# Scaffolding

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Versión

**Normativa front guia implementacion Scaffolding Vigente**

**Versión:** 1.0.0

**Estado**: **ACTIVO**

**Entrada en vigor desde: 1 jun 2023**

**Obligado cumplimiento desde: Sept-2023**

**Normativa front guia implementacion Scaffolding PRE-RELEASE**

**Versión:** 1.0.0

**Estado**: RELEASE

**Entrada en vigor desde: 28 nov 2022**

**Obligado cumplimiento desde:** -

## *Tabla de Contenido*

## *Cumplimiento Normativo*

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

**Contacto Arquitectura:** l-arquitectura.stic@juntadeandalucia.es

## *Histórico de cambios*

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabeceras de la tablas dónde se indican las fechas de entrada en vigor y versión.

| Versión | Pre-release | Adopción | Activa | Retiro | Alcance |
|---|---|---|---|---|---|
| v01r00 | 15 jun 2022 | | | | • Versión inicial del documento |

# 1. Introducción

Este proyecto es un command line interface (cli) para node que pretende facilitar la creación de proyectos para el uso de web components con Lit y TypeScript. La estructura del proyecto, así como la de sus elementos, son acordes a la normativa arquitectónica definida por las STIC.

# 2. Uso

Una vez descargado el proyecto, desde la raíz del mismo ejecutamos por consola:

```
foo@bar:~$ sas-cli <comando> [opción]
```

| Contexto | Acción | Comando | Opción | Descripción |
|---|---|---|---|---|
| app | | | | Todas las acciones relativas a un proyecto de tipo aplicación |
| | create | *app:create* | | creación de un proyecto de tipo aplicación |
| | clean | *app:clean* | | limpia elementos temporales de la construcción de la aplicación (out-ts, dist,back-up y coverage ) |
| | | | *-a --all* | limpia todo elemento que no sea parte del código fuente (elementos temporales, node_modules, package-lock.json) |
| monorepo | | | | Todas las acciones relativas a un proyecto de tipo monorepo. Este tipo de proyecto es el ideal para el desarrollo de componentes reutilizables y publicables (web-componentes, librerías) |
| | create | *monorepo: create* | | Creación de un proyecto de tipo monorepo |
| | init | *monorepo: init* | | Inicializa un monorepo existente creando los enlaces y las carpetas necesarias para los componentes existentes |
| | clean | *monorepo: clean* | | Limpia elementos temporales de la generación del monorepo (types, dist,back-up y coverage ). También elimina los elementos temporales para los componentes creados en el proyecto |
| | | | *-a --all* | Limpia todo elemento que no sea parte del código fuente (elementos temporales, node_modules, package-lock.json). También elimina todo elemento que no sea parte del código fuente para los componentes creados en el proyecto |
| WebComponent | | | | Todas las acciones relativas a un componente de tipo webcomponent |
| | create | *wc:create* | | Creación de un componente tipo webcomponent |

| | | | | |
|---|---|---|---|---|
| | | | *-d --dir* | Crea el web componente en una carpeta concreta dentro de la carpeta package del monorepo. Admite profundidad (carpeta1/carpeta2/carpeta3 ... ) |
| Librería | | | | Todas las acciones relativas a un componente de tipo librería |
| | create | *lib:create* | | Creación de un componente tipo librería |
| | | | *-d --dir* | Crea la librería en una carpeta concreta dentro de la carpeta package del monorepo. Admite profundidad (carpeta1/carpeta2/carpeta3 ... ) |

# 3. Proyectos

En esta sección se van a abordar los siguientes tipos de proyectos que actualmente se proporcionan por sas-cli: **monorepo y aplicación**

### 3.1 Monorepo

El proyecto está configurado para ser ejecutado como un monorepo, haciendo uso de la opción workspace de npm a partir de la versión ***7.X***. Puede albergar un catálogo de componentes reutilizables (WebComponents, librerías)

```
foo@bar:~$ sas-cli monorepo:<acción> [options]
```

## 3.1.1 Generación del monorepo

```
foo@bar:~$ sas-cli monorepo:create
```

- Este comando creará un proyecto de tipo monorepo sobre una carpeta que no contenga un proyecto creado con anterioridad.
- Se creará la estructura esqueleto del proyecto
- Se pedirá información básica para la construcción del monorepo
- Se generarán los ficheros con la información recopilada

Esta opción generará un fichero ***(.sas-project.config)*** donde se almacenará dicha información de configuración

## 3.1.2 Inicialización del monorepo

```
foo@bar:~$ sas-cli monorepo:init
```

- Inicializa el proyecto creando los links necesarios con los componentes existentes.

