || this.meuInputGenerico.touched)
  }

}
```

app/shared/input/input.component.html

```
<div class="form-group has-feedback" [class.has-success]="hasSuccess()" [class.has-error]="!meuInputGenerico.valid && (meuInputGenerico.dirty || meuInputGenerico.touched)">

<label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> {{label}}</label>

<ng-content></ng-content><!--similar ao componente <router-outlet> aqui entrara um componente qualquer dinamicamente por exemplo:  order.component.html-->

<span class="help-block" *ngIf="showTip && hasSuccess()">
<i class="fa fa-check"></i> Ok!!!
</span>

<span class="help-block" *ngIf="showTip && hasError()">
<i class="fa fa-remove"></i> {{errorMessage}}
</span>

</div>

<!--
<div class="form-group" [class.has-success]="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)"
        [class.has-error]="!meuInputGenerico.valid && (meuInputGenerico.dirty || iptAddress.touched)">
  <label class="control-label sr-only" for="inputSuccess"><i class="fa fa-check"></i> Endereço</label>
  <input type="text" class="form-control" name="address" required minlength="5" #iptAddress="ngModel" ngModel placeholder="Endereço" autocomplete="off" >
  <span class="help-block" *ngIf="iptAddress.valid && (iptAddress.dirty || iptAddress.touched)">
  <i class="fa fa-check"></i> Ok!!!
  </span>
   <span class="help-block" *ngIf="!iptAddress.valid && (iptAddress.dirty || iptAddress.touched)">
  <i class="fa fa-remove"></i> Campo obrigatório!!!
  </span>
</div>
-->
```

Comando necessário para rodar a aplicação "*meat-app-starter-60*"

```
>nodemon --watch backend backend/dist/server.js  //aqui, estamos monitorando o arquivo server.js caso haja alguma modificação no arquivo.
>ng serve //levantando a aplicação na porta default (4200) ==> ng serve  - -port NUMERO_DE_PORTA_DIFERENTE
```

Não necessários:

```
meat-app-starter-52\src\app>tsc -w
```

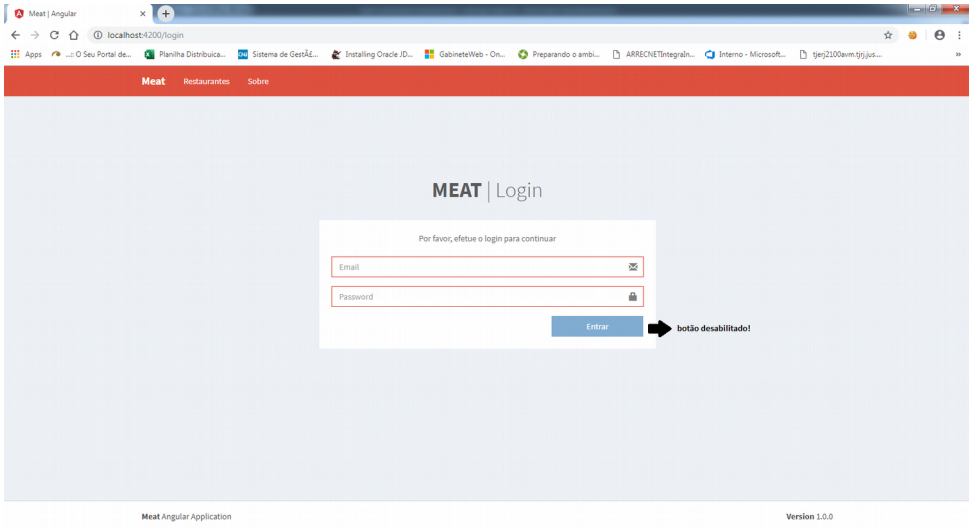
meat-app-starter-52>node backend/dist/server //ao executarmos o arquivo "server.js" sem extensão estaremos inicializando o json-server

// a partir do script contido em server.js

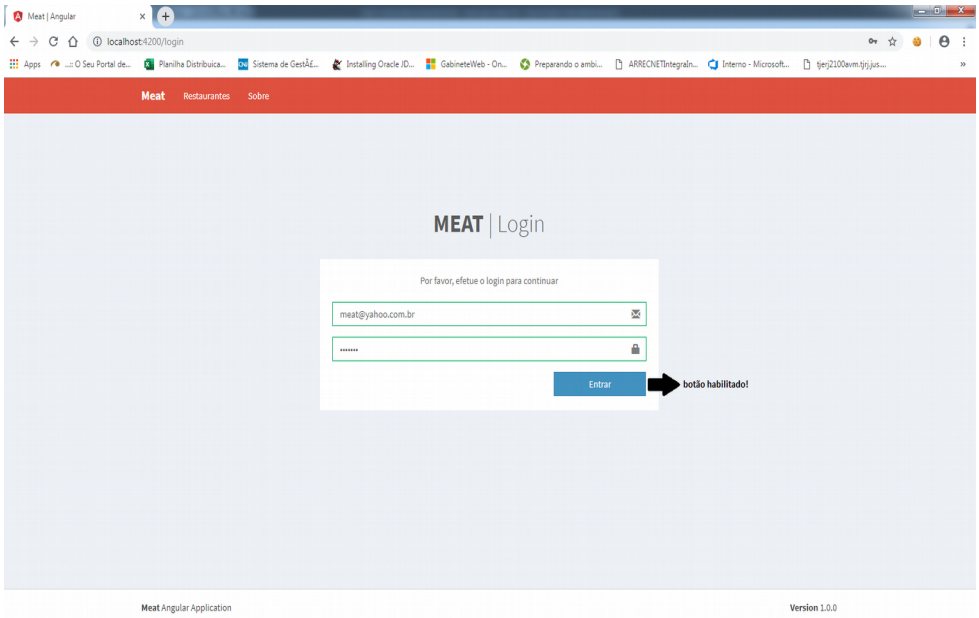
<u>No navegador faça:</u>

localhost:4200

localhost:4200/login



localhost:4200/login



localhost:4200/login

Seção 15

Vídeo 118

<u>Criando o Serviço de Login</u>

/app/security/login/login.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { MEAT_API } from '../../../app.api';
import { User } from "./user.model";
import 'rxjs/add/operator/do'  //com esta biblioteca poderemos guardar o usuario localmente para referencia-lo no restante da aplicacao


@Injectable()
export class LoginService {
   usuario: User; //com esta variavel guardaremos o usuario logado para referencia-lo no restante da aplicacao
   constructor(private http: HttpClient){
   }
   //O HttpClient nao exige mais especificar o content-type no header do request, pois o mesmo ja considere o JSON como default.
   login(email: string, password: string): Observable<any>{ // Observable<any> ==> mudaremos a tipagem nos próximos exercicios
      return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                  .do(user => this.usuario = user);//com este operador "do" poderemos guardar o usuario localmente.
   }
   isLoggedIn(): boolean { //metodo criado apenas para verificar se o usuario esta logado
     return this.usuario !== undefined //true => usuario logado / false => nao logado
   }


}
```

/app/security/login/user.models.ts

```
export interface User { //interface apenas para tipagem
   name: string,
   email: string,
   accessToken: string
}
```

/app/security/login/login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { LoginService } from './login.service';
import { User } from './user.model';
```

```
@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})

export class LoginComponent implements OnInit { //trabalhando com Reactive Form

  loginForm: FormGroup

 // constructor(private fb: FormBuilder,) { } no exercicio 53

  constructor(private fb: FormBuilder,private loginService: LoginService) { }  //no exericio 54 injetamos o servico de login

  ngOnInit() {
    this.loginForm = this.fb.group({
      email: this.fb.control('',[Validators.required, Validators.email]),
      password: this.fb.control('',[Validators.required])
    })
  }

 //this.loginForm.value.email, this.loginForm.value.password ==> pegando do nosso form em login.component.html
  login(){
    this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)//aqui, temos um Observable
                    .subscribe(user => console.log(user)); // neste exercicio por enquanto estamos apenas imprimindo no console
  }

}
```

/app/security/login/login.component.html

```
<!-- INÍCIO DO CONTEÚDO -->
<section class="content login-content" style="margin: 10% auto">
  <div class="login-logo">
      <b>MEAT</b> | Login
  </div>
  <div class="col-xs-12 col-sm-offset-3 col-sm-6 login-box-body">
  <p class="login-box-msg">Por favor, efetue o login para continuar</p>

  <form [formGroup]="loginForm">

      <mt-input-customizado-container [showTip]="false"><!-- [showTip]="false" objetivo de disabilitar os sinas de erro -->
        <input type="text" formControlName="email" class="form-control" placeholder="Email"><!-- type="email" para type="text" para o browser nao tentar validar-->
        <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
      </mt-input-customizado-container>
```

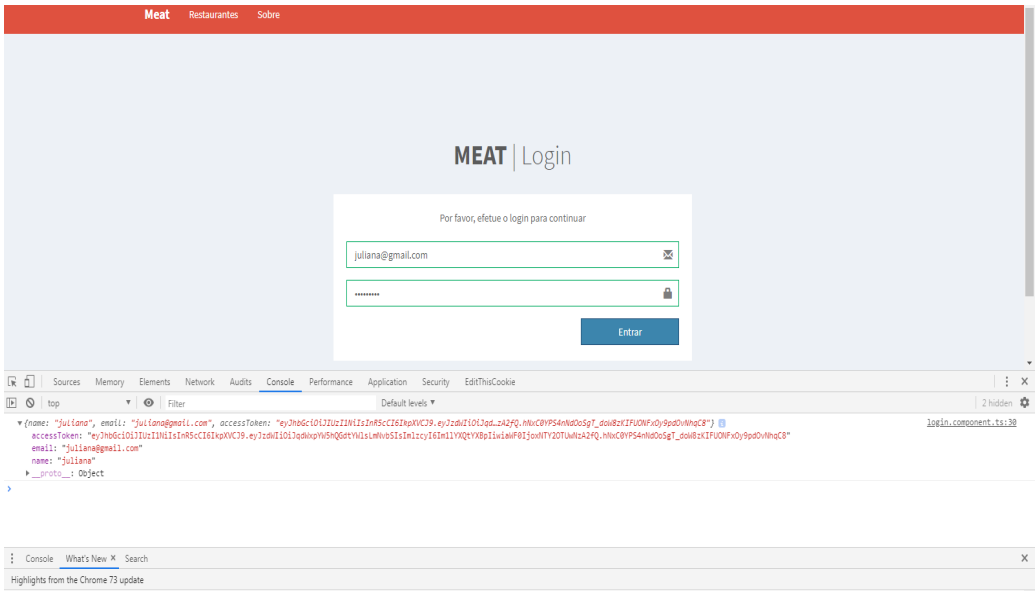
```
    <mt-input-customizado-container [showTip]="false"><!-- [showTip]="false" objetivo de disabilitar os sinas de erro -->
      <input type="password" formControlName="password" class="form-control" placeholder="Password">
      <span class="glyphicon glyphicon-lock form-control-feedback"></span>
    </mt-input-customizado-container>


   <div class="row">
   <!-- /.col -->
   <div class="col-xs-offset-8 col-xs-4">
     <button type="submit" [disabled]="loginForm.invalid" class="btn btn-primary btn-block btn-flat" (click)="login()">Entrar</button>
   </div>
   <!-- /.col -->
   </div>
  </form>
  </div>
</section>
<!-- FIM DO CONTEÚDO →
```

/app/shared/shared.module.ts

```
import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'

import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                  //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)



import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                  //permitira criar uma funcao capaz de retornar os
                                  //nosso servicos

import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';

import {LoginService} from '../security/login/login.service'

@NgModule({
  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent],//componentes que serao utilizados neste modulo
  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
               RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
```

```
            ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                CommonModule//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
//Perceba que neste modulo nao teremos navegacao.

static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                        //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                        //app.module.ts
  return {
          ngModule : SharedModule,
          providers: [RestaurantsService,ShoppingCartService, OrderService, LoginService] //declarando servicos na lista de providers
        }
 }

//FUNCAO ORIGINAL
//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                        //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                        //app.module.tsimport { LoginService } from './../security/login/login.service';

//   return {
//           ngModule : SharedModule,
//           providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//         }
//  }

}
```

localhost:4200/login



Comando necessário para rodar a aplicação "*meat-app-starter-60*"

```
>nodemon --watch backend backend/dist/server.js  //aqui, estamos monitorando o arquivo server.js caso haja alguma modificação no arquivo.
>ng serve //levantando a aplicação na porta default (4200) ==> ng serve  - -port NUMERO_DE_PORTA_DIFERENTE
```

Não necessários:

