

PROYECTO “MEJORAMIENTO DE LA CAPACIDAD PARA LA GENERACIÓN DEL
CONOCIMIENTO Y MEJORA CONTINUA EN LA GESTIÓN DE LA CONTRATACIÓN
PÚBLICA” CON CUI N°2394412”

DOCUMENTO DE CONFIGURACIÓN DEL COMPONENTE MICROFRONTEND EN PROYECTOS HOST

v0.10

Autor: Renee Alfredo Ruiz Chunga

Fecha: 09/04/2025

ORGANISMO SUPERVISOR DE LAS CONTRATACIONES DEL ESTADO

HISTORIAL DE CAMBIOS

Versión	Descripción del Cambio	Fecha	Autor
v0.1	Documento inicial.	30/01/2025	Renee Ruiz
v0.2	Se añade instalación de librería compartida sgu-shared y estilos compartidos.	31/01/2025	Renee Ruiz
v0.3	Se incluye configuración del archivo 'federation.config.js' y sección para configurar WebSocket para notificaciones.	07/02/2025	Renee Ruiz
v0.4	Se añade configuración del cierre de sesión con observable cerrarSesion\$.	12/02/2025	Renee Ruiz
v0.5	Se actualiza función de cierre de sesión: se pasa clientId y URL a la función 'sesionFinalizada'.	18/02/2025	Renee Ruiz
v0.6	Se añade envío de token y llaves de acceso para catálogos comunes desde el header.	11/03/2025	Renee Ruiz
v0.7	Se actualiza librería a la versión 0.0.19. Se actualiza nombre del observable de cierre de sesión a cerrarSesionClick\$ y se cambia a función 'finalizarSesion'.	13/03/2025	Renee Ruiz
v0.8	Se actualiza librería a la versión 0.0.24. Se actualiza uso de funciones con nueva versión de la librería.	26/03/2025	Renee Ruiz
v0.9	Se actualiza librería a la versión 0.0.26. Se actualiza uso de funciones de la librería osce-sgu-shared. (Ver puntos 4, 6 y 7 de este documento)	27/03/2025	Renee Ruiz
v0.10	Se actualiza url del publicado del microfrontend (Ver punto 2)	09/04/2025	Renee Ruiz

INDICE

1.	Instalar Dependencias Necesarias.....	4
2.	Configurar el archivo federation.manifest.json.....	4
3.	Configurar federation.config.js.....	4
4.	Instalar librería compartida:	5
5.	Uso de páginas del microfrontend	5
6.	Establecer parámetros del sistema:	7
7.	Uso del header del microfrontend y configuración del cierre de sesión	7
8.	Verificar la Integración	9

Integración de Microfrontend de SGU

1. Instalar Dependencias Necesarias

En el proyecto donde se integrará el microfrontend (host), instala las siguientes dependencias de @angular-architects/native-federation:

```
ng add @angular-architects/native-federation --project  
<<nombre_del_proyecto>> --port <<numero_del_puerto>> --type dynamic-  
host
```

Por ejemplo:



```
ng add @angular-architects/native-federation --project siscom-gesusua-app-ui --port 4200 --type dynamic-host
```

2. Configurar el archivo federation.manifest.json

La instalación de la librería native-federation agrega el archivo federation.manifest.json en este archivo se debe configurar la ruta del remoteEntry el microfrontend:



```
{  
  "mfeSGU": "https://gestionusuario.osce.gob.pe/perfil/remoteEntry.json"  
}
```

Nota: La ruta para QA será:

```
"mfeSGU": "https://qa-gestionusuario.osce.gob.pe/perfil/remoteEntry.json"
```

3. Configurar federation.config.js

Abrir el archivo **federation.config.js** y realizar los siguientes cambios:

En el objeto **shared** agregar las propiedades:

```
"commander": { singleton: true, requiredVersion: false },  
"fs": false,  
"path": false,  
"util": false,  
"child_process": false
```

En el arreglo **skip** verificar que no se encuentre el valor **'rxjs/webSocket'** (eliminarlo en caso esté presente).

El archivo debe quedar como así:

```
const { withNativeFederation, shareAll } = require('@angular-architects/native-federation/config');

module.exports = withNativeFederation({
  shared: {
    ...shareAll({ singleton: true, strictVersion: true, requiredVersion: 'auto' }),
    "commander": { singleton: true, requiredVersion: false },
    "fs": false,
    "path": false,
    "util": false,
    "child_process": false
  },
  skip: [
    'rxjs/ajax',
    'rxjs/fetch',
    'rxjs/testing',
    // Add further packages you don't need at runtime
  ]
});
```

4. Instalar librería compartida:

Instalar el paquete “osce-sgu-shared” (versión 0.0.27 al 09/04/2025):

```
npm i osce-sgu-shared@latest
```

5. Uso de páginas del microfrontend

En la configuración de rutas importar **loadRemoteModule**:

```
import { loadRemoteModule } from '@angular-architects/native-federation';
```

Luego se configura la ruta del componente del microfrontend:

