

DESENVOLVIMENTO MOBILE

IONIC

edson.sensato@espm.br



Introdução

Desenvolvimento pode ser feito nos frameworks Javascript:

Angular

React

Vue



Instalação NodeJS

NodeJS:

<https://nodejs.org/en/download/>

Verificar a versão (se) instalada:

`node -v`

Para atualizar (instalar o módulo **n**):

`npm cache clean -f`

`npm install -g n`

`n stable`

Instalação Ionic

Referência:

<https://ionicframework.com/docs/angular/your-first-app>



Criar Aplicação

Existem templates que podem ser utilizados para criar uma aplicação inicial:

```
ionic start --list
```

Criar uma nova aplicação com o template (starter) blank:

```
ionic start contador blank
```

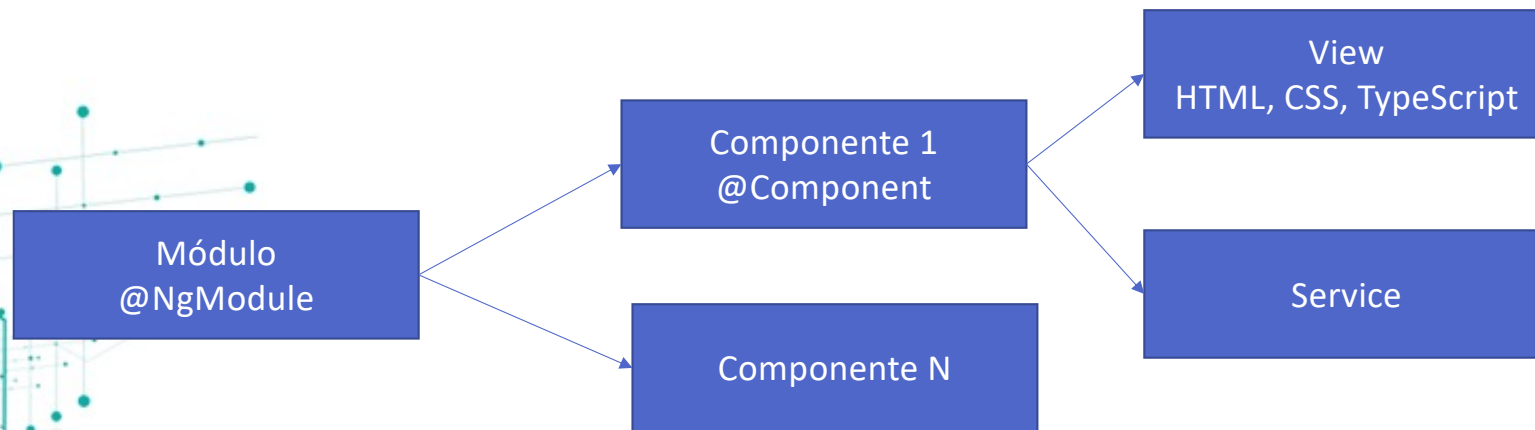
Para executar a aplicação deve-se estar no diretório que foi criado (no exemplo, contador) e executar:

```
ionic serve --lab
```

Obs: pode ser solicitada a instalação da extensão lab. Basta confirmar (Y)

Introdução ao Angular

Plataforma e framework para construção de aplicações cliente single-page utilizando HTML e TypeScript



Referência: <https://angular.io/>

Configuração Módulo

Cada módulo possui os seguintes atributos básicos:

declarations → Componentes que fazem parte do módulo

imports → Módulos externos que são importados para o projeto

exports → Partes do módulo que podem ser utilizados em outros projetos

providers → Serviços utilizados pelo projeto

bootstrap → Root Component

Configuração Componente

Cada componente possui os seguintes atributos básicos:

`selector` → Identificação do componente que será utilizado para incluir o componente na página HTML por meio de uma tag correspondente ao definido neste atributo

`templateUrl` → Indica qual será o template, isto é a página HTML que irá representar a camada visual do componente

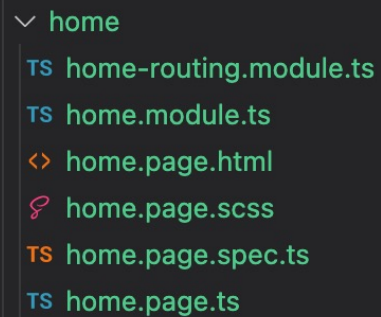
`styleUtl` → Indica a folha de estilo associada (CSS)

`providers` → Serviços utilizados pelo componente

Exemplo Estrutura Básica Componente

Aplicação composta por diversos componentes

Cada componente possui uma pasta com seu nome e seus arquivos:



```
▼ home
  TS home-routing.module.ts
  TS home.module.ts
  <> home.page.html
  home.page.scss
  TS home.page.spec.ts
  TS home.page.ts
```

home.page.html → camada visual do componente

home.page.scss → folha de estilo (tema)

home.page.ts → lógica de negócio do componente

Para gerar um componente:

ionic generate component contador

Referência: <https://ionicframework.com/docs/cli/commands/generate>

Estrutura Componente

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

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

  constructor() { }

  ngOnInit() {}
}
```

```
▼ src
  ▼ app
    ▼ contador
      <> contador.component.html
      📄 contador.component.scss
      TS contador.component.spec.ts
      TS contador.component.ts
```

selector → identifica o componente

Declarando o Componente

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';
import { FormsModule } from '@angular/forms';
import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';
import { ContadorComponent } from '../contador/contador.component';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage, ContadorComponent],
  exports: [ContadorComponent]
})
export class HomePageModule {}
```

```
▼ src
  ▼ app
    ▼ contador
      <> contador.component.html
      🔗 contador.component.scss
      TS contador.component.spec.ts
      TS contador.component.ts
    ▼ home
      TS home-routing.module.ts
      TS home.module.ts
```