```
meat-app-starter-60\src\app>tsc -w
meat-app-starter-60>node backend/dist/server  //ao executarmos o arquivo "server.js" sem extensão estaremos inicializando o json-server
                                              // a partir do script contido em server.js
```

No navegador faça:

localhost:4200

localhost:4200/login

Seção 15

Vídeo 119

Realizando uma Compra com Autenticação

Nesta seção refinaremos um pouco mais a nossa implementação de login melhorando as mensagens em vez de imprimir-mos o usuário autenticado no console emitiremos o usuário autenticado através de notificações.

.../security/login/login.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { MEAT_API } from './../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado

@Injectable()
export class LoginService {//Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
                //que contem o name, login e o accessToken no nosso LoginService.

  userLog: User;

  constructor(private http:HttpClient){//com HttpClient nao precisamos mais informar o contentType
                //no header da requisicao uma vez que o default eh application/JSON.
  }
```

```
  isLoggedIn(): boolean{
    return this.userLog !== undefined;
  }

  //  login(email: string, password: string): Observable<any>{
  //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
  //  }

  login(email: string, password: string): Observable<User>{
    return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
            .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                                //de passar as informaçoes para o componente LoginComponent
  }

}
```

.../security/login/login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { LoginService } from './login.service';
import { NotificationService } from './../../shared/messages/notification.service';

@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit { //trabalhando com Reactive Form

  loginForm: FormGroup;

  constructor(private fb: FormBuilder,
          private loginService: LoginService,
          private notificationService: NotificationService) { }

  ngOnInit() {
    this.loginForm = this.fb.group({
      email: this.fb.control('',[Validators.required, Validators.email]),
      password: this.fb.control('',[Validators.required])
    })
  }

  //login(){ //ate aqui temos apenas um Observable
```

```
 //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password);
 //}


 //login(){
 //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
 //                .subscribe(user => console.log(user));
 //}


 login(){//No metodo subscribe(...) podemos passar tres callbacks
   this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
                .subscribe(user => this.notificationService.notify(`Bem vindo, ${user.name}`),
                        response  => //reponse eh do tipo HttpErrorResponse que representa o erro da resposta
                        this.notificationService.notify(response.error.message));//resp.status(403).json({message: 'Dados invalidos!'})
 }                                          //response.error ==> body da resposta
                                           //message ==> eh o atributo que contem a mensagem no body da resposta
}
```

.../order/order.service.ts

```
import {Injectable} from  '@angular/core';
import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';
import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';
import { Order, OrderItem } from './order.model';
//import { Http } from  '@angular/http'; // Angular 4.0
import { HttpClient, HttpHeaders } from  '@angular/common/http'; // Angular 4.3
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';
import {MEAT_API} from '../app.api';
//import { Headers, RequestOptions } from  '@angular/http'; // Angular 4.0
//import { Headers, RequestOptions } from  '@angular/common/http' // Angular 4.3 ja esta embutido no HttpClient


import { LoginService } from './../security/login/login.service';//com LoginService pegaremos a referencia do User autenticado


@Injectable() //injetando o servico ShoppingCartService
export class OrderService {


 // constructor(private cartService: ShoppingCartService, private http: Http){
 //}


  constructor(private cartService: ShoppingCartService,
            private http: HttpClient,
            private loginService: LoginService){
  }


  cartItems() : CartItem[]{
```

```
  return this.cartService.items
}

increaseQty(item : CartItem){
  return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
}

decreaseQty(item : CartItem){
  return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
}

remove(item : CartItem){
    this.cartService.removeItem(item)
  }

 itemsValue(): number {  //valor total da compra dos items
     return this.cartService.total()
  }

 clear(){
   this.cartService.clear()
  }

// Angular 4.0
//checkOrder(order: Order): Observable<string>{
//                  //quando o nosso objeto Order for enviando para o servidor
//                  //o mesmo produzira um id do tipo string para o objeto Order,
//                  //o qual representa a nossa compra.
//                  //Neste metodo utilizaremso POST uma vez que estamos inserindo
//                  //dados na nossa base de dado json.
//                  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
//                  //JSON.stringify(order) =>cria a representacao textual do objeto Order
//                  //headers.append("Content-Type","application/json") => indica o formato do dado
//                  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
//                  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
//                  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
//                  //que fora salvo no nosso arquivo db.json
//                  //Para resumir quando usamos um objeto Http no servico estamos de fato
//                  //configurando o nosso objeto Observable
//   const headers = new Headers()
```

```
//   headers.append("Content-Type","application/json")//nome do header, valor


//   return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
//                .map(response=> response.json())
//                .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
//                                //para retornar apenas o ID da compra
// }


 // Angular 4.3
//checkOrder(order: Order): Observable<string>{
  //quando o nosso objeto Order for enviando para o servidor
  //o mesmo produzira um id do tipo string para o objeto Order,
  //o qual representa a nossa compra.
  //Neste metodo utilizaremso POST uma vez que estamos inserindo
  //dados na nossa base de dado json.
  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
  //JSON.stringify(order) =>cria a representacao textual do objeto Order
  //headers.append("Content-Type","application/json") => indica o formato do dado
  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
  //que fora salvo no nosso arquivo db.json
  //Para resumir quando usamos um objeto Http no servico estamos de fato
  //configurando o nosso objeto Observable



  //const headers = new Headers()
  //headers.append("Content-Type","application/json")//nome do header, valor
  //return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
  //return this.http.post<Order>(`${MEAT_API}/orders`,order) //t<Order> este eh o tipo que iremos receber no post
  //.map(response=> response.json())
  //.map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                //para retornar apenas o ID da compra
//}



 // Angular 4.3
checkOrder(order: Order): Observable<string>{


  let headers = new HttpHeaders();


  if(this.loginService.isLoggedIn()){
    headers = headers.set('Authorization', `Bearer ${this.loginService.userLog.accessToken}`);
  }
```

```
    //quando o nosso objeto Order for enviando para o servidor
    //o mesmo produzira um id do tipo string para o objeto Order,
    //o qual representa a nossa compra.
    //Neste metodo utilizaremso POST uma vez que estamos inserindo
    //dados na nossa base de dado json.
    //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
    //JSON.stringify(order) =>cria a representacao textual do objeto Order
    //headers.append("Content-Type","application/json") => indica o formato do dado
    //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
    //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
    //que neste caso devido ao metodo POST sera apenas o id do objeto Order
    //que fora salvo no nosso arquivo db.json
    //Para resumir quando usamos um objeto Http no servico estamos de fato
    //configurando o nosso objeto Observable



    //const headers = new Headers()
    //headers.append("Content-Type","application/json")//nome do header, valor
    //return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
    return this.http.post<Order>(`${MEAT_API}/orders`,order, {headers:headers}) //t<Order> este eh o tipo que iremos receber no post
    //.map(response=> response.json())
    .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                        //para retornar apenas o ID da compra
  }

}
```

.../order/order.component.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';
import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';
import { Order, OrderItem } from './order.model';
import {Router} from '@angular/router';
import {FormGroup, FormBuilder} from '@angular/forms';
import {Validators} from '@angular/forms';
import {AbstractControl} from '@angular/forms';



@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
```

```
export class OrderComponent implements OnInit {

 emailPattern = /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i
 numberPattern = /^[0-9]*$/   //==> perceba que isto nao eh uma string mas uma sintaxe de expressa regular
  deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega

  paymentOptions : RadioOption[] = [
     {label: 'Dinheiro', value: 'MON'},
     {label: 'Cartão de Débito', value: 'DEB'},
     {label: 'Cartão Refeição', value: 'REF'}
  ]

  orderForm: FormGroup  //Veja no template order.component.html, o metodo hasError('chave') eh um metodo default
                        //da classe FormGroup




  constructor(private orderService : OrderService,
              private router : Router,
              private formBuilder: FormBuilder) { }

  ngOnInit() {
     this.orderForm = this.formBuilder.group({///Aqui, estamos passando nossos campos para o FormBuilder
                                        //Com esta configuração já podemos linkar nossos campos da
                                        //pagina com as configuracoes do orderForm.

       //name: this.formBuilder.control(''), //name: ''
       name: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
       email: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
       emailConfirmation: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
       address: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
       number: this.formBuilder.control('',[Validators.required,Validators.pattern(this.numberPattern)]),
       optionalAddress: this.formBuilder.control(''),
       paymentOption: this.formBuilder.control('',[Validators.required])
     }, {validator: OrderComponent.equalsTo}) //neste ponto estamos associando validadores ao nosso grupo (group)
  }

  //funcao
  //o retorno boolean sera utilizado no template para dar um retorno visual
  static equalsTo(group: AbstractControl) : {[key: string] : boolean} {
     const emailZ = group.get('email') //pegando a referencia do campo a partir do group
     const emailConfirmationZ = group.get('emailConfirmation')
```

```
    if(!emailZ || !emailConfirmationZ){ //caso um dos campos(atributos) nao existam no group
        return undefined
    }

    if(emailZ.value !== emailConfirmationZ.value){ //retornando uma chave(emailsNotMatch) com um valor
        return {emailsNotMatch: true}            //caso o valor seja true uma mensagem de erro sera exibida
    }                                   //Perceba que a chave 'emailsNotMatch' sera usada no template

    return undefined //caso o valor dos campos(atributos) sejam iguais, logo nao exibo mensagem de erro

  }



  cartItems() : CartItem[] {
    return this.orderService.cartItems()
  }

  increaseQtyOrder(item : CartItem){
    this.orderService.increaseQty(item)
  }

  decreaseQtyOrder(item : CartItem){
    this.orderService.decreaseQty(item)
  }

  removeOrder(item : CartItem){
    this.orderService.remove(item)
  }

  itemsValueOrder(): number { //valor total da compra dos items
    return this.orderService.itemsValue()
  }

//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                              //A classe Order possui alem dos dados do endereco contera tambem um array
//                              //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                              //e o "id >> atributo contido em MenuItem".

                            //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                            //apenas a "quantidade >> atributo contido em CartItem" e o
                            //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                            //em MenuItem ja se encontram em nossa base de dados json
```

```
//    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
// }

checkOrder(order: Order): void{ //metodo que finaliza a compra
                    //A classe Order possui alem dos dados do endereco contera tambem um array
                    //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                    //e o "id >> atributo contido em MenuItem".

                    //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                    //apenas a "quantidade >> atributo contido em CartItem" e o
                    //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                    //em MenuItem ja se encontram em nossa base de dados json

  order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))

  this.orderService.checkOrder(order)
            .subscribe( (orderId:string) => {
                    this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                    console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                    this.orderService.clear() //para limpar a compra depois de concluir
            })
  console.log(order)
 }
}
```

Observação:

Toda vez que for feito um reflash na pagina as informações do usuário serão perdidas uma vez que a nossa aplicação ela será reiniciada pelo browser, assim para manter a informação por um pouco mais de tempo teremos que guardar em um escopo maior. Há dois escopos que poderíamos trabalhar que é o local storage(escopo mais resistente poderíamos fechar o browser que a informação continuaria sendo mantida) ou session storage (semelhante a uma sessão web, uma vez que o browser for fechado a informação será perdida), porém não são seguros. No caso o nosso escopo de service, lembra que a informação do usuário foi guardada em LoginService, o qual tem um escopo bem menor, pois ao fazermos um simples reflash na página a informação será perdida. Tente acessar qualquer endereço da aplicação depois de fazer login diretamente pela barra de endereço, perceba que ao acessarmos um recurso perderemos a informação do usuário, a não ser que naveguemos pela aplicação neste caso a informação do usuário será mantida.

Seção 15

Vídeo 120

Protegendo o Modulo de Compra CanLoad

Nesta seção aprenderemos a apresentar a página de login de forma automática para o usuário, quando o mesmo tentar entrar no componente de compra sem estar autenticado e isto pode ser feito através de guardas de rotas.

