1.Crear el espacio de trabajo

ng new workspace - - create-application=false

2.Crear la primera aplicación host

Ng g application host - - style=css - - routing=true - -no-standalone --routing --ssr=false

3. crear la aplicacion de micriofrontEnd

sudo ng g application mf-simulador --style=css --no-standalone --routing --ssr=false

4. crear un componente en el componente host

ng g c --project=host header

ng g c --project=host home

5. correr un Proyecto

Ng serve --project=host

6.Crear un modulo en el mf-simulador

Ng g m simulador –-project=mf-simulador –routing

7.crear un componente en mf-simulador

ng g c --project=mf-simulador simulador

ng g c --project=mf-simulador default

8. configurar el archive routin.modulo de mf-simulador

Que te mande al componente default

const routes: *Routes* = [

{

path:'',

component:DefaultComponent,

pathMatch:'full'

}

];

9.Agregamos su ptro path del mismo componente de arriba pero para que cargue por lazyloading

ç const routes: *Routes* = [

{

path:'',

component:DefaultComponent,

pathMatch:'full'

},

*//Para que cargue con lazyloading*

{

path:'simulador',

loadChildren:()=>import('./simulador/simulador.module').then(m=>m.SimuladorModule)

}

];

Tenemos que crear un modulo por cada ruta

10. Agregar en simulador.routing

Que cargue su propio componente

const routes: *Routes* = [{

path:'',

component:SimuladorComponent

}];

11. ahora si hacer microfront end

Ng add @angular-architects/module-federation --project=mf-simulador –port 4242

12. configurar su archive webpack de el proyecto simulador

Como es un remoto = microservicio se configura de esta manera

const { shareAll, withModuleFederationPlugin } = require('@angular-architects/module-federation/webpack');

*module*.*exports* = withModuleFederationPlugin({

name: 'mf-simulador',

exposes: {

'./Component': './projects/mf-simulador/src/app/app.component.ts',

},

name:'mf-simulador',

filename:'remoteEntry.js',

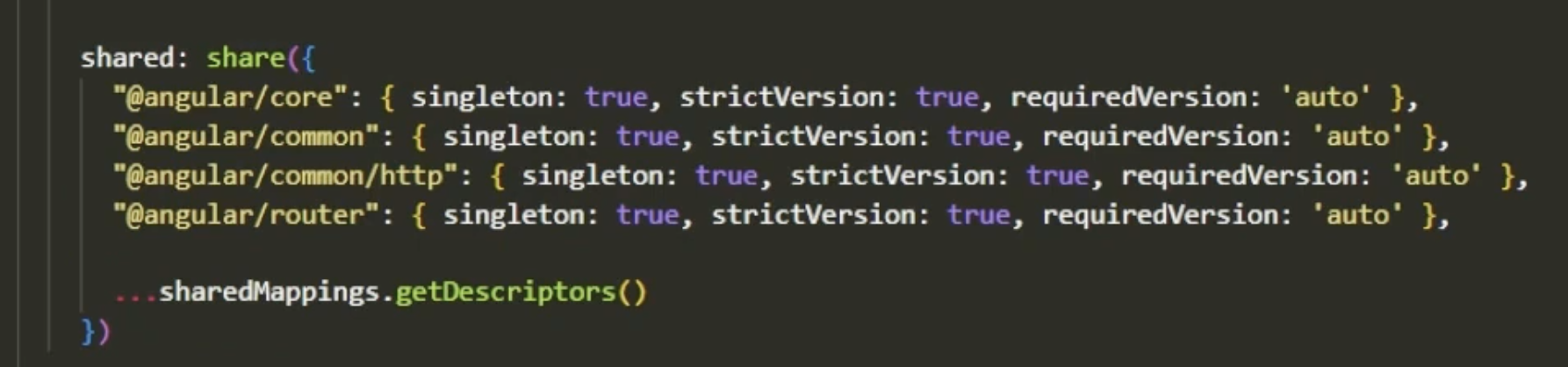
exposes:{'./SimuladorModule':'./projects/mf-simulador/src/app/simulador/simulador.module.ts'},

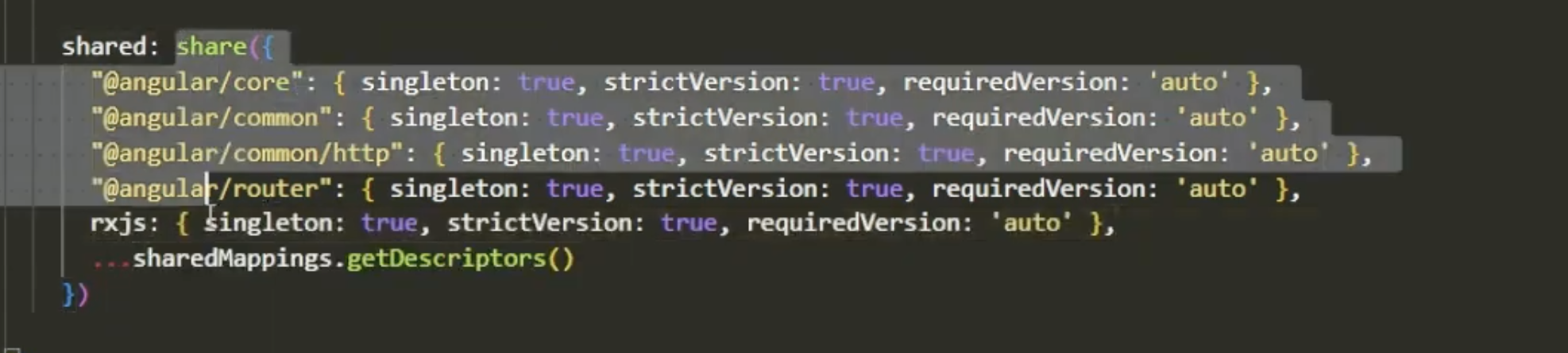
shared: {

...shareAll({ singleton: true, strictVersion: true, requiredVersion: 'auto' }),

},

});





12.crear el microfrontEnd de la host

sudo ng add @angular-architects/module-federation --project=host --port 5000

13. configurar su archive webpack de el proyecto host

const { shareAll, withModuleFederationPlugin } = require('@angular-architects/module-federation/webpack');

*module*.*exports* = withModuleFederationPlugin({

remotes:{

"mf-simulador":"http://localhost:4242/remoteEntry.js",

},

shared: {

...shareAll({ singleton: true, strictVersion: true, requiredVersion: 'auto' }),

},

});

13.Configurar el routing de el host para que sea con lazyloading la ruta de simulador del proyecto mf-simulador

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

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

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

const routes: *Routes* = [

{

path:'',

component:HomeComponent,

pathMatch:'full'

},

{

path:'mf-simulador',

loadChildren:()=>import('mf-simulador/SimuladorModule').then(m=>m.SimuladorModule)

}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class *AppRoutingModule* { }

(tiene que ser igual al webpack del microservicio = mf-simulador)

se pone rojo el loadChildren para eso creamos un archive en la carpeta de host llamado modulos.d.ts

dentro esto

declare module 'mf-simulador/SimuladorModule'

corregido

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

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

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

const routes: *Routes* = [

{

path:'',

component:HomeComponent,

pathMatch:'full'

},

{

path:'mf-simulador',

loadChildren:()=>import('../../../mf-simulador/src/app/simulador/simulador.module').then(m=>m.SimuladorModule)

}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class *AppRoutingModule* { }