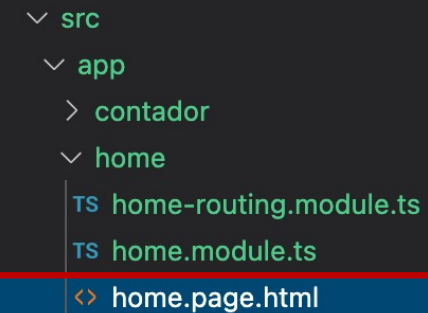
Usando o Componente

```
<ion-header [translucent]="true">
<ion-toolbar>
<ion-title>Alô Ionic!</ion-title>
</ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
<ion-header collapse="condense">
<ion-toolbar>
<ion-title size="large"> Alô Ionic!</ </ion-title>
</ion-toolbar>
</ion-header>

<app-contador></app-contador>

</ion-content>
```



```
src
├── app
│   ├── contador
│   └── home
│       ├── home-routing.module.ts
│       ├── home.module.ts
│       └── home.page.html
```

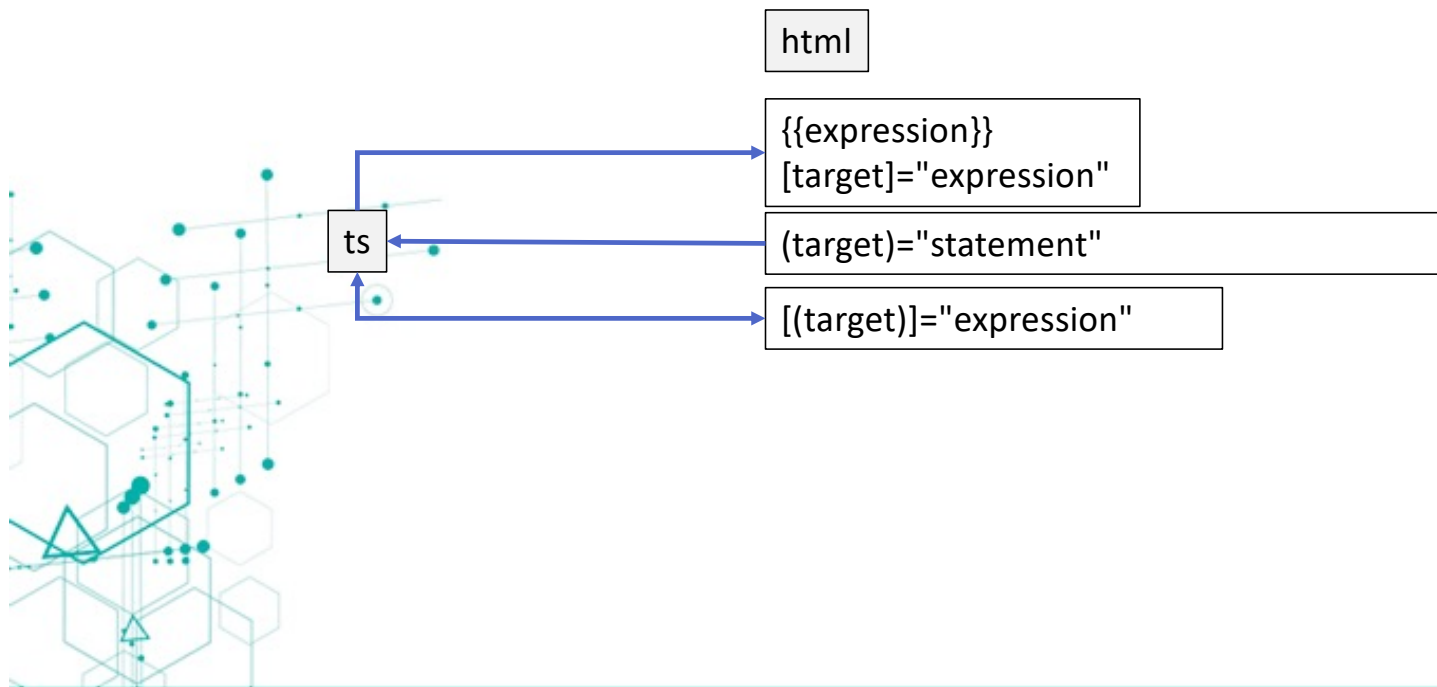
Eventos

```
<ion-button (click)="onClick()">Contar Mais Um!</ion-button>
```

```
export class ContadorComponent implements OnInit {  
  onClick() {  
    window.alert('Ok');  
  }  
}
```

```
src  
└─ app  
   └─ contador  
      ├── <> contador.component.html  
      ├── 📄 contador.component.scss  
      ├── TS contador.component.spec.ts  
      └── TS contador.component.ts
```

Variáveis de Estado



Variáveis de Estado

```
<ion-button expand="block" (click)="onClick()">Contar Mais Um!</ion-button>  
<span>{{ contador }}</span>
```