.../security/loggedin.guard.ts

```
import { CanLoad, Route } from "@angular/router";
import { Injectable } from "@angular/core";
import { LoginService } from './login/login.service';

@Injectable()
export class LoggedInGuard implements CanLoad{

  constructor(private loginService: LoginService){//usamos o LoginService uma vez que este service eh
                                        //capaz de informar se o usuario esta autenticado

  }

  //metodo associada com as rotas com LoadChildren (carregamento tardio)
  // canLoad(route: Route): boolean { //metodo que eh uma implementacao a partir de uma interface
    //Esse método recebe um parâmetro que é exatamente um parâmetro que diz
    //qual a configuração da rota que foi associada com LoggedInGuard no arquivo de rotas app.routes.ts.
    //o metodo canLoad eh especialmente criado para decidir se o modulo sera
    //carregado tardiamente (true) ou nao (false) dependendo da condicao criada no metodo canLoad
   //    console.log(route);
   //    return false;
   //}

  // canLoad(route: Route) : boolean{
  //     console.log("Rota de guarda acionada. Veja o metodo canLoad() em loggedin.guard.ts");
  //     console.log(route);
  //     const loggedIn = this.loginService.isLoggedIn();
  //     if(!loggedIn){
  //       this.loginService.handleLogin(); //enviara o usuario para a pagina de autenticacao
  //     }
  //     return loggedIn;//o valor true permitira ir para o fechamento da compra ou seja
                //ir para o componente de compra
  // }

   canLoad(route: Route) : boolean{
     console.log("Rota de guarda acionada. Veja o metodo canLoad() em loggedin.guard.ts");
```

```
        console.log(route);
        const loggedIn = this.loginService.isLoggedIn();
        if(!loggedIn){
          this.loginService.handleLogin(`/${route.path}`); //enviara o usuario para a pagina de autenticacao
        }
        return loggedIn;//o valor true permitira ir para o fechamento da compra ou seja
                        //ir para o componente de compra
    }
}
```

.../shared/shared.module.ts

```
import {NgModule} from '@angular/core'

import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'

import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)


import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                        //permitira criar uma funcao capaz de retornar os
                                        //nosso servicos

import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';
import { SnackbarComponent } from './messages/snackbar/snackbar.component'; // componente de animacao criado por marques
import { NotificationService } from './messages/notification.service'

import { LoginService } from './../security/login/login.service';

import { LoggedInGuard } from 'app/security/loggedin.guard'

@NgModule({

  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent, SnackbarComponent],//componentes que serao utilizados neste modulo

  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo
```

```
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
               RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
            ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  CommonModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
              SnackbarComponent //componente de animacao criado por marques - fazendo o export desta diretiva nao precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
//Perceba que neste modulo nao teremos navegacao.

static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                              //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                              //app.module.ts
  return {
          ngModule : SharedModule,
          providers: [RestaurantsService,ShoppingCartService, OrderService,
          NotificationService, LoginService, LoggedInGuard] //declarando servicos na lista de providers
        }
  }



//FUNCAO ORIGINAL


//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                              //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                              //app.module.ts
//   return {
//           ngModule : SharedModule,
//           providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//         }
//  }


}
```

app/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
```

```
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
//import {OrderComponent} from './order/order.component'
//import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'
import { LoginComponent } from './security/login/login.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component'
import { LoggedInGuard } from 'app/security/loggedin.guard'

export const ROUTES: Routes = [

        {path: '', component: HomeComponent},
        {path: 'login/to', component: LoginComponent},
        {path: 'login', component: LoginComponent},
        //{path: 'about', component: AboutComponent},//esta rota nao sera mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule - rota responsavel pelo Lazy Loading
        {path: 'restaurants', component: RestaurantsComponent},
        //{path: 'order', component: OrderComponent},//esta rota nao sera mais usada uma vez que criamos o OrderModule
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                    {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a pagina menu
                            {path: 'menu', component: MenuComponent},          //como padrao quando for especificado
                            {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.
                ]},                                     //Veja: http://localhost:4200/restaurants/green-food
                                        //eh igual a http://localhost:4200/restaurants/green-food/menu
                                 //devido ao redirectTo.
        {path: 'order-summary', component: OrderSummaryComponent},

        ////rota para o Modulo OrderModule - rota responsavel pelo Lazy Loading
        //// {path: 'order', loadChildren: './order/order.module#OrderModule'}

        //associando uma rota com rota de guarda. Veja o arquivo loggedin.guard.ts
        {path: 'order', loadChildren: './order/order.module#OrderModule',
    canLoad: [LoggedInGuard]}//canLoad => pode receber um array de route guard
]
```

.../security/login/login.service.ts

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

```
import { Injectable } from '@angular/core';
import { MEAT_API } from '../../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado
import { Router } from '@angular/router';

@Injectable()
export class LoginService {

    //Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
    //que contem o name, login e o accessToken no nosso LoginService.

   userLog: User;

   constructor(private http:HttpClient,  //com HttpClient nao precisamos mais informar o contentType
            private router:Router) {  //no header da requisicao uma vez que o default eh application/JSON.
   }

   isLoggedIn(): boolean{
      return this.userLog !== undefined;
   }

   //  login(email: string, password: string): Observable<any>{
   //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
   //  }

   login(email: string, password: string): Observable<User>{
      return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                    .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                                  //de passar as informaçoes para o componente LoginComponent
   }

  // handleLogin() {
  //     this.router.navigate(['/login']);//au tentar acessar algo sem autenticado
  //                                  //voce sera direcionado para a tela de login
  // }

  handleLogin(path?: string) {
   this.router.navigate(['/login', path]);//voce sera direcionado para a tela de login
   }
    }
```

…/security/login/login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { LoginService } from './login.service';
import { NotificationService } from '../../../shared/messages/notification.service';

import { ActivatedRoute} from '@angular/router';//permite pegar parametros na rota
import { Router } from '@angular/router';



@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit { //trabalhando com Reactive Form

  loginForm: FormGroup;
  navigateTo: string

  constructor(private fb: FormBuilder,
              private loginService: LoginService,
              private notificationService: NotificationService,
              private activatedRoute: ActivatedRoute,
              private router: Router) { }

  ngOnInit() {
    this.loginForm = this.fb.group({
      email: this.fb.control('',[Validators.required, Validators.email]),
      password: this.fb.control('',[Validators.required])
    })
    //this.navigateTo = this.activatedRoute.snapshot.params['to']; //pegando o valor do parametro 'to'


                                                        //se chamar a pagina de login e nao tiver uma rota
    this.navigateTo = this.activatedRoute.snapshot.params['to'] || '/'; //seremos encaminhados para a raiz '/'
  }

  //login(){ //ate aqui temos apenas um Observable
  //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password);
  //}

  //login(){
  //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
  //                  .subscribe(user => console.log(user));
```

```
//}

//login(){//No metodo subscribe(...) podemos passar tres callbacks sao elas: 1 callback de sucesso -
        //2 callback de erro - 3 callback para quando o Observable terminar
//  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
//                .subscribe(user => this.notificationService.notify(`Bem vindo, ${user.name}`),
//                        response  => //reponse eh do tipo HttpErrorResponse que representa o erro da resposta
//                        this.notificationService.notify(response.error.message));//resp.status(403).json({message: 'Dados invalidos!'})
//}                                        //response.error ==> body da resposta
                                        //message ==> eh o atributo que contem a mensagem no body da resposta

login(){//No metodo subscribe(...) podemos passar tres callbacks sao elas: 1 callback para pegar a resposta -
        //2 callback de erro - 3 callback para quando o Observable terminar
    this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
                .subscribe(user => this.notificationService.notify(`Bem vindo, ${user.name}`),
                        response  => this.notificationService.notify(response.error.message),
                        ()=>{
                            this.router.navigate([this.navigateTo]);//enviando para rota desejada
                        }
                         );
            }
}
```

Seção 15

Vídeo 121

Protegendo o Componente de Compra com CanActivate

Na seção anterior foi criado um "route guard" com CanLoad pra determinar se o usuário poderia ou não carregar o módulo de compra baseado no fato de ele estar autenticado ou não, porém precisaremos de um outro "route guard" que é o CanActivate. Imagine que pode acontecer é a seguinte situação: o usuário pode se autenticar entrar no canload carregar, fazer o logoff e entrar novamente. Então ele poderia entrar novamente não autenticado já que a aplicaçao carregou aquela rota uma vez. Logo, para impedir esta situação utilizaremos além do CanLoad, usaremos o CanActivate.

.../security/loggedin.guard.ts

```
import { Injectable } from "@angular/core";
import { LoginService } from './login/login.service';
import { CanLoad, Route } from "@angular/router";
import { CanActivate } from "@angular/router";
```

```
import { ActivatedRouteSnapshot } from "@angular/router";
import { RouterStateSnapshot } from "@angular/router";

@Injectable()
export class LoggedInGuard implements CanLoad, CanActivate{

  constructor(private loginService: LoginService){//usamos o LoginService uma vez que este service eh
                                                   //capaz de informar se o usuario esta autenticado

  }

  //metodo associada com as rotas com LoadChildren (carregamento tardio)
  // canLoad(route: Route): boolean { //metodo que eh uma implementacao a partir de uma interface
     //Esse método recebe um parâmetro que é exatamente um parâmetro que diz
     //qual a configuração da rota que foi associada com LoggedInGuard no arquivo de rotas app.routes.ts.
     //o metodo canLoad eh especialmente criado para decidir se o modulo sera
     //carregado tardiamente (true) ou nao (false) dependendo da condicao criada no metodo canLoad
   //    console.log(route);
   //    return false;
   //}

  // canLoad(route: Route) : boolean{
  //     console.log("Rota de guarda acionada. Veja o metodo canLoad() em loggedin.guard.ts");
  //     console.log(route);
  //     const loggedIn = this.loginService.isLoggedIn();
  //     if(!loggedIn){
  //       this.loginService.handleLogin(); //enviara o usuario para a pagina de autenticacao
  //     }
  //     return loggedIn;//o valor true permitira ir para o fechamento da compra ou seja
                    //ir para o componente de compra
  // }

  //  canLoad(route: Route) : boolean{
  //     console.log("Rota de guarda acionada. Veja o metodo canLoad() em loggedin.guard.ts");
  //     console.log(route);
  //     const loggedIn = this.loginService.isLoggedIn();
  //     if(!loggedIn){
  //       this.loginService.handleLogin(`/${route.path}`); //enviara o usuario para a pagina de autenticacao
  //     }
  //     return loggedIn;//o valor true permitira ir para o fechamento da compra ou seja
                    //ir para o componente de compra
  // }

  checkAuthentication(path: string): boolean{
```

```
    console.log("Rota de guarda acionada. Veja o metodo checkAuthentication(...) em loggedin.guard.ts");
    const loggedIn = this.loginService.isLoggedIn();
    if(!loggedIn){
      this.loginService.handleLogin(`/${path}`); //enviara o usuario para a pagina de autenticacao
    }
    return loggedIn;//o valor true permitira ir para o fechamento da compra ou seja
                    //ir para o componente de compra


  }


  canLoad(route: Route) : boolean{ //aqui a rota ainda nao esta ativada e passa apenas a configuracao da rota
                                   //para voce decidir. Veja a implementacao do metodo checkAuthentication
    console.log("Rota de guarda acionada. Veja o metodo canLoad(...) em loggedin.guard.ts");
    return this.checkAuthentication(route.path);
  }


  //aqui voce tem todas as informacoes da rota que sera ativada
  canActivate(activatedRouteSnapshot: ActivatedRouteSnapshot, //representa uma rota da copia que fora ativada
              routerStateSnapshot: RouterStateSnapshot): boolean{//possui todas as rotas que foram ativadas
             console.log("Rota de guarda acionada. Veja o metodo canActivate(...) em loggedin.guard.ts");
             return this.checkAuthentication(activatedRouteSnapshot.routeConfig.path);
  }
}
```

.../app/app.routes.ts

```
import {Routes} from '@angular/router'
import {HomeComponent} from './home/home.component'
//import {AboutComponent} from './about/about.component'
import {RestaurantsComponent} from './restaurants/restaurants.component'
import {RestaurantDetailComponent} from './restaurant-detail/restaurant-detail.component'
import {MenuComponent} from './restaurant-detail/menu/menu.component'
import {ReviewsComponent} from './restaurant-detail/reviews/reviews.component'
//import {OrderComponent} from './order/order.component'
//import {OrderSummaryComponent } from './order-summary/order-summary.component'
import {AboutModule} from './about/about.module'
import { LoginComponent } from './security/login/login.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component'
import { LoggedInGuard } from 'app/security/loggedin.guard'



export const ROUTES: Routes = [


                {path: '', component: HomeComponent},
```

```
        {path: 'login', component: LoginComponent},
        {path: 'login/:to', component: LoginComponent},//com o parametro poderemo ir para pagina desejada depois da autenticacao
        //{path: 'about', component: AboutComponent},//esta rota nao sera mais usada uma vez que criamos o AboutModule
        {path: 'about', loadChildren: './about/about.module#AboutModule'},//rota para o Modulo AboutModule - rota responsavel pelo Lazy Loading
        {path: 'restaurants', component: RestaurantsComponent},
        //{path: 'order', component: OrderComponent},//esta rota nao sera mais usada uma vez que criamos o OrderModule
        {path: 'restaurants/:id', component: RestaurantDetailComponent,
                children: [  // rotas de navegacao filhas
                  {path: '', redirectTo: 'menu', pathMatch: 'full'}, //==> esta linha coloca a pagina menu
                          {path: 'menu', component: MenuComponent},         //como padrao quando for especificado
                          {path: 'reviews', component: ReviewsComponent}     //na url apenas o id do restaurante.
                ]},                                  //Veja: http://localhost:4200/restaurants/green-food
                                        //eh igual a http://localhost:4200/restaurants/green-food/menu
                           //devido ao redirectTo.
        {path: 'order-summary', component: OrderSummaryComponent},


        ////rota para o Modulo OrderModule - rota responsavel pelo Lazy Loading
        //// {path: 'order', loadChildren: './order/order.module#OrderModule'}


        //associando uma rota com rota de guarda. Veja o arquivo loggedin.guard.ts
        //a associacao de rota de guarda so faz sentido ser usado com loadChildren ou seja com carregamento tardio
        {path: 'order', loadChildren: './order/order.module#OrderModule',
  canLoad: [LoggedInGuard], canActivate: [LoggedInGuard]}//canLoad e canActivate=> podem receber um array de route guard
]
```

