

# Microfrontends Webpack (Module federation).

## Configuraciones

### Comandos

Primeramente se ejecutan los siguientes comandos

### En cada Microfront

```
npm i @angular-architects/module-federation@18.0.6
```

```
ng g @angular-architects/module-federation:init --project remote --port 4201 --type remote
```

**Nota:** asegurarse que el flag `--project` sea el nombre del `angular.json`

### En host-app

```
ng g @angular-architects/native-federation:init --project host-app --port 4200 --type dynamic-host
```

## Modificaciones al código.

### En cada Microfront

Crear un modulo **entry/entry.module** con las importaciones raices y un routing **entry/entry-routing.module**

**Nota:** ver proyecto de referencia: (mfcajasMenores) es el primero que se configuro como microfrontend

## Forma de exponer Microfrontends

Exponiendo modulo raiz de la aplicacion en **webpack.config**

en este caso exponemos **entry.module.ts** y un **HeaderInterceptor** que se inyectará en la raiz del **Host**

## webpack.config.js

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

module.exports = withModuleFederationPlugin({
  name: 'mfcajamenor',

  exposes: {
    './Routes': './src/app/entry/entry.module.ts',
    './HeaderInterceptor':
'./src/app/core/interceptor/header.interceptor.ts'
  },

  shared: {
    ...shareAll({ singleton: true, strictVersion: false, requiredVersion:
'auto' }),
  },
});
```

Ejemplo Carga remota desde el **app.routes** del Host

## .app.routes.ts

```
import { Routes } from '@angular/router';
import { loadRemoteModule } from '@angular-architects/module-federation';
import { environment } from '../environments/environment';

export const routes: Routes = [
  {
    path: 'cajas-menores',
    loadChildren: () =>
      loadRemoteModule({
        type: 'module',
        remoteEntry: `${environment.mfcajas_menores_url}/remoteEntry.js`,
        exposedModule: './Routes'
      }).then(m => m.EntryModule)
  },
  {
    path: 'centro-costos',
    loadChildren: () =>
      loadRemoteModule({
        type: 'module',
        remoteEntry: `${environment.mfcentro_costos_url}/remoteEntry.js`,
        exposedModule: './Routes'
      }).then(m => m.EntryModule)
  }
];
```

## Modificacion en la navegacion de los microfrontends

Anteponer el nombre del microfrontend en todos los enlaces para que pueda ser tomado por el HOST

\* router.navigate Asi siendo **“caja-menores/”** el nombre con el que se llamo el microfront en el path de app.routes.ts desde el HOST

```
this._router.navigate([`/cajas-menores/gastos`]);
```

## Se debe cargar el headerInterceptor en el app.config.ts del HOST

```
const loadInterceptor = async () => {
  const headersInterceptor = await loadRemoteModule({
    type: 'module',
    remoteEntry: `${environment.mfcajas_menores_url}/remoteEntry.js`,
    exposedModule: './HeaderInterceptor'
  }).then(m => m.headersInterceptor);
  return headersInterceptor;
};

// Creamos un interceptor que delegará al interceptor remoto una vez que
esté cargado
const delegatingInterceptor: HttpInterceptorFn = (req, next) => {
  return new Observable(subscriber => {
    loadInterceptor().then(remoteInterceptor => {
      remoteInterceptor(req, next).subscribe(subscriber);
    });
  });
};
```

inyectar la funcion arriba ilustrada **“delegatingInterceptor()”** de tipo **HttpInterceptorFn** asi:

```
provideHttpClient(
  withInterceptors([delegatingInterceptor])
)
```

[←Regresar](#)

From:  
<http://wiki.adacsc.co/> - Wiki

Permanent link:  
<http://wiki.adacsc.co/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:front:configuracion-microfrontend>

Last update: 2025/03/06 16:40