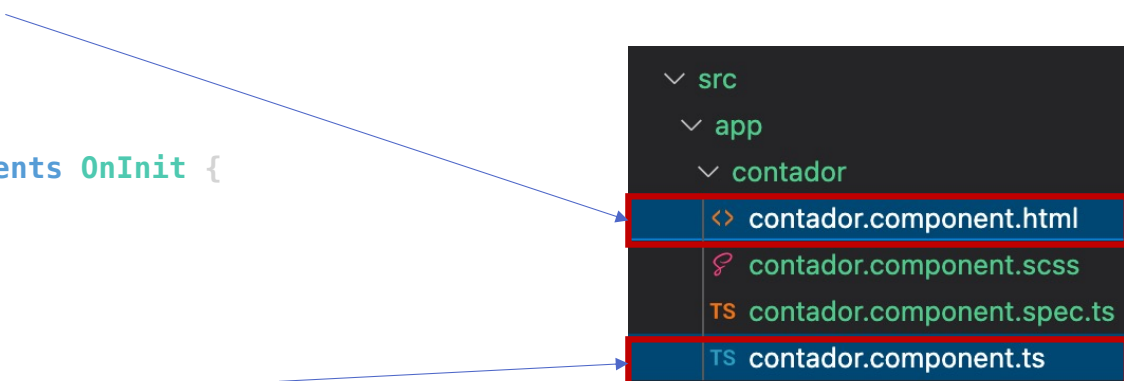
```
export class ContadorComponent implements OnInit {
```

```
  contador = 0
```

```
  constructor() {}
```

```
  ngOnInit() {}
```

```
  onClick() {  
    this.contador++;
```



```
src  
└─ app  
   └─ contador  
      <> contador.component.html  
      📄 contador.component.scss  
      TS contador.component.spec.ts  
      TS contador.component.ts
```

Utilizando um Grid

```
<ion-grid>

<ion-row>
  <ion-col style="text-align:center"><span >{{ contador }}</span></ion-col>
</ion-row>

<ion-row>
  <ion-col style="text-align:center">
    <ion-button (click)="onClick()">Contar Mais Um!</ion-button>
  </ion-col>
</ion-row>

</ion-grid>
```


Criando um Serviço

Um componente que oferece métodos de negócio (sem interface) pode ser definido em um service:

ionic generate service Contador



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

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

  atual = 0

  constructor() { }

  contar() {
    this.atual++
  }
}
```

Utilizando um Serviço

```
import { Component, OnInit } from '@angular/core';
import { ContadorService } from '../contador.service';

@Component({
  selector: 'app-contador',
  templateUrl: './contador.component.html',
  styleUrls: ['./contador.component.scss'],
})
export class ContadorComponent implements OnInit {
  constructor(private contadorService : ContadorService) { }

  ngOnInit() {}

  contar() {
    this.contadorService.contar()
  }
}

<p>
<ion-button (click)="contar()">Contar Mais Um!</ion-button>
</p>
<div>{{ contadorService.atual }}</div>
```

Definindo Valor Contador

```
<ion-row>

<ion-col>

<ion-item>
<ion-label>Valor:</ion-label>
<ion-input [(ngModel)]="novoValor"></ion-input>
</ion-item>
</ion-col>

<ion-col>
<ion-button (click)="setNovoValor()">
<ion-icon name="checkmark-outline"></ion-icon>
</ion-button>
</ion-col>

</ion-row>
```

```
export class ContadorComponent implements OnInit {

  contador = 0
  novoValor = 0;

  constructor() { }

  ngOnInit() {}

  onClick() {
    this.contador++;
  }

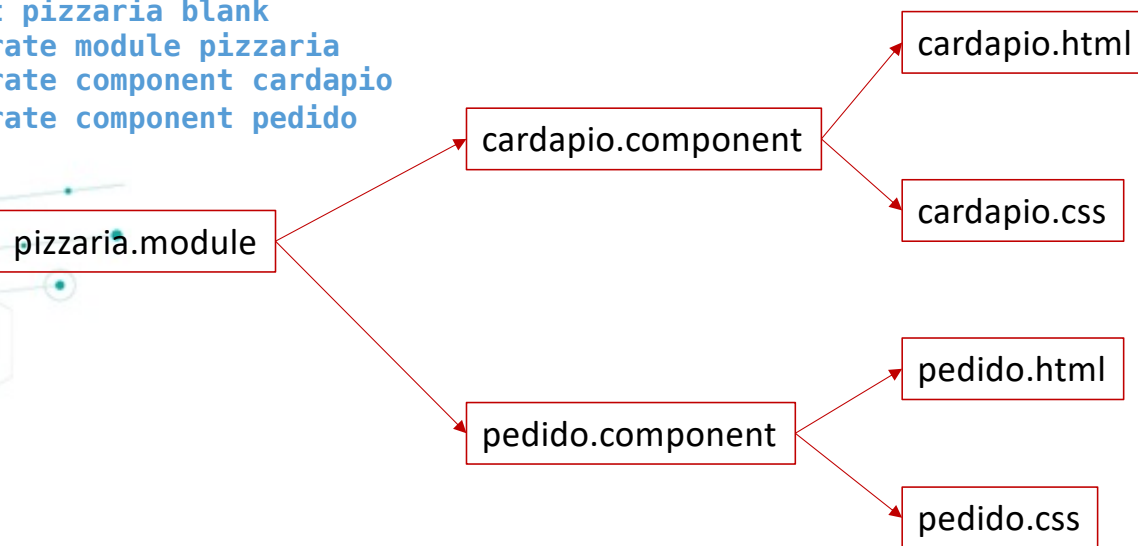
  setNovoValor() {
    this.contador = this.novoValor;
    this.novoValor = 0;
  }

}
```

Pizzaria

Criar um novo módulo

```
ionic start pizzaria blank  
ionic generate module pizzaria  
ionic generate component cardapio  
ionic generate component pedido
```



Pizzaria – cardapio.component.ts

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

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

  ngOnInit() { }
```

Pizzaria – cardapio.component.scss

```
#container {  
  text-align: center;  
  position: absolute;  
  left: 0;  
  right: 0;  
  top: 50%;  
  transform: translateY(-50%);  
}  
#container strong {  
  font-size: 20px;  
  line-height: 26px;  
}  
#container p {  
  font-size: 16px;  
  line-height: 22px;  
  color: #8c8c8c;  
  margin: 0;  
}
```

Pizzaria – cardapio.component.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Cardápio</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  </ion-content>
```

Pizzaria – pedido.component.ts

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

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

  ngOnInit() { }
}
```