.../security/login/login.service.ts

//Reparando o encoding da url convertendo string(%2F) para base64(L29yZGVy)

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { MEAT_API } from './../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado
import { Router } from '@angular/router';

@Injectable()
export class LoginService {
```

```
  //Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
  //que contem o name, login e o accessToken no nosso LoginService.

  userLog: User;

  constructor(private http:HttpClient,  //com HttpClient nao precisamos mais informar o contentType
            private router:Router) {  //no header da requisicao uma vez que o default eh application/JSON.
  }

  isLoggedIn(): boolean{
     return this.userLog !== undefined;
  }

  //  login(email: string, password: string): Observable<any>{
  //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
  //  }

  login(email: string, password: string): Observable<User>{
     return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                    .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                                  //de passar as informaçoes para o componente LoginComponent
  }

 // handleLogin() {
 //     this.router.navigate(['/login']);//au tentar acessar algo sem autenticado
 //                                  //voce sera direcionado para a tela de login
 // }

 handleLogin(path?: string) {
  this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
                                       //btoa => funcao nativa do javascript a qual converte a string para base64
 }
}
```

.../security/login/login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
```

```
import { LoginService } from './login.service';
import { NotificationService } from './../../shared/messages/notification.service';

import { ActivatedRoute} from '@angular/router';//permite pegar parametros na rota
import { Router } from '@angular/router';



@Component({
  selector: 'mt-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit { //trabalhando com Reactive Form

  loginForm: FormGroup;
  navigateTo: string

  constructor(private fb: FormBuilder,
              private loginService: LoginService,
              private notificationService: NotificationService,
              private activatedRoute: ActivatedRoute,
              private router: Router) { }

  ngOnInit() {
    this.loginForm = this.fb.group({
      email: this.fb.control('',[Validators.required, Validators.email]),
      password: this.fb.control('',[Validators.required])
    })
    //this.navigateTo = this.activatedRoute.snapshot.params['to']; //pegando o valor do parametro 'to'


    //se chamar a pagina de login e nao tiver uma rota seremos encaminhados para a raiz '/'
    this.navigateTo = this.activatedRoute.snapshot.params['to'] || btoa('/');//btoa => funcao nativa do javascript a qual converte a string para base64


  }

  //login(){ //ate aqui temos apenas um Observable
  //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password);
  //}

  //login(){
  //  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
  //                  .subscribe(user => console.log(user));
  //}
```

```
//login(){//No metodo subscribe(...) podemos passar tres callbacks sao elas: 1 callback de sucesso -
        //2 callback de erro - 3 callback para quando o Observable terminar
//  this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
//                .subscribe(user => this.notificationService.notify(`Bem vindo, ${user.name}`),
//                        response  => //reponse eh do tipo HttpErrorResponse que representa o erro da resposta
//                        this.notificationService.notify(response.error.message));//resp.status(403).json({message: 'Dados invalidos!'})
//}                                        //response.error ==> body da resposta
                                        //message ==> eh o atributo que contem a mensagem no body da resposta


login(){//No metodo subscribe(...) podemos passar tres callbacks sao elas: 1 callback para pegar a resposta -
        //2 callback de erro - 3 callback para quando o Observable terminar
    this.loginService.login(this.loginForm.value.email, this.loginForm.value.password)
                .subscribe(user => this.notificationService.notify(`Bem vindo, ${user.name}`),
                        response  => this.notificationService.notify(response.error.message),
                        ()=>{
                            this.router.navigate([atob(this.navigateTo)]);//enviando para rota desejada
                            //atob => funcao nativa do javascript a qual converte de base64 para string
                        }


                    );
    }
}
```

Resultado do encoding da url convertendo string(%2F) para base64(L29yZGVy)

Seção 15

Vídeo 122

Disparando o Login Manualmente

Nesta seção criaremos uma interface que vai dar a possibilidade para o usuário autenticar-se na hora que ele quiser. Atualmente, a tela de login é apresentada quando o usuário chega a um ponto onde a aplicação onde a mesma detecta que o usuário precisa autenticar-se para ***fechar o pedido***, pois o mesmo não fez o login ainda. Isso ainda estará valendo, mas deixaremos a possibilidade do usuário entrar na interface ver a opção para autenticar-se, clicar no botão e ser apresentado na tela de login.

>***ng generate component user-detail - - spec=false***

./header/user-detail/user-detail.component.html

```
<div class="navbar-custom-menu">
  <ul class="nav navbar-nav">
    <li class="user user-menu">
      <span *ngIf="isLoggedIn(); else noLoggedIn"> <!-- uso do if else com ngIf-->
        Olá, {{userMetodo().name}} | <a (click)="logout()">Sair</a> <!-- | => eh um separador e nao um pipe-->
      </span>
      <ng-template #noLoggedIn> <!-- #noLoggedIn é uma template variable
                                    se isLoggedIn() for true  o span sera renderizado,
                                    porem se o mesmo for false  #noLoggedIn será true
                                    renderizando ng-template-->
        <a (click)="login()">Entrar</a>
      </ng-template>
    </li>
  </ul>
</div>
```

./header/user-detail/user-detail.component.css

```
.user-menu>span{ /**/
    position: relative;
    display: block;
    padding: 15px 15px;
    color: #fff;
  }

  .user-menu a{
```

```
  color: #fff;
  cursor: pointer
 }

 @media (max-width: 991px){
  .navbar-custom-menu {
     display: block;
     position: absolute;
     top: 0;
     right: 40px;
    }
 }
```

./header/user-detail/user-detail.component.ts

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

import { LoginService } from './../../security/login/login.service';
import { User } from './../../security/login/user.model';

@Component({
 selector: 'mt-user-detail',
 templateUrl: './user-detail.component.html',
 styleUrls: ['./user-detail.component.css']
})
export class UserDetailComponent implements OnInit {

 constructor(private loginService: LoginService) { }

 ngOnInit() {
 }

 userMetodo(): User {
  return this.loginService.userLog;
 }

 isLoggedIn(): boolean {
  return this.loginService.isLoggedIn();
 }

 login() {
  return this.loginService.handleLogin();
```

```
  }

  logout() {
    return this.loginService.logout();
  }
}
```

…/security/login/login.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { MEAT_API } from './../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado
import { Router } from '@angular/router';
import 'rxjs/add/operator/filter' //utilizaremos este operador para fazer uma filtragem dos dados
import { NavigationEnd } from '@angular/router';

@Injectable()
export class LoginService {

    //Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
    //que contem o name, login e o accessToken no nosso LoginService.

   userLog: User;
   lastUrl: string;

   constructor(private http:HttpClient,  //com HttpClient nao precisamos mais informar o contentType
            private router:Router) {  //no header da requisicao uma vez que o default eh application/JSON.
      //this.router.events //=> Observable para mudanças de navegacao

      //this.router.events.filter(ev => ev instanceof NavigationEnd) //ev => representa o nosso evento
      //                .subscribe((ev: NavigationEnd)=> console.log(ev.url));

      this.router.events.filter(ev => ev instanceof NavigationEnd)
                     .subscribe((ev: NavigationEnd)=> this.lastUrl = ev.url);//captura a ultima url acionada
   }

   isLoggedIn(): boolean{
      return this.userLog !== undefined;
   }

   //  login(email: string, password: string): Observable<any>{
```

```
  //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
  //  }

  login(email: string, password: string): Observable<User>{
     return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                    .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                                //de passar as informaçoes para o componente LoginComponent
  }

  // handleLogin() {
  //     this.router.navigate(['/login']);//au tentar acessar algo sem autenticado
  //                                //voce sera direcionado para a tela de login
  // }

  //handleLogin(path?: string) {
  //this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
  //                                    //btoa => funcao nativa do javascript a qual converte a string para base64
  //}

  handleLogin(path: string = this.lastUrl) {
   this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
                                    //btoa => funcao nativa do javascript a qual converte a string para base64
  }

  logout() {
   this.userLog = undefined;
  }



}
```

app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
//import { HttpModule } from '@angular/http';
import { HttpClientModule } from '@angular/common/http';
import { RouterModule } from '@angular/router';

import {ROUTES} from './app.routes'

import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
```

```
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                                //de um novo modulo OrderModule

import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                                //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                        //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                                //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                        //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router';
import { LoginComponent } from './security/login/login.component'//Import necessario para fazer o carregamento
                                        //em background dos modulos Lazy

//import {LocationStrategy, HashLocationStrategy} from '@angular/common'

import { BrowserAnimationsModule } from '@angular//platform-browser/animations';

import { UserDetailComponent } from './header/user-detail/user-detail.component'

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
```

```
  HomeComponent,
  //AboutComponent,
  RestaurantsComponent,
  RestaurantComponent,
  RestaurantDetailComponent,
  MenuComponent,
  ShoppingCartComponent,
  MenuItemComponent,
  ReviewsComponent,
  //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
  //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
  //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
  OrderSummaryComponent,
  LoginComponent,
  UserDetailComponent,
  //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
 ],
 imports: [
  BrowserModule,
  //HttpModule, //==> exemplo de Core Module. Veja o github do Angular //Angular 4.0
  HttpClientModule, //Angular 4.3
  //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
  RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
  //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                        //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
  //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular nao
                //sendo necessario portanto fazer imports novamente neste modulo.


  SharedModule.retornaNossosServicos(),//Com esta funcao temos a possibilidade de importar os serviços ou nao.Neste caso
                            //estamos importando. Veja o Modulo OrderModule onde nao importamos os
                            //serviços apenas os componentes declarados no SharedModule.
  //OrderModule => este modulo nao foi inserido aqui, pois nao queremos
  //que o mesmo seja carregado na inicializacao do projeto, porem faremos
  //este carregamento como lazy loading no app.routes.ts

  //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule

  BrowserAnimationsModule //modulo de animacao
 ],

 providers: [{provide: LOCALE_ID, useValue: 'pt-BR'}],
 //providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}], utilizado para evitar problemas de navegação ao
 //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url
 //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
```

```
  //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url


 //transferido para CoreModule

  //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo

 bootstrap: [AppComponent]

})

export class AppModule { }
```

app/header/header.component.html

```
<!-- Início do header -->

 <header class="main-header">

 <nav class="navbar navbar-static-top">

   <div class="container">

     <div class="navbar-header">

       <a  [routerLink]="['/']" class="navbar-brand"><b>Meat</b></a>

       <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar-collapse">

         <i class="fa fa-bars"></i>

       </button>

     </div>


     <!-- Collect the nav links, forms, and other content for toggling -->

     <div class="collapse navbar-collapse pull-left" id="navbar-collapse">

       <ul class="nav navbar-nav">

          <li routerLinkActive="active"><a [routerLink]="['/restaurants']">Restaurantes</a></li>

          <li routerLinkActive="active"><a [routerLink]="['/about']">Sobre</a></li>

       </ul>

     </div>

     <!-- /.navbar-collapse -->


     <mt-user-detail></mt-user-detail>


   </div>

 </nav>

 </header>

 <!-- Fim do header -->
```

..security/login/login.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { MEAT_API } from '../../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado
import { Router } from '@angular/router';
import 'rxjs/add/operator/filter' //utilizaremos este operador para fazer uma filtragem dos dados
import { NavigationEnd } from '@angular/router';

@Injectable()
export class LoginService {


    //Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
    //que contem o name, login e o accessToken no nosso LoginService.

   userLog: User;
   lastUrl: string;

   constructor(private http:HttpClient,  //com HttpClient nao precisamos mais informar o contentType
            private router:Router) {  //no header da requisicao uma vez que o default eh application/JSON.
      //this.router.events //=> Observable para mudanças de navegacao

      //this.router.events.filter(ev => ev instanceof NavigationEnd)
      //                 .subscribe((ev: NavigationEnd)=> console.log(ev.url));

      this.router.events.filter(ev => ev instanceof NavigationEnd)
                     .subscribe((ev: NavigationEnd)=> this.lastUrl = ev.url);
   }

   isLoggedIn(): boolean{
      return this.userLog !== undefined;
   }


   //  login(email: string, password: string): Observable<any>{
   //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
   //  }

   login(email: string, password: string): Observable<User>{
      return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                  .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                          //de passar as informaçoes para o componente LoginComponent
   }
```

```
  // handleLogin() {
  //     this.router.navigate(['/login']);//au tentar acessar algo sem autenticado
  //                                 //voce sera direcionado para a tela de login
  // }

  //handleLogin(path?: string) {
  //this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
  //                                    //btoa => funcao nativa do javascript a qual converte a string para base64
  //}

  handleLogin(path: string = this.lastUrl) {//<== se nao for passado parametro pegue a ultima rota em que esta
   this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
                                       //btoa => funcao nativa do javascript a qual converte a string para base64
  }

  logout() {
   this.userLog = undefined;
 }


}
```

Seção 15

Vídeo 123

Implementando Route Guard CanDeactivate na Compra

Nesta seção iremos implementar um outro tipo de route guard que é o CanDeactivate o qual pergunta ao usuário se realmente ele quer desistir de fazer a compra, caso opte por desistir todos os dados serão perdidos.