## 3.1.3 Acciones del monorepo

Una vez creado el monorepo se proporciona los siguientes scripts:

#### Generación de un distribuible

Esta opción es la que ejecutará el **ALM del SAS** por defecto en la entrega del proyecto. Incluye los transpilados *(.js)*, los types *(d.ts)* y los mappers *(.js.map)*

```
foo@bar:~$ npm run build
```

### Generación del proyecto para entornos de desarrollo

Opción ideal para la versión de **desarrollo** y al incluir un nuevo elemento en el proyecto (ej. un nuevo componente). Excluye los types de la carpeta de dist permitiendo hacer un *hot-reload*.

```
foo@bar:~$ npm run build:dev
```

### Limpieza y generación del proyecto de entornos de desarrollo

Opción ideal para cuando se quiera realizar una reconfiguración del proyecto en desarrollo. Se recomienda su ejecución **antes de hacer el entregable** para asegurarse que no hay ningún problema con una construcción limpia (como se realizará en el ALM del SAS)

```
foo@bar:~$ npm run build:clean:dev
```

### Lanzar storybook del monorepo

Si se quiere consultar el catálogo de componentes desarrollado en el proyecto, se debe ejecutar:

```
foo@bar:~$ npm run storybook
```

Si se quiere generar un distributable para desplegar en un servidor de estáticos (nginx, apache), ejecutar:

```
foo@bar:~$ npm run storybook:build
```

Para más detalle sobre el storybook revisar esta sección

### Ejecutar Test del monorepo

Si se quieren lanzar los test con su análisis de cobertura de los test realizados para todo el catálogo de componentes desarrollado en el proyecto se debe ejecutar:

```
foo@bar:~$ npm run test:dev
```

Para más detalle sobre el testing revisar esta sección

### Generación del proyecto con la instalación y transpilación en modo debug

Opción para cuando se necesita información más detallada tanto en la compilación como en la transpilación

```
foo@bar:~$ npm run build:monorepo:debug
```

### Cambio de versión

Para facilitar el cambio de las versiones de todos los componentes del proyecto, y mantener la coherencia del versionado, se proporcionan dos scripts de npm *(uno para windows y otro para linux)* para ejecutar

- ***LINUX***

```
foo@bar:~$ npm run version:set:linux --new_version=1.0.6
```

- ***WINDOWS***

```
foo@bar:~$ npm run version:set:windows --new_version=1.0.6
```

### 3.2 Aplicación

El proyecto está configurado para ser ejecutado a partir de la versión ***7.X*** de npm.

## 3.2.1 Generación de la app

La estructura de la aplicación generada mediante este CLI, siguen las directrices marcadas por el área de arquitectura de la STIC. La base tecnológica es Lit (2.X) y TypeScript (4.x)

```
foo@bar:~$ sas-cli app:create
```

- Este comando creará un proyecto de tipo aplicación sobre una carpeta que no contenga un proyecto creado con anterioridad.
- Se creará la estructura esqueleto del proyecto
- Se pedirá información básica para la construcción de la aplicación
- Se generarán los ficheros con la información recopilada

Esta opción generará un fichero ***(.sas-app.config)*** donde se almacenará dicha información de configuración

## 3.2.2 Construir la app

Para construir la app hay que añadir al package.json del proyecto, en la sección script, lo siguiente, substituyendo <nombre_app> por su valor (app-<area_desarrollo>-<nombre_app>)

```
foo@bar:~$ npm run start:build
```

## 3.2.3 Lanzar la app

```
foo@bar:~$ npm run start
```

Notas técnicas

- Para más información sobre el servidor de desarrollo de la aplicación propuesto revisar esta sección
- Si se quiere que la app cargue el theme:
    - **Añadir la dependencia del theme** correspondiente al **package.json** del componente
    - **Cargar en el index.html**. Se recomienda revisar la propuesta hecha por la plantilla de la aplicación. Esta plantilla está basada en la forma de implementar el theme de la stic.

- Polyfills. En caso de que la aplicación deba ser compatible con IE11 se cargarán los polyfills necesarios en el index.html de la aplicación

### 3.2.4 Limpiar la app

Limpieza de todos los elementos generados durante la construcción de la aplicación (out-ts, dist,back-up y coverage )

```
foo@bar:~$ sas-cli app:clean
```

Limpieza completa de todos los elementos generados durante la construcción e instalación de la aplicación (temporales, node_modules)

```
foo@bar:~$ sas-cli app:clean -a
```

# 4. Componentes

En esta sección se van a abordar los dos tipos de proyectos que actualmente se proporcionan por sas-cli, monorepo y aplicación

#### 4.1 WebComponent

Esta sección abordará las acciones que se pueden realizar sobre los componentes de tipo Web.

### 4.1.1 Generación de WebComponents

La estructura de los componentes generados mediante este CLI, siguen las directrices marcadas por el área de arquitectura de la STIC. La base tecnológica es Lit (2.X) y TypeScript (4.x).

Se generará un componente con la siguiente estructura en su nombre ***wc-<area_desarrollo>-<nombre_componente>***.

```
foo@bar:~$ sas-cli wc:create <nombre_componente>
```

El componente se creará en la carpeta packages (la determinada para almacenar el catálogo de componentes reutilizables).