Pizzaria – pedido.component.scss

```
#container {  
  text-align: center;  
  position: absolute;  
  left: 0;  
  right: 0;  
  top: 50%;  
  transform: translateY(-50%);  
}  
#container strong {  
  font-size: 20px;  
  line-height: 26px;  
}  
#container p {  
  font-size: 16px;  
  line-height: 22px;  
  color: #8c8c8c;  
  margin: 0;  
}
```

Pizzaria – pedido.component.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>Pedido</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
  </ion-content>
```

Rotas

As rotas são utilizadas quando quando existe a necessidade de navegação entre views distintas

Cada rota é identificada por um caminho de URL (path) apontando para o componente que será exibido

Uma vez definido o array de rotas (objeto Routes) elas devem ser de fato criadas por meio da função `RouterModule.forChild(array rotas)` definido



Pizzaria – pizzaria.module.ts

```
const routes: Routes = [  
  {  
    path: 'cardapio',  
    component: CardapioComponent  
  },  
  {  
    path: 'pedido',  
    component: PedidoComponent  
  }  
];
```

declarations → o que faz parte do módulo

exports → o que será visível fora do módulo

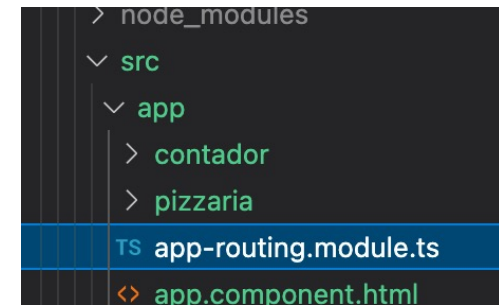
imports → demais módulos e componentes utilizados pelo módulo

```
@NgModule({  
  declarations: [CardapioComponent, PedidoComponent],  
  imports: [CommonModule, RouterModule.forChild(routes)],  
  exports: [CardapioComponent, PedidoComponent]  
})  
  
export class PizzariaModule { }
```

Pizzaria – app-routing.module.ts

```
const routes: Routes = [  
  {  
    path: 'pizzaria',  
    loadChildren: () => import('./pizzaria/pizzaria.module').then( m => m.PizzariaModule)  
  },  
  {  
    path: '',  
    redirectTo: 'pizzaria/cardapio',  
    pathMatch: 'full'  
  }  
];
```

A rota inicial (path: "") efetua um redirecionamento para pizzaria → cardapio



Pizzaria – cardapio.component.ts

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

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

export class CardapioComponent implements OnInit {

  Cardapio: any = [];

  constructor() { }






  ngOnInit() {
    this.Cardapio = {"pizzas": [{"id": "1", "nome": "Quatro Queijos", "preco": 30.0},
    {"id": "2", "nome": "Frango com Catupiry", "preco": 35.0 }]};
  }
}
```

Pizzaria – cardapio.component.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Cardápio
    </ion-title>
  </ion-toolbar>
</ion-header>
```

```
<ion-content [fullscreen]="true">
  <ion-list>
    <ion-list-header>
      <ion-label>Pizzas</ion-label>
    </ion-list-header>
    <ion-item *ngFor="let pizza of Cardapio.pizzas">
      <ion-label><h2>{{pizza.nome}}</h2>
      <h3>{{pizza.preco}}</h3></ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

```
{
  "pizzas": [
    {
      "id": "1",
      "nome": "Quatro Queijos",
      "preco": 30.0
    },
    {
      "id": "2",
      "nome": "Frango com Catupiry",
      "preco": 35.0
    },
    ...
  ]
}
```

Cardápio	
Cardápio	
	Quatro Queijos 30
	Frango com Catupiry 35
	Atum com queijo 40
	Presunto Parma 80
	Brigadeiro 27
	Banana com Doce de Leite 40

***ngFor** realiza um loop percorrendo todos os objetos armazenados na variável Cardapio que contém um array JSON com as pizzas

Pizzaria – cardapio.component.html e cardapio.component.ts

```
<ion-content [fullscreen]="true">
<ion-list>
<ion-list-header>
<ion-label>Pizzas</ion-label>
</ion-list-header>
<ion-item *ngFor="let pizza of Cardapio.pizzas" button (click)="selPizza(pizza)">
<ion-label><h2>{{pizza.nome}}</h2>
<h3>{{pizza.preco}}</h3></ion-label>
</ion-item>
</ion-list>
</ion-content>
```

Evento (click) aciona a função selPizza passando o objeto JSON correspondente à pizza selecionada pelo cliente...

```
export class CardapioComponent implements OnInit {

  PizzaSelecionada: any;

  constructor() { }

  selPizza(item) {
    this.PizzaSelecionada = item;
  }

}
```


Acesso Backend

Uma aplicação quase sempre necessita acessar dados ou serviços implementados em um servidor (backend)

Normalmente, tais serviços estão disponíveis na forma de endpoints RESTFul

No Angular pode-se utilizar uma biblioteca para gerar acessos HTTP (GET, POST, PUT, etc...)

Para isso será necessário importar o componente http do pacote **@angular/common/http**

Exemplo de POST:

```
this.http.post(URL, {param: "Valor"}).subscribe((response) => {  
  console.log(response);  
});
```

Pizzaria – cardapio.component.ts

Obter a lista de pizzas de um arquivo disponível na internet (link abaixo)

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

const URL = 'https://raw.githubusercontent.com/esensato/mobile-2022-01/main/services/lista-pizza.json';

export class CardapioComponent implements OnInit {
  Cardapio: any = [];

  constructor(private http: HttpClient) { }

  ngOnInit() {
    this.http.get(URL).subscribe((response) => {
      this.Cardapio = response;
    });
  }
}
```

Pizzaria – cardapio.component.html

Exibe as imagens das pizzas

```
<ion-header [translucent]="true">
<ion-toolbar>
<ion-title>Cardápio</ion-title>
</ion-toolbar>
</ion-header>