./order/leave-order.guard.ts

```
import { CanDeactivate, ActivatedRouteSnapshot, RouterStateSnapshot } from "@angular/router"
import { OrderComponent } from './order.component';
export class LeaveOrderGuard implements CanDeactivate<OrderComponent> {


   canDeactivate(orderComponent: OrderComponent,
                    activatedRouteSnapshot: ActivatedRouteSnapshot,
                    routerStateSnapshot: RouterStateSnapshot): boolean{


      if(!orderComponent.isOrderComplited()){//==> se nao tiver completada
        //se desejar sair as compras serao perdidas
        return window.confirm("Deseja desistir da compra?");//window.confirm  => retorna true ou false
      }else{
          return true; //prossiga
      }
   }
}
```

./order/leave-order.guard.ts

```
import { Component, OnInit } from '@angular/core';
import { RadioOption } from '../shared/radio/radio-option.model';
import { OrderService } from './order.service';
import { CartItem } from '../restaurant-detail/shopping-cart/cart-item.model';
import { Order, OrderItem } from './order.model';
import {Router} from '@angular/router';
import {FormGroup, FormBuilder} from '@angular/forms';
import {Validators} from '@angular/forms';
import {AbstractControl} from '@angular/forms';


@Component({
  selector: 'mt-order',
  templateUrl: './order.component.html'
})
export class OrderComponent implements OnInit {


  emailPattern = /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i
  numberPattern = /^[0-9]*$/   //==> perceba que isto nao eh uma string mas uma sintaxe de expressa regular
   deliveryTax: number = 8 //valor fixo para nosso delivery - taxa de entrega


    paymentOptions : RadioOption[] = [
    {label: 'Dinheiro', value: 'MON'},
    {label: 'Cartão de Débito', value: 'DEB'},
    {label: 'Cartão Refeição', value: 'REF'}
  ]


  orderForm: FormGroup  //Veja no template order.component.html, o metodo hasError('chave') eh um metodo default
                                //da classe FormGroup
 orderId: string;


  constructor(private orderService : OrderService,
    private router : Router,
            private formBuilder: FormBuilder) { }


            ngOnInit() {
              this.orderForm = this.formBuilder.group({//Aqui, estamos passando nossos campos para o FormBuilder
                //Com esta configuração já podemos linkar nossos campos da
                //pagina com as configuracoes do orderForm.


                //name: this.formBuilder.control(''), //name: ''
                 name: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
               email: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
               emailConfirmation: this.formBuilder.control('',[Validators.required,Validators.pattern(this.emailPattern)]),
               address: this.formBuilder.control('',[Validators.required,Validators.minLength(5)]),
               number: this.formBuilder.control('',[Validators.required,Validators.pattern(this.numberPattern)]),
              optionalAddress: this.formBuilder.control(''),
```

```
        paymentOption: this.formBuilder.control('',[Validators.required])
  }, {validator: OrderComponent.equalsTo}) //neste ponto estamos associando validadores ao nosso grupo (group)
}


//funcao
//o retorno boolean sera utilizado no template para dar um retorno visual
static equalsTo(group: AbstractControl) : {[key: string] : boolean} {
  const emailZ = group.get('email') //pegando a referencia do campo a partir do group
  const emailConfirmationZ = group.get('emailConfirmation')


  if(!emailZ || !emailConfirmationZ){ //caso um dos campos(atributos) nao existam no group
    return undefined
  }


  if(emailZ.value !== emailConfirmationZ.value){ //retornando uma chave(emailsNotMatch) com um valor
    return {emailsNotMatch: true}                //caso o valor seja true uma mensagem de erro sera exibida
  }                                              //Perceba que a chave 'emailsNotMatch' sera usada no template


     return undefined //caso o valor dos campos(atributos) sejam iguais, logo nao exibo mensagem de erro


}


cartItems() : CartItem[] {
      return this.orderService.cartItems()
}


increaseQtyOrder(item : CartItem){
  this.orderService.increaseQty(item)
}


decreaseQtyOrder(item : CartItem){
  this.orderService.decreaseQty(item)
}


removeOrder(item : CartItem){
  this.orderService.remove(item)
}


itemsValueOrder(): number { //valor total da compra dos items
  return this.orderService.itemsValue()
}


//   checkOrder(order: Order): void{//metodo que finaliza a compra
//                          //A classe Order possui alem dos dados do endereco contera tambem um array
//                          //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
//                          //e o "id >> atributo contido em MenuItem".
```

```
  //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                        //apenas a "quantidade >> atributo contido em CartItem" e o
                        //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                        //em MenuItem ja se encontram em nossa base de dados json


                        //    order.orderItems = this.cartItems().map((item:CartItem)=>new OrderItem(item.qtd, item.menuItem.id))
//      this.orderService.checkOrder(order)
//      console.log(order)
// }


checkOrder(order: Order): void{ //metodo que finaliza a compra
 //A classe Order possui alem dos dados do endereco contera tambem um array
 //de objetos do tipo OrderItem que possuira a "quantidade >> atributo contido em CartItem"
                        //e o "id >> atributo contido em MenuItem".


                        //Este metodo transforma um conjunto de CartItem em OrderItem o qual possuira
                        //apenas a "quantidade >> atributo contido em CartItem" e o
                        //"id >> atributo contido em MenuItem" uma vez que os demais dados contidos
                        //em MenuItem ja se encontram em nossa base de dados json


  order.orderItems = this.cartItems().map((item:CartItem) => new OrderItem(item.qtd, item.menuItem.id))


  this.orderService.checkOrder(order)
              .do( (orderId:string) => {
                this.orderId = orderId;//guardando o orderId para saber se a compra foi completada ou seja se
                                        //tem orderId entao a compra esta completa
              })
              .subscribe( (orderId:string) => {
                this.router.navigate(['/order-summary'])//regra de navegacao sem usar "routeLink"
                      console.log(`Compra concluída: ${orderId}`)//esta linha eh desnecessaria ==> "console"
                      this.orderService.clear() //para limpar a compra depois de concluir
              })
          console.log(order);
  }


  isOrderComplited() {
     return this.orderId !== undefined;
  }
}
```

./shared/shared.module.ts

```
import {NgModule} from '@angular/core'
import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
```

```
import {RatingComponent} from './rating/rating.component'
import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)



import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                //permitira criar uma funcao capaz de retornar os
                                //nosso servicos

import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';
import { SnackbarComponent } from './messages/snackbar/snackbar.component'; // componente de animacao criado por marques
import { NotificationService } from './messages/notification.service'
import { LoginService } from './../security/login/login.service';
import { LoggedInGuard } from 'app/security/loggedin.guard'
import { LeaveOrderGuard } from './../order/leave-order.guard';

@NgModule({
  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent, SnackbarComponent],//componentes que serao utilizados neste modulo
   imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo
  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                 RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                CommonModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                SnackbarComponent //componente de animacao criado por marques - fazendo o export desta diretiva nao precisarei importa-la em outro modulo
        ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado

//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
//Perceba que neste modulo nao teremos navegacao.

static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                              //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                              //app.module.ts
  return {
          ngModule : SharedModule,
          providers: [RestaurantsService,ShoppingCartService, OrderService,
          NotificationService, LoginService, LoggedInGuard, LeaveOrderGuard] //declarando servicos na lista de providers
        }
  }

//FUNCAO ORIGINAL
//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
```

```
                    //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                    //app.module.ts
//   return {
//          ngModule : SharedModule,
//          providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//        }
//  }
}
```

./order/order.module.ts

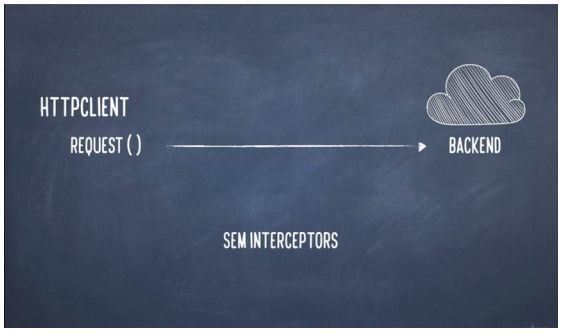
```
import {NgModule} from '@angular/core';
import {OrderComponent} from './order.component';
import {OrderItemsComponent} from './order-items/order-items.component';
import {DeliveryCostsComponent} from './delivery-costs/delivery-costs.component';
import {SharedModule} from '../shared/shared.module'; //estamos importando um modulo compartilhado criado por mim
import {RouterModule, Routes} from '@angular/router';
import { LeaveOrderGuard } from './leave-order.guard';

const ROUTES : Routes = [
            {path: '', component: OrderComponent,//caminho padrao que carregara o componente principal
                    canDeactivate: [LeaveOrderGuard]}
]

@NgModule({//define esta classe como um modulo
            declarations : [OrderComponent, OrderItemsComponent, DeliveryCostsComponent], //componentes usados neste modulo
            imports : [SharedModule, RouterModule.forChild(ROUTES)] //estamos importando um modulo compartilhado criado por mim
   //imports : [SharedModule,RouterModule.forChild(ROUTES)] => retorna o modulo ja com as rotas
   //RouterModule.forRoot(ROUTES) => utilizado apenas no modulo raiz
   //exports : [] //exportamos apenas os componentes que serao usados fora deste modulo
})
export class OrderModule{
  //Depois de aprontarmos o nosso modulo devemos configura-lo
  //como um lazy loading no arquivo app.routes.ts e no arquivo
  //app.module.ts teremos que remover as dependencias, componentes,
  //que ja sao carregadas em outros modulos.
}
```

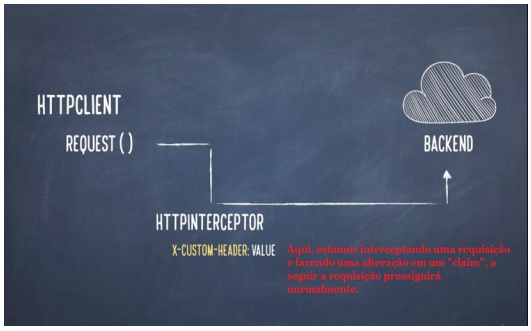
Seção 15

Vídeo 124

O que são Interceptors?

HttpInterceptors são uma grande adição feita pelo novo modolu HttpClient. Com esse mecanismo a aplicação consegue capturar o momento anterior ou posterior a uma chamada HTTP e fazer alterações comuns a todas as chamadas, como por exemplo atribuir um novo header a uma aquisição dinamicamente, sem precisar colocar em vários pontos da aplicação; o que é uma grande vantagem em termos de manutenção. A seguir as figuras abaixo ilustram uma requisição sem interceptor e outra com interceptor.