Ejemplo para las páginas de Información Personal y Actualizar Contraseña:

```
export const routes: Routes = [
  ...,
  {
    path: '',
    canMatch: [authGuard],
    loadComponent: () => import('./core/layouts/default/default.component'),
    children: [
      ...,
      {
        path: 'informacion-personal',
        loadComponent: () => loadRemoteModule('mfeSGU', './InformacionPersonal')
          .then(m => m.InformacionPersonalComponent),
        data: {
          breadcrumb: [
            { label: 'Información Personal' }
          ]
        }
      },
      {
        path: 'cambiar-contrasena',
        loadComponent: () => loadRemoteModule('mfeSGU', './CambiarContrasenia')
          .then(m => m.CambiarContraseniaComponent),
        data: {
          breadcrumb: [
            { label: 'Actualizar Contraseña' }
          ]
        }
      }
    ]
  }
];
```

6. Establecer parámetros del sistema:

En el componente principal se establece los parámetros generales del sistema como el título a mostrar, página de inicio y las llaves para el consumo de servicios de catálogos comunes:

```
ngOnInit() {
  this.sguSharedService.setNombreSistema(
    'Sistema de Gestión de Catálogos', // Título para la cabecera
    'SGC' // Nombre corto para pantallas pequeñas (celular)
  );

  this.sguSharedService.setDatosSistema(
    '/home', // ruta de la página de inicio del sistema
    'ws://gestionusuario.osce.gob.pe/notificaciones' // url del web socket de notificaciones (solo si tiene)
  );

  // llaves para el consumo de apis de catálogos comunes
  this.sguSharedService.setApiKeysCatalogos(
    environment.publicKey,
    environment.apiKey
  );
}
```

Se espera que los mensajes enviados por este websocket tengan la siguiente estructura (pudiendo ser también un array de este tipo):

```
export type TipoNotificacion = 'info' | 'warning' | 'error' | 'success' | 'task';

export interface INotificacion {
  id?: number;
  idUsuario?: number;
  tipo?: TipoNotificacion;
  titulo?: string;
  mensaje?: string;
  fecha?: Date;
  leida?: boolean;
  urlDestino?: string; // Ruta interna en el sistema. Por ejemplo: /perfil
  esUrlExterna?: boolean; // Ruta externa. Por ejemplo: https://www.gob.pe
  datos?: any;
}
```

7. Uso del header del microfrontend y configuración del cierre de sesión

En el layout agregar un **ng-container**, por ejemplo:

```

<div class="flex flex-col min-h-screen">
  <ng-container #cabecera></ng-container>
  <div class="flex-grow flex items-center bg-alternative">
    <div class="gob-container py-5">
      <router-outlet />
    </div>
  </div>
  <app-footer />
</div>

```

En el typescript importar loadRemoteModule y SguSharedService:

```

import { loadRemoteModule } from '@angular-architects/native-federation';
import { SguSharedService } from 'sgu-shared';

```

Cuando se hace click en la opción Cerrar Sesión desde la cabecera del microfrontend se lanza un evento, en el cual cada sistema debe realizar el borrado de sus datos de sesión y luego permitir finalizar la sesión, para esto en el constructor se configura el subscribe del observable **cerrarSesionClick\$**.

A la función **finalizarSesion** del servicio `sguSharedService` se le pasa como primer parámetro el `clientId` del sistema host y como segundo parámetro la URL a donde se debe redirigir después de terminar el proceso de cerrar sesión

La carga del componente **Header** se realiza en la función **ngOnInit**, además se establece el **token** de sesión para que se muestren los datos del usuario en el header:


```

export class DefaultLayout implements OnInit {
  mfeCabecera = viewChild.required('cabecera', { read: ViewContainerRef });

  constructor(
    private sessionService: SessionService,
    private sguSharedService: SguSharedService,
  ) {
    this.sguSharedService.cerrarSesionClick$
      .pipe(takeUntilDestroyed())
      .subscribe(() => {
        // si se necesita validar algo antes de cerrar la sesión,
        // se podrían realizar aquí y según eso determinar si se cierra la sesión o no

        this.sessionService.borrarSesion(); // borrado de los datos propios de cada aplicación

        // Se envía como parámetro la URL a donde debe redireccionar después de cerrar sesión
        this.sguSharedService.finalizarSesion('client-front', 'https://gestionusuario.osce.gob.pe');
      });
  }

  async ngOnInit() {
    const { CabeceraComponent } = await loadRemoteModule('mfeSGU', './Cabecera');
    this.mfeCabecera().createComponent(CabeceraComponent);

    // Si está autenticado, se envía el token al microfrontend. Cada vez que se cambia el
    // token se debe volver a enviar el token al microfrontend (por ejemplo en el refreshToken)
    if (this.sessionService.Token) {
      this.sguSharedService.setToken(
        this.sessionService.Token,
        this.sessionService.RefreshToken
      );
    }
  }
}

```

8. Verificar la Integración

- Ejecuta tu aplicación (ng serve) y verifica que se carguen los componentes.