<ion-content [fullscreen]="true">
<ion-list [hidden]="0"cultarCardapio>
<ion-list-header>
<ion-label>Cardápio</ion-label>
</ion-list-header>

<ion-item *ngFor="let pizza of Cardapio.pizzas" button (click)="selPizza(pizza)">
<ion-avatar slot="start">

</ion-avatar>

<ion-label><h2>{{pizza.nome}}</h2>
<h3>{{pizza.preco}}</h3></ion-label>
</ion-item>
</ion-list>
```

Pizzaria – cardapio.component.html

Selecionar a bebida

```
<ion-list [hidden]!=0cultarCardapio>
<ion-list-header>
<ion-label>Bebidas</ion-label>
</ion-list-header>
<ion-item *ngFor="let bebida of Cardapio.bebidas" button (click)="selBebida(bebida)">
<ion-avatar slot="start">

</ion-avatar>

<ion-label><h2>{{bebida.nome}}</h2>
<h3>{{bebida.preco}}</h3></ion-label>
</ion-item>
</ion-list>
```

A propriedade [hidden] é utilizada aqui para exibir / ocultar o cardápio de bebidas

Pizzaria – cardapio.component.ts

```
import { Router, NavigationExtras } from '@angular/router';

export class CardapioComponent implements OnInit {

  PizzaSelecionada: any;
  OcultarCardapio = false;

  constructor(private router: Router, private http: HttpClient) { }

  selPizza(item) {
    this.PizzaSelecionada = item;
    this.OcultarCardapio = true;
  }

  selBebida(item) {
    this.BebidaSelecionada = item;
    let navigationExtras: NavigationExtras = {
      state: {
        pizzaSelecionada: this.PizzaSelecionada,
        bebidaSelecionada: this.BebidaSelecionada
      }
    };
    this.router.navigate(['pizzaria/pedido'], navigationExtras);
  }
}
```

Redireciona para a tela de resumo do pedido passando a pizza e a bebida selecionada como parâmetro

Pizzaria – pedido.component.ts

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

export class PedidoComponent implements OnInit {

  constructor(private route: ActivatedRoute, private router: Router) {
    this.route.queryParams.subscribe(params => {
      if (this.router.getCurrentNavigation().extras.state) {
        console.log (this.router.getCurrentNavigation().extras.state.pizzaSelecionada);
        console.log (this.router.getCurrentNavigation().extras.state.bebidaSelecionada);
      }
    });
  }
}
```

Exemplo da leitura dos parâmetros na tela de
resumo do pedido

Pizzaria – pedido.component.html

```
<ion-card>
<div style="display: flex; align-items: center; justify-content: center;">


</div>
<ion-card-header>
<ion-card-subtitle>Confira abaixo o seu pedido</ion-card-subtitle>
<ion-card-title>Pedido</ion-card-title>
</ion-card-header>
<ion-card-content>

</ion-card-content>
</ion-card>

</ion-content>
```

Uso de um Card para exibir o resumo do pedido

Pedido




Confira abaixo o seu pedido

Pedido

Presunto Parma - R\$ 80

Água - R\$ 4 3

☒ Queijo Extra

 Total: R\$ 92

Nome: ☒

Pizzaria – pedido.component.html

```
<ion-card-content>
<ion-item>
  {{PizzaSelecionada.nome}} - R$ {{PizzaSelecionada.preco}}
</ion-item>

<ion-item>
  <div style="padding-right: 20px;">{{BebidaSelecionada.nome}} - R$ {{BebidaSelecionada.preco}}</div>
  <div><ion-input type="number" size="2" value="1" (ionChange)="recalcular($event.target.value)"
  value={{quantidade}}></ion-input></div>
</ion-item>

<ion-item>
  <div style="padding-right: 5px;"><ion-checkbox (ionChange)="queijoExtra($event.target)"></ion-
  checkbox></div><div>Queijo Extra</div>
</ion-item>

<ion-item>
  <div style="padding-right: 5px;"><ion-icon name="cash-outline"></ion-icon></div>
  <div>Total: R$ {{total}}</div>
</ion-item>
</ion-card-content>
```

ion-input e ion-checkbox com evento (ionChange)

Pizzaria – pedido.component.html

```
<ion-card-content>
```

```
...
```

```
<ion-item>
<div style="padding-right: 5px;">Nome:</div>
<div><ion-input name="nome" ngDefaultControl [(ngModel)]="nome"></ion-input></div>
<div><ion-button (click)="enviar()">
<ion-icon name="checkmark-outline"></ion-icon>
</ion-button></div>
</ion-item>
</ion-card-content>
```

```
ion-input {
--placeholder-color: green;
color: var(--ion-color-primary);
}
```

Pizzaria – pedido.component.ts

```
import { ActivatedRoute, Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';

export class PedidoComponent implements OnInit {

  PizzaSelecionada: any;
  BebidaSelecionada: any;
  total: number = 0.0;
  quantidade: number = 1;
  nome: string = '';

  constructor(private route: ActivatedRoute, private router: Router, private http: HttpClient) {
    this.route.queryParams.subscribe(params => {
      if (this.router.getCurrentNavigation().extras.state) {
        this.PizzaSelecionada = this.router.getCurrentNavigation().extras.state.pizzaSelecionada;
        this.BebidaSelecionada = this.router.getCurrentNavigation().extras.state.bebidaSelecionada;
        this.total = parseFloat(this.PizzaSelecionada.preco) + parseFloat(this.BebidaSelecionada.preco);
      }
    });
  }
}
```