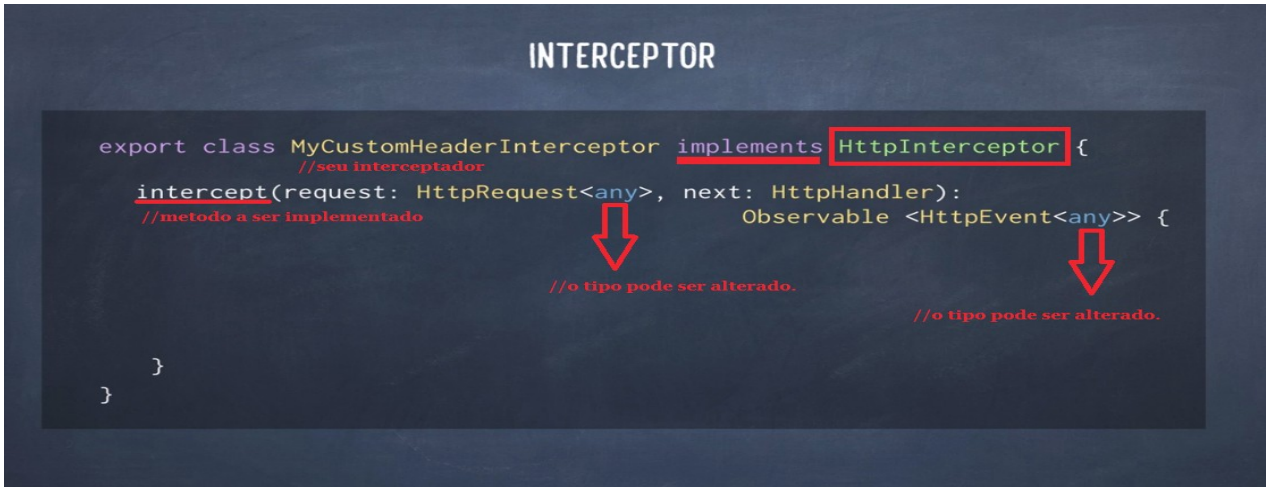
O que fazer com um Interceptors?

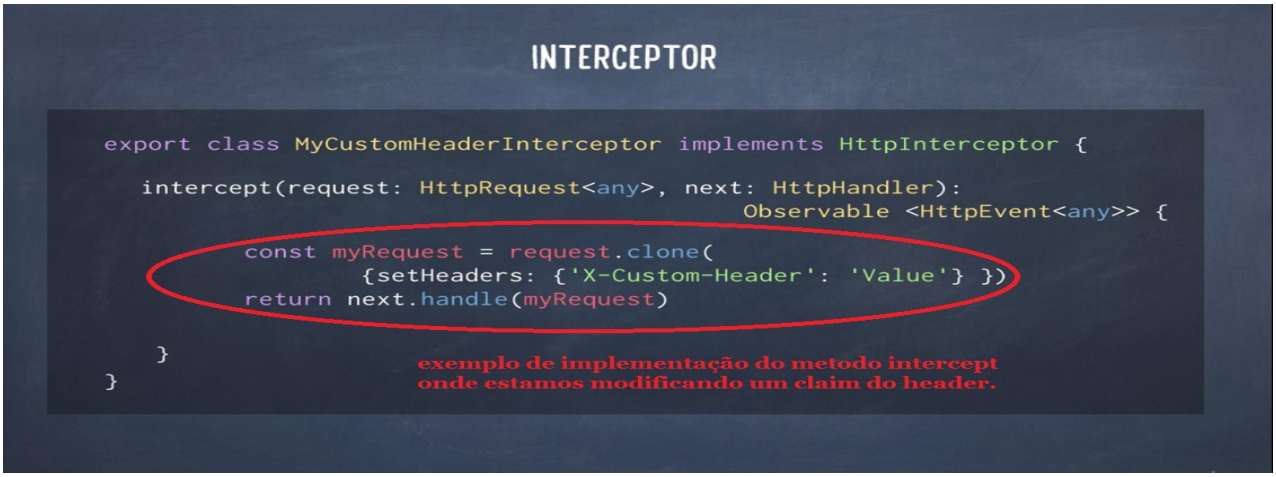
Interceptor é um padrão de projeto muito versátil. Com interceptor podemos fazer:

Logging: pode ser utilizado para logar quando a requisição está sendo feita e com o que, ou seja, com o que esta indo no request;

Profiling: pode ser utilizado para calcular o tempo em que um request esta levando e com isso melhorar os pontos da aplicação que precisam de otimização;

Error Handler: este uso é muito mais frequente onde se intercepta resposta da requisição e efetuar um tratamento global para a aplicação.





REGISTRANDO UM INTERCEPTOR NO MÓDULO PRINCIPAL

```
@NgModule({
  providers: [{provide:  HTTP_INTERCEPTORS,
               useClass: MyCustomInterceptor, multi: true}]
})
export class AppModule { }
```

REGISTRANDO MAIS DE UM INTERCEPTOR

```
@NgModule({
  providers: [{provide:  HTTP_INTERCEPTORS,
               useClass: MyCustomInterceptor, multi: true},
              {provide:  HTTP_INTERCEPTORS,
               useClass: MyAnotherInterceptor, multi: true}]
                //aqui, estamos declarando mais de um interceptor.
})
export class AppModule { }
```

INTERCEPTANDO A RESPOSTA

```
export class MyResponseInterceptor implements HttpInterceptor {

  intercept(request: HttpRequest<any>, next: HttpHandler):
                                        Observable <HttpEvent<any>> {

    return next.handle(request)
                .do( response => console.log(response.headers) )
                .catch( error => {
                     if (error.status === 401) { /* ... */ }
                     return Observable.throw(error)
                })

    }
  }
```

Registrando um Interceptor no Módulo Principal

```
@NgModule ({

    providers: [{provide: HTTP_INTERCEPTORS,
                 useClass: MyCustomInterceptor, multi: true}] //useClass => indica a classe do Intercptor
                                                              //multi: true => multi eh um atributo utilizado para indicar que essa
                                                              //classe fará parte da lista de interceptores ja registrados com o mesmo token
 })
export class AppModule {}
```

Seção 15

Vídeo 125

Utilizando HTTP Interceptors

security/auth.interceptor.ts

```
import { LoginService } from './login/login.service';


import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { Injectable, Injector } from '@angular/core';




@Injectable()
export class AuthInterceptor implements HttpInterceptor {




   constructor(private injector: Injector){
            //Injector => eh uma referencia para o mecanismo de injecao de dependencia do angular
            //com Injector posso pegar qualquer objeto e suas dependencias que estejam registradas
            //dentro do conteiner de injecao de dependencia, por exemplo pegar uma referencia para NotificationService
            //outro exemplo pegar uma referencia para LoginService.
    }


   //HttpRequest ==> objeto a ser interceptado e modificado se necessario
   //HttpHandler ==> objeto que representa o proximo interceptor na fila de de interceptors
   intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {


      console.log(">>>>>>>>> intercepting", request);//apenas para verificar se o interceptador esta funcionando
      const loginService = this.injector.get(LoginService);
      /* TRECHO REMOVIDO DE OrderService
      let headers = new HttpHeaders();
      if(this.loginService.isLoggedIn()){//setando o token de autorizacao
      headers = headers.set('Authorization', `Bearer ${this.loginService.userLog.accessToken}`);
```

```
    }
    */

    if(loginService.isLoggedIn()){//se esta autenticado. adicione o header
    //HttpRequest => eh um objeto imutavel, logo precisaremos cloná-lo para modificá-lo.
    //Aqui, estou clonando o request com todas as suas informacoes e adicionando uma outra
    //informacao que neste caso eh o Authorization no header do request.
    const authRequest = request.clone({setHeaders:{'Authorization' : `Bearer ${loginService.userLog.accessToken}`}})
    return next.handle(authRequest)

    }else{
    return next.handle(request)
    }

    //console.log(">>>>>>>>>>>>>>>>>> intercepting", request);
    //return next.handle(request) //neste exemplo estamos apenas prosseguindo com o request
                        //nao foi realizada nenhuma alteracao no request.
                        //Perceba que toda chamada http passara por aqui.
  }

}
```

shared/shared.module.ts

```
import {NgModule} from '@angular/core'

import {InputCustomizadoComponent} from './input-customizado/input-customizado.component'
import {RadioComponent} from './radio/radio.component'
import {RatingComponent} from './rating/rating.component'

import {FormsModule} from '@angular/forms'    //contem as diretivas NgForm - NgModel
import {ReactiveFormsModule} from '@angular/forms'  //contem as diretivas FormControlName - FormControlGroup
import {CommonModule} from '@angular/common' //contem as diretivas ngIf - ngFor / Estas diretivas sao importadas
                                //automaticamente no modulo raiz (class AppModule - arquivo app.module.ts)
```

```
import { ModuleWithProviders } from '@angular/core' //utilizacao do modulo ModuleWithProviders que nos
                                        //permitira criar uma funcao capaz de retornar os
                                        //nosso servicos


import { RestaurantsService } from '../restaurants/restaurants.service';
import { ShoppingCartService } from '../restaurant-detail/shopping-cart/shopping-cart.service';
import { OrderService } from '../order/order.service';
import { SnackbarComponent } from './messages/snackbar/snackbar.component'; // componente de animacao criado por marques
import { NotificationService } from './messages/notification.service'


import { LoginService } from './../security/login/login.service';


import { LoggedInGuard } from 'app/security/loggedin.guard'


import { LeaveOrderGuard } from './../order/leave-order.guard';


import { HTTP_INTERCEPTORS } from '@angular/common/http' //HTTP_INTERCEPTORS eh um token para o registro de intercptors
import { AuthInterceptor } from './../security/auth.interceptor'; //interceptor criado no arquivo auth.interceptor.ts


@NgModule({


  declarations:[InputCustomizadoComponent, RadioComponent, RatingComponent, SnackbarComponent],//componentes que serao utilizados neste modulo


  imports:[FormsModule, ReactiveFormsModule, CommonModule],//o que sera importado como dependencia deste modulo


  exports:[InputCustomizadoComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                    RadioComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                   RatingComponent,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                      FormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                ReactiveFormsModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                      CommonModule,//fazendo o export desta diretiva nao precisarei importa-la em outro modulo
                  SnackbarComponent //componente de animacao criado por marques - fazendo o export desta diretiva nao precisarei importa-la em outro modulo
          ] //informa quais sao os componentes que estao neste modulo que serao exportados para outros modulos


})
export class SharedModule  { // esta classe ira representar um modulo - modulo do tipo compartilhado



//Inserimos as diretivas FormsModule, ReactiveFormsModule e CommonModule na propriedade "exports" do modulo acima
//para nao ser necessario usarmos "import" nos modulos que irao utilizar o nosso SharedModule. Esta eh uma tentativa
//de enxugar os outros modulos.
```

```
//Perceba que neste modulo nao teremos navegacao.


static retornaNossosServicos() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                                //app.module.ts
  return {
          ngModule : SharedModule,


          providers: [RestaurantsService,ShoppingCartService, OrderService,
          NotificationService, LoginService, LoggedInGuard, LeaveOrderGuard,//declarando servicos na lista de providers
          {provide:HTTP_INTERCEPTORS, useClass:AuthInterceptor, multi:true}] //useClass => classe interceptor criada
        }
  }


//FUNCAO ORIGINAL
//static forRoot() : ModuleWithProviders {//Com esta funcao podemos importar ou nao os serviços alem dos componentes
                                //bastando para isso usar a funcao no import do modulo root. Veja o arquivo
                                //app.module.ts
//   return {
//           ngModule : SharedModule,
//           providers: [RestaurantsService,ShoppingCartService, OrderService] //declarando servicos na lista de providers
//         }
//  }


}
```

