O que são módulos.

Módulos angular são itens de configuração do framework que servem para agregar componente, diretivas, serviços e pipes, mas não apenas com objetivo de organizar seu código, você vai poder decidir o que vai ser exportado para fora de seu módulo e ainda como a injeção de dependência vai ser configurada.

Além disso, imagina que se tem uma aplicação grande e que seja acessada por milhares de usuários, é bom provável que esses usuáios façam partes de setores diferentes de uma empresa e isso faz com que esses usuários acessem informações bem distintas dentro da sua aplicação, por isso, não é interessante carregar toda aplicação para todos eles, pois uma parte vai estar interessada apenas em um pedaço da aplicação, com módulos é possível dividir minha aplicação em partes independentes e definir que essas partes não sejam carregadas de imediato e sim apenas quando requisitadas. Com isso o startup da aplicação vai ficar mais rápido porque de inicio as pessoas não precisaram esperar por algo que elas não vão usar – Isso é chamado de LazyLoading(carregamento tardio).

Tipos de módulos:

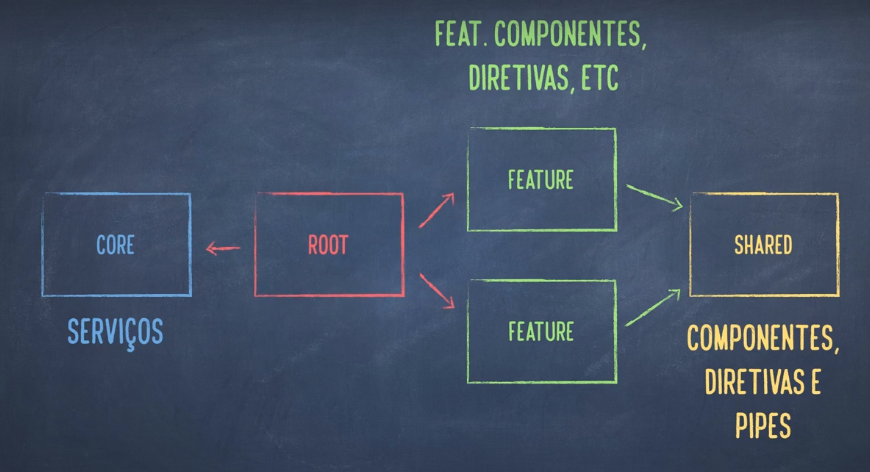
* ROOT MODULE - módulo raiz da aplicação onde tem as declarações do componente, é o modulo que faz o bootstrap da aplicação.
* SHARED E CORE MODULES – módulo compartilhado com todos os componentes que são usados por outros módulos da aplicação. Core módulo com todos os serviços da aplicação, esse módulo não contém componentes apenas serviços.
* FEATURE MODULE – são os módulos que contêm as implementações das funcionalidades da aplicação.

Separar em módulos pode ajudar a tornar o carregamento da aplicação mais rápido porque a gente pode escolher apenas um conjunto de filhos a serem carregadas no início.

Para declarar um módulo usamos um decorator chamado @NgModule({});

Nele listamos os componentes que devem ser visíveis fora do módulo no exports[];

Sem isso os componentes não serão visíveis.



Carregamento tardio de módulos (Lazy-Loading):

No angular é necessário criar um módulo separado para o componente que queremos adicionar o lazy loading:

Criamos o arquivo: about.module.ts.

Nesse arquivo colocamos as configurações do módulo:

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

const ROUTES: Routes = [

    {path: '', component: AboutComponent}

]

@NgModule({

    declarations:[AboutComponent],

    imports: [

        RouterModule.forChild(ROUTES)

    ]

})

export class AboutModule{}

É o módulo de roteamento router que faz o lazy loading. Configuramos para quando alguém acessar aquela rota carregamos o módulo naquele momento.

Na nossa rota é necessário deixar explicito que aquele conteúdo será carregado de forma tardia, para isso tiramos a importação do componente que vamos aplicar o carregamento tardio, e na rota trocamos o atributo “component” por loadChildren, e em vez de passar uma referência para o componente passamos uma string com o módulo que iremos usar:

Antes:

 {path: 'about, component: AboutComponent,

Depois:

{path: 'about', loadChildren: './about/about.module.ts#AboutModule'},

Quando estamos no nosso módulo raiz, nós importamos as rotas de cada módulo utilizando o método forRoot() passando uma referencia das rotas:

    RouterModule.forRoot(ROUTES),

É necessário definir rotas padrões no novo módulo criado, para isso criamos a mesma estrutura criada no módulo de rotas:

const ROUTES: Routes = [

    {path: '', component: AboutComponent}

]

Nesse caso, por se tratar de um componente filho e não o raiz, utilizamos o método forChild()

@NgModule({

    declarations:[AboutComponent],

    imports: [

        RouterModule.forChild(ROUTES)

    ]

})

Criando um módulo compartilhado (Shared módule):

Módulos compartilhados são usados quando temos componentes que serão utilizados por diversos outros componentes. Deixando mais organizada nossa aplicação, por exemplo quando temos 3 componentes compartilháveis, podemos criar um módulo para os três e depois importar no módulo raiz apenas uma vez. EX:

Temos 3 componentes compartilhados na aplicação: InputComponent, RadioComponent e RatingComponent. No módulo raiz os 3 componentes são declarados, e também os módulos que eles usam como por exemplo o FormsModule. Criando um módulo compartilhado, podemos juntar tudo isso em um módulo e depois importar apenas uma vez na raiz:

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

import {CommonModule} from '@angular/common';

import { InputComponent } from './input/input.component';

import { RadioComponent } from './radio/radio.component';

import { RatingComponent } from './rating/rating.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({

    declarations: [InputComponent, RadioComponent, RatingComponent],

    imports: [CommonModule,FormsModule, ReactiveFormsModule],

    exports: [InputComponent, RadioComponent, RatingComponent,

              CommonModule,FormsModule, ReactiveFormsModule]

})

export class SharedModule{}

No app.module:

 imports: [

    BrowserModule,

    HttpModule,

   // FormsModule, //já está sendo importado e exportado pelo módulo compartilhado

    //ReactiveFormsModule,

    SharedModule, // módulo compartilhado

    //HttpClientModule,

    RouterModule.forRoot(ROUTES),

   // HttpClientModule

    //Ng4LoadingSpinnerModule

  ],

Criando um Feature module também utilizando lazy-loading.

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

import { OrderComponent } from './order.component';

import { OderItemsComponent } from './oder-items/oder-items.component';

import { DeliveryCostsComponent } from './delivery-costs/delivery-costs.component';

import { SharedModule } from '../shared/shared.module';

import {RouterModule, Routes} from '@angular/router'

const ROUTES: Routes = [

    {path: '', component: OrderComponent}

]

@NgModule({

    declarations: [ DeliveryCostsComponent, OderItemsComponent, OrderComponent],

    //nós usamos alguns componentes importados no sherdModule, por isso iremos eimportá=lo aqui

    imports: [SharedModule, RouterModule.forChild(ROUTES)]

})

export class OrderModule{}