Definindo as variáveis de estado

Pizzaria – pedido.component.ts

```
export class PedidoComponent implements OnInit {  
  
  ...  
  
  recalcular(valor) {  
    this.quantidade = valor;  
    this.total = this.quantidade * parseFloat(this.BebidaSelecionada.preco) +  
    parseFloat(this.PizzaSelecionada.preco);  
    console.log(this.quantidade, valor);  
  }  
  
  queijoExtra(valor) {  
    console.log(valor.checked);  
    if (valor.checked) {  
      this.total += 10;  
    } else {  
      this.total -= 10;  
    }  
  }  
}
```

Recalculando os valores

Extra – Criação de um Backend na Cloud

Criar um backend na cloud para armazenar os pedidos de pizza

```
this.http.post(URL, {param: "Valor"}).subscribe((response) => {  
  console.log(response);  
});
```



Exercício

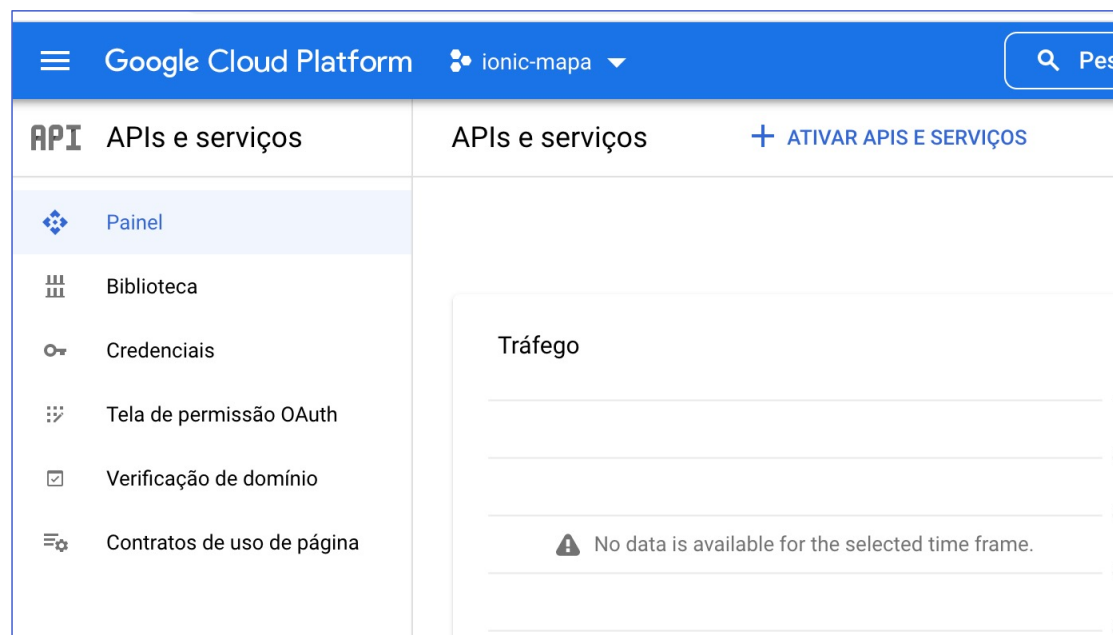
Criar um app que permita a busca por universidades ao redor do mundo passando os parâmetros name e country conforme a chamada abaixo:

<http://universities.hipolabs.com/search?name=middle&country=turkey>

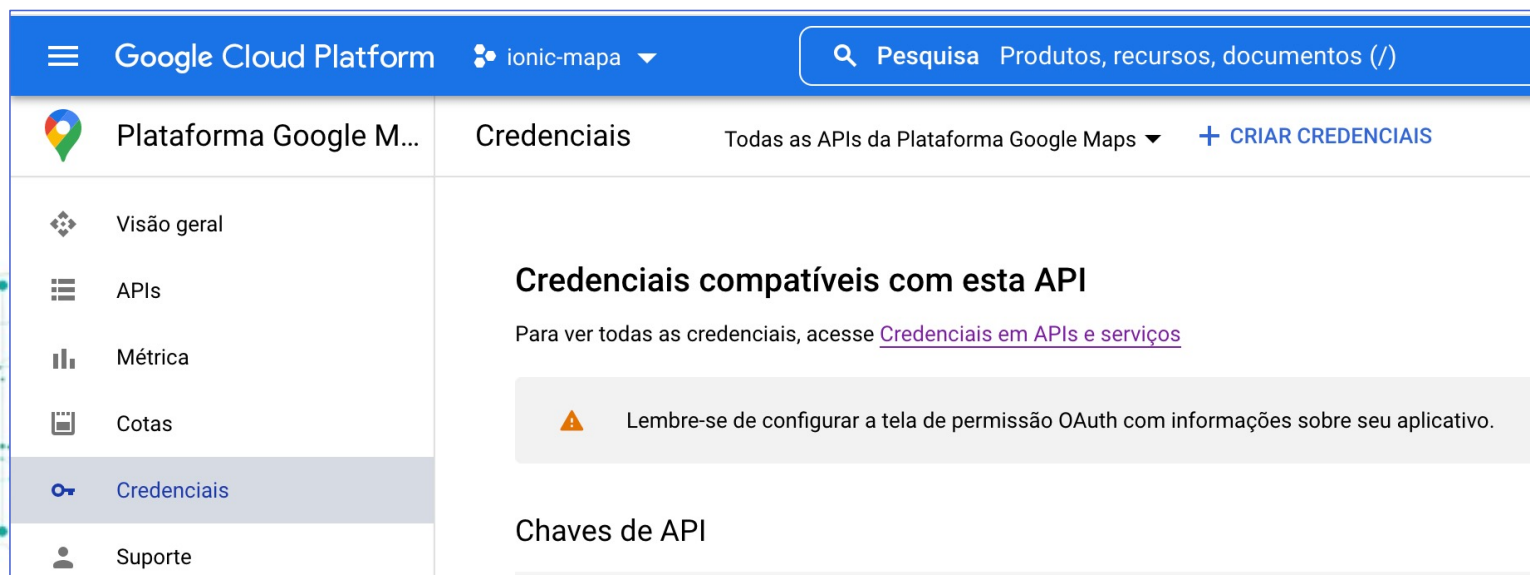


Google Maps

<https://console.developers.google.com/>



Google Maps



The screenshot displays the Google Cloud Platform console interface. At the top, the header includes the Google Cloud Platform logo, the project name 'ionic-mapa', and a search bar with the text 'Pesquisa Produtos, recursos, documentos (/)'. The main content area is titled 'Plataforma Google M...' and 'Credenciais'. It shows a list of APIs with 'Google Maps Platform' selected. The 'Credenciais' tab is active, displaying a warning message: 'Lembre-se de configurar a tela de permissão OAuth com informações sobre seu aplicativo.' Below this, the 'Chaves de API' section is visible.