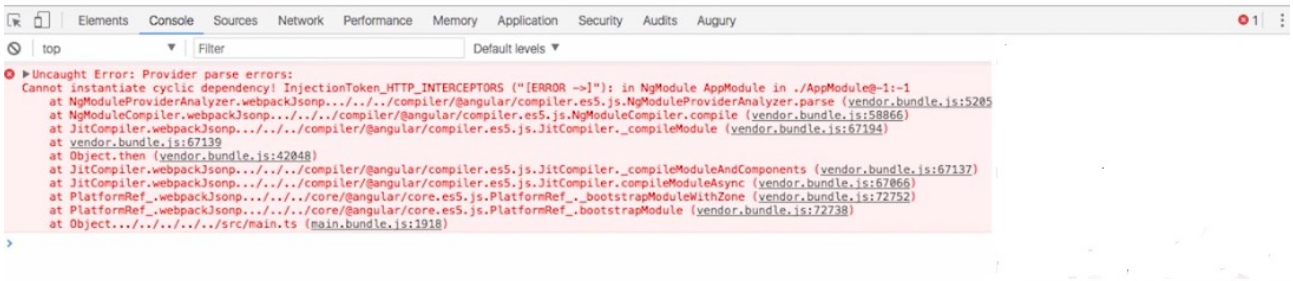
security/login/login.service.ts

```
import { HttpClient } from '@angular/common/http';
```

```
import { Observable } from 'rxjs/Observable';
import { Injectable } from '@angular/core';
import { MEAT_API } from './../../app.api';
import { User } from './user.model';
import 'rxjs/add/operator/do' //utilizaremos este operador para guardar uma referencia do objeto User que esta logado
import { Router } from '@angular/router';
import 'rxjs/add/operator/filter' //utilizaremos este operador para fazer uma filtragem dos dados
import { NavigationEnd } from '@angular/router';

@Injectable()
export class LoginService {


    //Com a utilizacao do operador "do" teremos a chance de guardar o objeto User
    //que contem o name, login e o accessToken no nosso LoginService.

   userLog: User;
   lastUrl: string;

   constructor(private http:HttpClient,  //com HttpClient nao precisamos mais informar o contentType
            private router:Router) {  //no header da requisicao uma vez que o default eh application/JSON.
      //this.router.events //=> Observable para mudanças de navegacao

      //this.router.events.filter(ev => ev instanceof NavigationEnd)
      //                  .subscribe((ev: NavigationEnd)=> console.log(ev.url));

      this.router.events.filter(ev => ev instanceof NavigationEnd)
                     .subscribe((ev: NavigationEnd)=> this.lastUrl = ev.url);
   }

   isLoggedIn(): boolean{
      return this.userLog !== undefined;
   }


   //  login(email: string, password: string): Observable<any>{
   //      return this.http.post(`${MEAT_API}/login`, {email: email, password: password});
   //  }

   login(email: string, password: string): Observable<User>{
      return this.http.post<User>(`${MEAT_API}/login`, {email: email, password: password})
                  .do(userX => this.userLog = userX);//este operador guardara o objeto User antes mesmo
                                          //de passar as informaçoes para o componente LoginComponent
   }

  // handleLogin() {
```

```
  //     this.router.navigate(['/login']);//au tentar acessar algo sem autenticado
  //                                //voce sera direcionado para a tela de login
  // }

  //handleLogin(path?: string) {
  //this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
  //                                //btoa => funcao nativa do javascript a qual converte a string para base64
  //}

  handleLogin(path: string = this.lastUrl) {//<== se nao for passado parametro pegue a ultima rota em que esta
   this.router.navigate(['/login', btoa(path)]);//voce sera direcionado para a tela de login
                                //btoa => funcao nativa do javascript a qual converte a string para base64
  }

  logout() {
   this.userLog = undefined;
  }
}
```

A utilização da classe Injector evitará erros do tipo abaixo:



order/order.service.ts

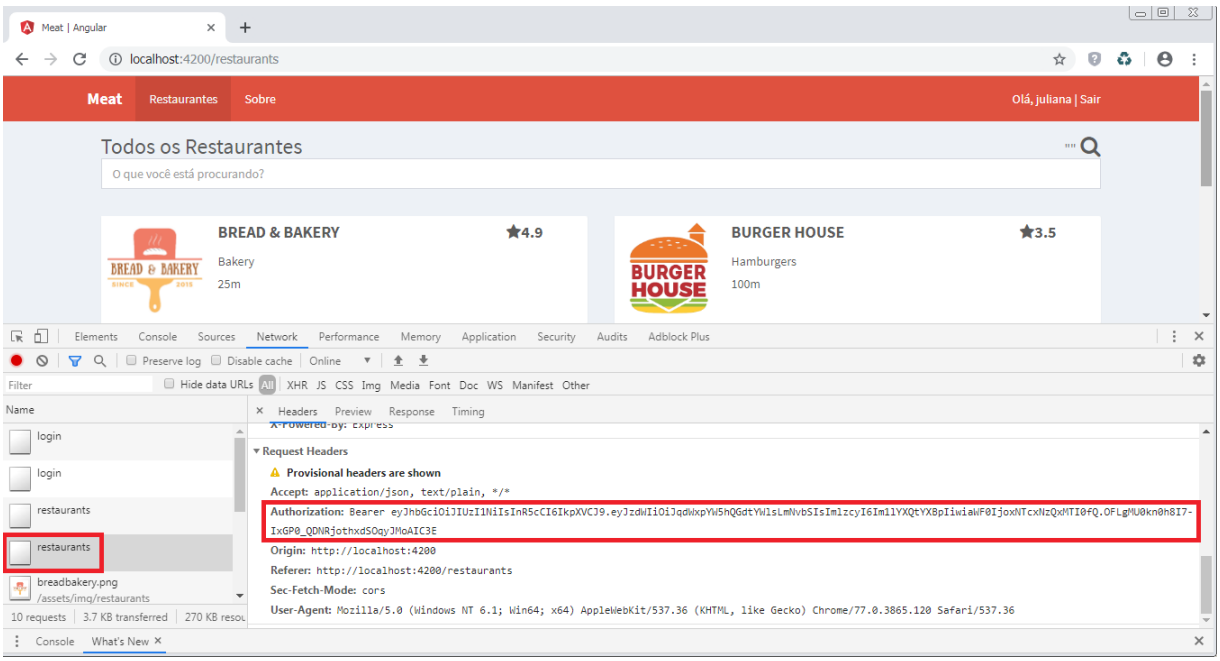
```
import {Injectable} from  '@angular/core';

import {ShoppingCartService} from '../restaurant-detail/shopping-cart/shopping-cart.service';

import {CartItem} from '../restaurant-detail/shopping-cart/cart-item.model';

import { Order, OrderItem } from './order.model';

//import { Http } from  '@angular/http'; // Angular 4.0

import { HttpClient, HttpHeaders } from  '@angular/common/http'; // Angular 4.3

import { Observable } from 'rxjs/Observable';
```

```
import 'rxjs/add/operator/map';

import {MEAT_API} from '../app.api';

//import { Headers, RequestOptions } from  '@angular/http'; // Angular 4.0

//import { Headers, RequestOptions } from  '@angular/common/http' // Angular 4.3 ja esta embutido no HttpClient

//import { LoginService } from './../security/login/login.service';//com LoginService pegaremos a referencia do User autenticado

@Injectable() //injetando o servico ShoppingCartService
export class OrderService {

 // constructor(private cartService: ShoppingCartService, private http: Http){
 //}


  constructor(private cartService: ShoppingCartService,
              private http: HttpClient){
  }

  cartItems() : CartItem[]{
    return this.cartService.items
  }

  increaseQty(item : CartItem){
    return this.cartService.increaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
  }

  decreaseQty(item : CartItem){
    return this.cartService.decreaseQty(item) //este metodo foi construido a partir de order.service.ts
                                  //para ShoppingCartService, ou seja, de tras para frente,
                                  //pois quem já esta trabalhando com CartItem eh o servico
                                  //ShoppingCartService
  }

  remove(item : CartItem){
     this.cartService.removeItem(item)
   }

   itemsValue(): number {  //valor total da compra dos items
      return this.cartService.total()
   }

   clear(){
     this.cartService.clear()
```

```
 }

// Angular 4.0
//checkOrder(order: Order): Observable<string>{
//                  //quando o nosso objeto Order for enviando para o servidor
//                  //o mesmo produzira um id do tipo string para o objeto Order,
//                  //o qual representa a nossa compra.
//                  //Neste metodo utilizaremso POST uma vez que estamos inserindo
//                  //dados na nossa base de dado json.
//                  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
//                  //JSON.stringify(order) =>cria a representacao textual do objeto Order
//                  //headers.append("Content-Type","application/json") => indica o formato do dado
//                  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
//                  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
//                  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
//                  //que fora salvo no nosso arquivo db.json
//                  //Para resumir quando usamos um objeto Http no servico estamos de fato
//                  //configurando o nosso objeto Observable
//   const headers = new Headers()
//   headers.append("Content-Type","application/json")//nome do header, valor

//   return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
//                .map(response=> response.json())
//                .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
//                                  //para retornar apenas o ID da compra
// }

 // Angular 4.3
//checkOrder(order: Order): Observable<string>{
  //quando o nosso objeto Order for enviando para o servidor
  //o mesmo produzira um id do tipo string para o objeto Order,
  //o qual representa a nossa compra.
  //Neste metodo utilizaremso POST uma vez que estamos inserindo
  //dados na nossa base de dado json.
  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
  //JSON.stringify(order) =>cria a representacao textual do objeto Order
  //headers.append("Content-Type","application/json") => indica o formato do dado
  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
  //que fora salvo no nosso arquivo db.json
  //Para resumir quando usamos um objeto Http no servico estamos de fato
  //configurando o nosso objeto Observable



  //const headers = new Headers()
  //headers.append("Content-Type","application/json")//nome do header, valor
  //return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
```

```
  //return this.http.post<Order>(`${MEAT_API}/orders`,order) //t<Order> este eh o tipo que iremos receber no post
  //.map(response=> response.json())
  //.map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                        //para retornar apenas o ID da compra
//}



 // Angular 4.3
checkOrder(order: Order): Observable<string>{

 // TRECHO LEVADO PARA O INTERCEPTADOR
 // let headers = new HttpHeaders();
 // if(this.loginService.isLoggedIn()){//setando o token de autorizacao
 //   headers = headers.set('Authorization', `Bearer ${this.loginService.userLog.accessToken}`);
 // }

  //quando o nosso objeto Order for enviando para o servidor
  //o mesmo produzira um id do tipo string para o objeto Order,
  //o qual representa a nossa compra.
  //Neste metodo utilizaremso POST uma vez que estamos inserindo
  //dados na nossa base de dado json.
  //`${MEAT_API}/orders` => local para onde nossos dados serao salvos
  //JSON.stringify(order) =>cria a representacao textual do objeto Order
  //headers.append("Content-Type","application/json") => indica o formato do dado
  //new RequestOptions({headers: headers}) => objeto que informa o nossos headers
  //.map(response=> response.json()) => pegando apenas o corpo do objeto Response
  //que neste caso devido ao metodo POST sera apenas o id do objeto Order
  //que fora salvo no nosso arquivo db.json
  //Para resumir quando usamos um objeto Http no servico estamos de fato
  //configurando o nosso objeto Observable



  //const headers = new Headers()
  //headers.append("Content-Type","application/json")//nome do header, valor
  //return this.http.post(`${MEAT_API}/orders`, JSON.stringify(order), new RequestOptions({headers: headers}))
  //return this.http.post<Order>(`${MEAT_API}/orders`,order, {headers:headers}) //t<Order> este eh o tipo que iremos receber no post
  return this.http.post<Order>(`${MEAT_API}/orders`,order)
  //.map(response=> response.json())
  .map(order=> order.id); //fazendo uma transformacao no corpo do objeto response
                        //para retornar apenas o ID da compra
 }



}
```

A partir de agora o nosso token de acesso "Authorization" será enviado junto com nosso request:



Seção 16

Vídeo 126

Criando um Error Handler Global

Continuando a nossa série de melhorias está na hora da gente mexer no Error Handler. Tínhamos feito uma primeira modificação que era só para adaptar para o novo modelo HTTP que é o HttpClient, mas a gente não estava mais usando na aplicação de forma alguma. Chegou a hora da gente fazer um Error Handler melhor, ou seja, um Error Handler Global. Um tratador de erros que consiga lidar com alguns erros de forma padrão. Nossa aplicação já checa pela segurança em alguns pontos; então a gente já tem que lidar com os erros 401 ou 403 e podemos adicionar outros como 404, 422 e o erro 500 e dar uma notificação ao usuário utilizando o nosso NotificationService.