Google Cloud Platform

ionic-mapa

Pesquisa Produtos, recursos, documentos (/)

Plataforma Google M...

Credenciais

Todas as APIs da Plataforma Google Maps

+ CRIAR CREDENCIAIS

Visão geral

APIs

Métrica

Cotas

Credenciais

Suporte

Credenciais compatíveis com esta API

Para ver todas as credenciais, acesse [Credenciais em APIs e serviços](#)

Lembre-se de configurar a tela de permissão OAuth com informações sobre seu aplicativo.

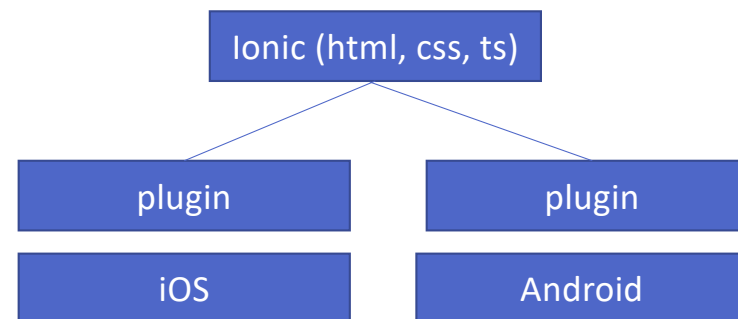
Chaves de API

Capacitor Plugins

Instalar os pacotes abaixo para incluir as capacidades nativas de acesso a mapas e geolocalização

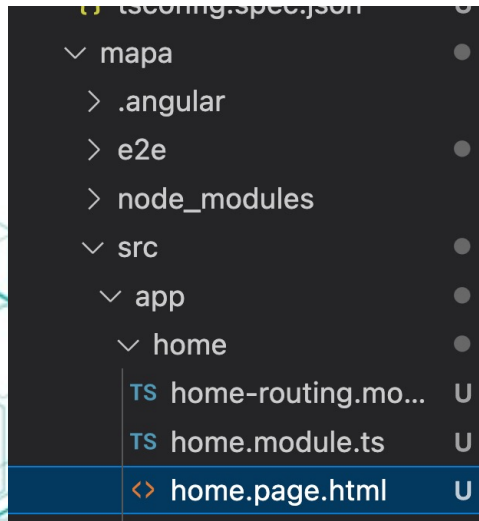
Essas capacidades são independentes de plataforma (iOS, Android), implementadas por meio de plugins contidos no projeto capacitor (vide <https://capacitorjs.com/>)

```
npm i --save @capacitor/google-maps  
npm i --save @capacitor/geolocation
```



Google Maps

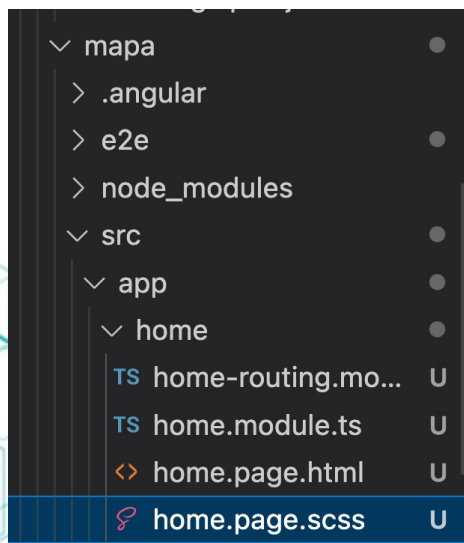
Incluir o componente na página HTML com um identificador (no caso #map)



```
<capacitor-google-maps #map></capacitor-google-maps>
```

Google Maps

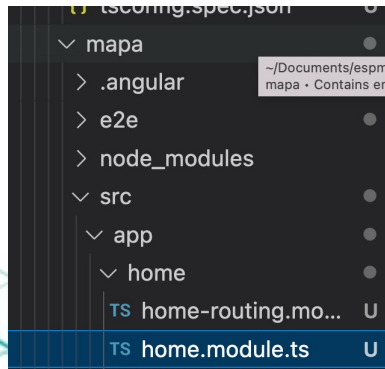
Definir a folha de estilo



```
capacitor-google-maps {  
  display: inline-block;  
  width: 275px;  
  height: 400px;  
}
```

Google Maps

Habilitar o custom schema especificamente para o Angular

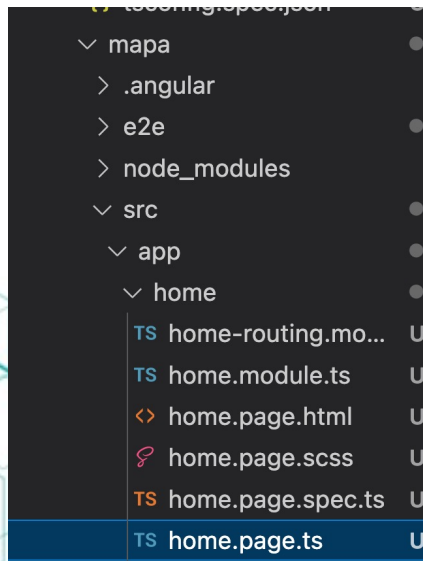


```
import { CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
```

```
@NgModule({  
  imports: [  
    CommonModule,  
    FormsModule,  
    IonicModule,  
    HomePageRoutingModule  
  ],  
  declarations: [HomePage],  
  schemas: [CUSTOM_ELEMENTS_SCHEMA]  
})  
export class HomePageModule {}
```