…/app-error-handler.ts

```
import { NotificationService } from './shared/messages/notification.service';

import {HttpErrorResponse} from '@angular/common/http' // Angular 4.3

import 'rxjs/add/observable/throw'

import { ErrorHandler } from '@angular/core'

import { Injectable } from '@angular/core'

//import { LoginService } from './security/login/login.service';

import {Injector } from '@angular/core'

import { LoginService } from './security/login/login.service';
```

```
//ULTIMA ALTERACAO - ERROR HANDLER GLOBAL
@Injectable() //==> todo provider deve ter a anotacao @Injectable
export class AplicationErrorHandler extends ErrorHandler { //criando uma classe de erro global
//Esta classe sera o ultimo ponto de tratamento de excecao dentro de nosso aplicacao.
//O Angular ja possui um tratador de erro padrao, mas se nos nao tratarmos o erro basicamente
//o erro sera impresso no console.

//Para configurarmos o nosso AplicationErrorHandler como o nosso tratador de erro padrao
//teremos que inseri-lo na lista de providers do app.module.

        constructor(private notificationService: NotificationService,
                private injector: Injector){//private loginService: LoginService){//==> provoca erro de dependencia ciclica
            super();//chamando o super de ErrorHandler
        }

        handleError(errorResponse : HttpErrorResponse | any){ //metodo sobrescito

        console.log(">>>======== AplicationErrorHandler ==========> " + errorResponse);//imprime no console

        if(errorResponse instanceof HttpErrorResponse){
            const message = errorResponse.error.message;
            switch(errorResponse.status){
                case 401:
                    this.injector.get(LoginService).handleLogin()//nao utilizamos o LoginService devido ao erro de dependencia ciclica
                  break;
                case 403:
                    this.notificationService.notify(message || 'Não autorizado!')
                  break;
                case 404:
                    this.notificationService
                    .notify(message || 'Recurso não encontrado. Verifique o console para mais detalhes!')
                  break;
                //case 500: //Poderíamos adicionar outro tipos de erro como: 422, 500 entre outros.
                    //        this.notificationService.notify(message || 'XXXXXX!');
                    //        break;
                }
          }
         super.handleError(errorResponse);//imprime a resposta
}

//// SEGUNDA MODIFICACAO
//export class ErrorHandler { //criando uma classe de erro
//
//static handleError(error : HttpErrorResponse | any){
//
//      let errorMessage: string
//
//      if(error instanceof HttpErrorResponse){
//          const body = error.error
//          errorMessage = `${error.url} : ${error.status} - ${error.statusText || ''} ${body}`
//      }else{
//          errorMessage = error.message ? error.message : error.toString()
```

```
//                  }
//
//                  console.log(">>>==================> " + errorMessage)
//
//                  return Observable.throw(errorMessage);
//
//                  }


//// PRIMEIRA  MODIFICACAO
// Angular 4.0

//import {Response} from '@angular/http' // Angular 4.0

//import {Observable} from 'rxjs/Observable'

//import 'rxjs/add/observable/throw'

//export class ErrorHandler { //criando uma classe de erro

//static handleError(error : Response | any){ // Angular 4.0

//                  let errorMessage: string

//                  if(error instanceof Response){
//                                  errorMessage = `Erro [ ${error.status} ] ao acessar a URL [ ${error.url} ] - [ ${error.statusText} ]`
//                  }else{
//                                  errorMessage = error.toString()
//                  }

//                  console.log(">>>==================> " + errorMessage)

//                  return Observable.throw(errorMessage);



//                  }


}
```

…/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule, LOCALE_ID } from '@angular/core';
//import { HttpModule } from '@angular/http';
import { HttpClientModule } from '@angular/common/http';
import { RouterModule } from '@angular/router';
import {ROUTES} from './app.routes'
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { HomeComponent } from './home/home.component';
```

```
//import {AboutComponent} from './about/about.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo AboutModule
import { RestaurantsComponent } from './restaurants/restaurants.component';
import { RestaurantComponent } from './restaurants/restaurant/restaurant.component'

//import { RestaurantsService } from './restaurants/restaurants.service';//transferido para CoreModule
import { RestaurantDetailComponent } from './restaurant-detail/restaurant-detail.component';
import { MenuComponent } from './restaurant-detail/menu/menu.component';
import { ShoppingCartComponent } from './restaurant-detail/shopping-cart/shopping-cart.component';
import { MenuItemComponent } from './restaurant-detail/menu-item/menu-item.component';
import { ReviewsComponent } from './restaurant-detail/reviews/reviews.component';
//import { ShoppingCartService } from './restaurant-detail/shopping-cart/shopping-cart.service';//transferido para CoreModule
//import { OrderComponent } from './order/order.component';//comentado devido ao componente fazer parte
                                        //de um novo modulo OrderModule

//import { FormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Template Forms
//import { InputCustomizadoComponent } from './shared/input-customizado/input-customizado.component';//agora teremos um modulo compartilhado, SharedModule,
                                                                        //com este componente
//import { RadioComponent } from './shared/radio/radio.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente

//import { OrderItemsComponent } from './order/order-items/order-items.component';//comentado devido ao componente fazer parte
                                                        //de um novo modulo OrderModule

//import { OrderService } from './order/order.service';//transferido para CoreModule

//import { DeliveryCostsComponent } from './order/delivery-costs/delivery-costs.component';
import { OrderSummaryComponent } from './order-summary/order-summary.component';
//import { RatingComponent } from './shared/rating/rating.component';//agora teremos um modulo compartilhado, SharedModule,
                                                //com este componente
import { ReactiveFormsModule } from '@angular/forms';  //utilizado na validacao de formularios com Reactive Forms

import { SharedModule } from './shared/shared.module' //modulo customizado por mim

//import { CoreModule } from './core/core.module'//==> importando um modulo core, mas foi comentado
                                        //uma vez que mudamos os providers para o SharedModule

import { PreloadAllModules } from '@angular/router';
import { LoginComponent } from './security/login/login.component'//Import necessario para fazer o carregamento
                                        //em background dos modulos Lazy

//import {LocationStrategy, HashLocationStrategy} from '@angular/common'

import { BrowserAnimationsModule } from '@angular//platform-browser/animations';

import { UserDetailComponent } from './header/user-detail/user-detail.component'

import { AplicationErrorHandler } from './app-error-handler';

import { ErrorHandler } from '@angular/core';

@NgModule({
```

```
declarations: [
  AppComponent,
  HeaderComponent,
  HomeComponent,
  //AboutComponent,
  RestaurantsComponent,
  RestaurantComponent,
  RestaurantDetailComponent,
  MenuComponent,
  ShoppingCartComponent,
  MenuItemComponent,
  ReviewsComponent,
  //OrderComponent,//agora teremos um modulo OrderModule que contem este componente
  //InputCustomizadoComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //RadioComponent,//agora teremos um modulo compartilhado, SharedModule, com este componente.
  //OrderItemsComponent,//agora teremos um modulo OrderModule que contem este componente
  //DeliveryCostsComponent,//agora teremos um modulo OrderModule que contem este componente
  OrderSummaryComponent,
  LoginComponent,
  UserDetailComponent,
  //RatingComponent //agora teremos um modulo compartilhado, SharedModule, com este componente.
],
imports: [
  BrowserModule,
  //HttpModule, //==> exemplo de Core Module. Veja o github do Angular //Angular 4.0
  HttpClientModule, //Angular 4.3
  //FormsModule, //utilizado na validacao de formularios com Template Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //ReactiveFormsModule, //utilizado na validacao de formularios com Reactive Forms //agora teremos um modulo compartilhado, SharedModule, com esta diretiva.
  //AboutModule =>caso quisessemos que nosso "Modulo AboutModule" fosse carregado na inicializacao
  RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),//carregara os modulos Lazy em bakground
  //RouterModule.forRoot(ROUTES), //importa as rotas atraves do metodo forRoot em outros modulos usaremos
                          //RouterModule.forChild(ROUTES). Veja o arquivo about.module.ts
  //SharedModule, //este modulo customizado contem componentes criado por mim alem de exportar diretivas do angular nao
                //sendo necessario portanto fazer imports novamente neste modulo.


  SharedModule.retornaNossosServicos(),//Com esta funcao temos a possibilidade de importar os serviços ou nao.Neste caso
                              //estamos importando. Veja o Modulo OrderModule onde nao importamos os
                              //serviços apenas os componentes declarados no SharedModule.
  //OrderModule => este modulo nao foi inserido aqui, pois nao queremos
  //que o mesmo seja carregado na inicializacao do projeto, porem faremos
  //este carregamento como lazy loading no app.routes.ts

  //CoreModule //importando um modulo core, mas foi comentado uma vez que mudamos os providers para o SharedModule

  BrowserAnimationsModule //modulo de animacao
],


 providers: [
   {provide: LOCALE_ID, useValue: 'pt-BR'},
   {provide: ErrorHandler, useClass: AplicationErrorHandler}//registrando o nosso provider errorhandler global
],
```

```
  //providers: [{provide: LocationStrategy, useClass: HashLocationStrategy},{provide: LOCALE_ID, useValue: 'pt-BR'}], utilizado para evitar problemas de navegação ao
  //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url
  //{provide: LocationStrategy, useClass: HashLocationStrategy} => utilizado para evitar problemas de navegação ao
  //se tentar fazer bookmark com nossas paginas usando a tecnica do Hash na url


 //transferido para CoreModule
 //providers: [RestaurantsService,ShoppingCartService, OrderService, {provide: LOCALE_ID, useValue: 'pt-BR'}], //declarando um servico na lista de providers com escopo para este modulo
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Seção 16

Vídeo 127

Utilizando Zonas

Resolvendo o problema das mensagens de notificação para a classe AplicationErrorHandler onde estes erros, 401, 402, 403, 404 entre outros, não desaparecerem depois de um tempo após serem exibidos ao usuário.

.../app-error-handler.ts

```
import { NotificationService } from './shared/messages/notification.service';
import {HttpErrorResponse} from '@angular/common/http'; // Angular 4.3
import 'rxjs/add/observable/throw';
import { ErrorHandler } from '@angular/core';
import { Injectable } from '@angular/core';
//import { LoginService } from './security/login/login.service';
import {Injector } from '@angular/core';
import { LoginService } from './security/login/login.service';
import { NgZone } from '@angular/core'

//ULTIMA ALTERACAO - ERROR HANDLER GLOBAL
@Injectable() //==> todo provider deve ter a anotacao @Injectable
export class AplicationErrorHandler extends ErrorHandler { //criando uma classe de erro global
//Esta classe sera o ultimo ponto de tratamento de excecao dentro de nosso aplicacao.
//O Angular ja possui um tratador de erro padrao, mas se nos nao tratarmos o erro basicamente
//o erro sera impresso no console.

//Para configurarmos o nosso AplicationErrorHandler como o nosso tratador de erro padrao
//teremos que inseri-lo na lista de providers do app.module.

        constructor(private notificationService: NotificationService,
                private injector: Injector,//private loginService: LoginService){//==> provoca erro de dependencia ciclica
                private zone: NgZone){
                        super();//chamando o super de ErrorHandler
        }
```

```
handleError(errorResponse : HttpErrorResponse | any){ //metodo sobrescito

console.log(">>>======== AplicationErrorHandler ==========> " + errorResponse);//imprime no console

if(errorResponse instanceof HttpErrorResponse){
        const message = errorResponse.error.message;

        this.zone.run(() => { //this.zone.run => metodo que resolve o problema das mensagens nao desaparcerem
                //apos as mesmas serem exibidas ao usuario apos um tempo.
            switch(errorResponse.status){
                case 401:
                    this.injector.get(LoginService).handleLogin()//nao utilizamos o LoginService devido ao erro de dependencia ciclica
                    break;
                case 403:
                     this.notificationService.notify(message || 'Não autorizado!')
                    break;
                case 404:
                     this.notificationService
                     .notify(message || 'Recurso não encontrado. Verifique o console para mais detalhes!')
                    break;
                //case 500: //Poderíamos adicionar outro tipos de erro como: 422, 500 entre outros.
                     //          this.notificationService.notify(message || 'XXXXXX!');
                     //          break;
            }
        })

        }
  super.handleError(errorResponse);//imprime a resposta
}

//// SEGUNDA MODIFICACAO
//export class ErrorHandler { //criando uma classe de erro
//
//static handleError(error : HttpErrorResponse | any){
//
//      let errorMessage: string
//
//      if(error instanceof HttpErrorResponse){
//              const body = error.error
//              errorMessage = `${error.url} : ${error.status} - ${error.statusText || ''} ${body}`
//      }else{
//              errorMessage = error.message ? error.message : error.toString()
//      }
//
//      console.log(">>>==================> " + errorMessage)
//
//      return Observable.throw(errorMessage);
//
//      }
```

```
//// PRIMEIRA MODIFICACAO
// Angular 4.0

//import {Response} from '@angular/http' // Angular 4.0

//import {Observable} from 'rxjs/Observable'

//import 'rxjs/add/observable/throw'

//export class ErrorHandler { //criando uma classe de erro

//static handleError(error : Response | any){ // Angular 4.0

//		let errorMessage: string

//		if(error instanceof Response){
//			errorMessage = `Erro [ ${error.status} ] ao acessar a URL [ ${error.url} ] - [ ${error.statusText} ]`
//		}else{
//			errorMessage = error.toString()
//		}
//		console.log(">>>==================> " + errorMessage)
//		return Observable.throw(errorMessage);
//		}

}
```

Seção 17

Vídeo 128

Atualização das Dependências da Aplicação