Google Maps

Exibir o mapa



```
export class HomePage {

  @ViewChild('map')
  mapRef: ElementRef<HTMLInputElement>;
  newMap: GoogleMap;

  constructor() {}

  ionViewDidEnter() {
    this.createMap();
  }

  async createMap() {
    this.newMap = await GoogleMap.create({
      id: 'my-map',
      element: this.mapRef.nativeElement,
      apiKey: '<<CHAVE_API>>',
      config: { center: {lat: 33.6, lng: -117.9}, zoom: 8}});
  }
}
```

Google Maps

Capturando eventos de click sobre o mapa

```
this.newMap.setOnMapClickListener((val) => {  
  console.log(val.latitude, val.longitude)  
})
```



Google Maps

Adicionando marcadores

```
let m: Marker = {coordinate:{lat: val.latitude, lng:val.longitude}, title: 'Aqui', snippet: 'Meu marcador'}  
this.newMap.addMarker(m)
```

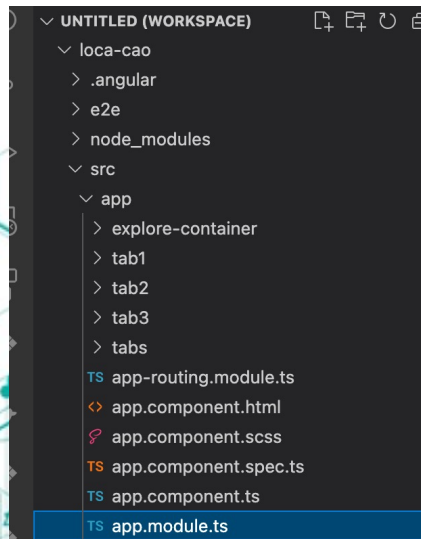
Capturando eventos de click sobre os marcadores

```
this.newMap.setOnMarkerClickListener((marker) => {  
  console.log(marker)  
})
```

Localização Atual

```
npm install @awesome-cordova-plugins/geolocation --save
```

```
npm install cordova-plugin-geolocation --save
```



```
import { Geolocation } from '@awesome-cordova-plugins/geolocation/ngx';
```

```
@NgModule({  
  declarations: [AppComponent],  
  entryComponents: [],  
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule],  
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy, }, Geolocation],  
  bootstrap: [AppComponent],  
})
```

Localização Atual

```
constructor(public router: Router, private geolocation: Geolocation) {}
```

```
this.geolocation.getCurrentPosition().then((resp) => {
```

```
  this.myLat = resp.coords.latitude;
```

```
  this.myLong = resp.coords.longitude;
```

```
}).catch((error) => {
```

```
  console.log('Error getting location', error);
```

```
});
```


Monitorando a Localização

```
navigator.geolocation.watchPosition((pos) => {  
  alert(pos);  
}, (err) => {  
  alert(err);  
}, {  
  maximumAge: 3600000,  
  timeout: 3000,  
  enableHighAccuracy: true,  
});
```

Instalando Android SDK

Efetuar o download do **Android Command Line Tools**:

<https://developer.android.com/studio#command-tools>

Extrair arquivos em:

/android-sdk/cmdline-tools/latest

Executar os comandos na pasta **bin**:

```
sdkmanager.bat --no_https --list
```

```
sdkmanager.bat --no_https --install "platforms;android-32"
```

```
sdkmanager.bat --no_https --install "build-tools;32.0.0"
```

```
sdkmanager.bat --no_https --install platform-tools
```

```
sdkmanager.bat --no_https --install "system-images;android-32;google_apis_playstore;x86_64"
```

```
sdkmanager.bat --no_https --install emulator
```

Instalando Android SDK - Windows

```
sdkmanager.bat --no_https --install "extras;intel;Hardware_Accelerated_Execution_Manager"
```

Na pasta android-sdk\extras\intel\Hardware_Accelerated_Execution_Manager executar a instalação do IntelHAXM
(é um arquivo executável)



Instalando Android SDK

`sdkmanager.bat --licenses`

Ir para a pasta

`android-sdk\cmdline-tools\latest\bin`

`avdmanager.bat create avd -n dev -k "system-images;android-32;google_apis_playstore;x86_64"`

Listar AVD's (pasta `android-sdk\emulator`):

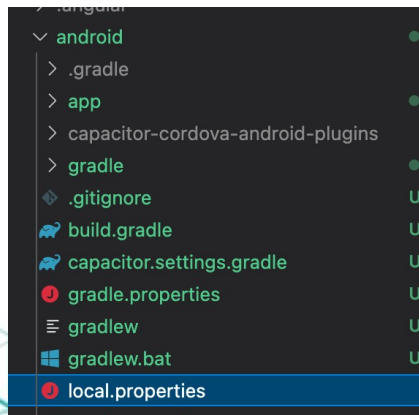
`emulator.bat -list-avds`

Iniciar o AVD (pasta `android-sdk\emulator`):

`emulator.bat -avd dev`

`emulator.bat -avd dev -skindir /Users/esensato/Documents/espm/software/android-sdk -skin Galaxy_S22_Ultra`

Executando no Emulador



`sdk.dir=/espm/software/android-sdk`

`ionic capacitor add android`

`ionic capacitor run android -l --address 0.0.0.0`