Después de la creación de los componentes necesarios, para la correcta configuración del monorepo, se recomienda la ejecución del comando.

```
foo@bar:~$ npm run build:dev
```

### 4.1.2 Lanzar una demo del WebComponent

Si se quiere montar una prueba de concepto de algún componente se puede hacer uso de la carpeta demo.

Para lanzarla hay que añadir al **package.json del proyecto**, en la sección **script**, lo siguiente, sustituyendo ***<nombre_componente>*** por su valor ***(Wc-<area_desarrollo>-<nombre_componente>)***

```
"start:<nombre_componente>": "concurrently --kill-others --names tsc,web-dev-server \"npm run tsc:dev:
watch\" \"web-dev-server --app-index packages/<nombre_componente>demo/index.html --node-resolve --open --
watch\""
```

Una vez añadido el script se debe ejecutar, sustituyendo ***<nombre_componente>*** por su valor ***(Wc-<area_desarrollo>-<nombre_componente>)***, con

```
foo@bar:~$ npm run start:<nombre_componente>
```

Notas técnicas para arrancar una demo sobre web components

- Para más información sobre el servidor de lanzamiento de la demo propuesto revisar esta sección
- Si se quiere que el componente cargue el theme durante la ejecución de la demo:
    - **Añadir la dependencia del theme** correspondiente al **package.json** del componente
    - **Cargar en el index.html**. Se recomienda revisar la propuesta hecha por la plantilla de los componentes. Esta plantilla está basada en la forma de implementar el theme de la stic.

### 4.2 Librería

Esta sección abordará las acciones que se pueden realizar sobre los componentes de tipo Librería.

## 4.2.1 Generación de Librería

La estructura de los componentes generados mediante este CLI, siguen las directrices marcadas por el área de arquitectura de la STIC. La base tecnológica es Lit (2.X) y TypeScript (4.x).

Se generará una librería con la siguiente estructura en su nombre ***lib-<area_desarrollo>-<nombre_componente>***.

```
foo@bar:~$ sas-cli lib:create <nombre_componente>
```

El componente se creará en la carpeta packages (la determinada para almacenar el catálogo de librerías reutilizables).

Después de la creación de los componentes necesarios, para la correcta configuración del monorepo, se recomienda la ejecución del comando.

```
foo@bar:~$ npm run build:dev
```

# 5. Técnico

### 5.1 Server

- Por la técnica utilizada para la construcción del monorepo (a través de symlinks) la opción que se elija deben tener en cuenta y activada la opción **preserveSymlinks**
- Los navegadores no son compatibles con "bare" modules. Para solventar esto hay dos formas:
    - Uso web-dev-server, que, entre otras cosas, permite la transpilación al vuelo y substituye para el navegador los "bare" modules por la carga de módulos compatible para el navegador.

- Uso de herramientas de transpilación y bundling como Webpack o rollup

> *Nota: Por facilidad de configuración se propone y recomiendo el uso del web-dev-server*

## Configuración

- El fichero **web-dev-server.config.mjs** es el que contiene la configuración mínima propuesta.
- Para ampliar la configuración consultar la información

### 5.2 Storybook

El proyecto en desarrollo se basa en el storybook de ModernWeb. Se ha elegido esta versión customizada del storybook por ser más ligera de cara al desarrollo. Este es un storybook basado en rollup y que limita algunas de las capacidades del storybook original.

Los componentes deben poder desplegarse en cualquiera de los dos storybook

## Configuración

Las configuraciones aplicadas están basadas en el storybook original para WebComponents. Dichas configuraciones se encuentran en la carpeta raíz_**.storybook**_.

Algunas de las configuraciones extras que se pueden aplicar:

- **Exclusión de componentes**: en el fichero ***main.js*** se pueden añadir al array la lista de componentes sin el ***wc-***, tal y como se muestra en el ejemplo

```
const excludeWCList = [
  "stic-theme",
  "stic-button",
]
```

- **Carga del theming en el storybook**: Si el theming de la aplicación se puede realizar de forma generalizada para todos los componentes en el fichero ***preview-head.html***. El ejemplo muestra como se actuaría en caso de seguir la estructura del theming propuesto por la stic. Al crear un nuevo proyecto, se necesita descomentar el código y adaptarlo al theme del área o aplicación correspondiente

```
<script type="module">
    import { STICTheme } from '@sas/wc-stic-theme';
    new STICTheme().loadHeadStyles();
</script>
```

Por la técnica utilizada para la construcción del monorepo (a través de symlinks) la opción que se elija debe tener en cuenta y activada la opción **preserveSymlinks**.

Si se utiliza el storybook proporcionado por ModernWeb, se debe tener en cuenta que utiliza web-dev-server como servidor de arranque por lo que dicho servidor debe tener la configuración adecuada. Para más información revisar la sección

### 5.3 Testing

Las plantillas del sas-cli en materia de testing están adaptadas a los recursos y configuración proporcionados por ModernWeb. Es recomendable, pero no obligatorio, seguir el stack tecnológico propuesto por ModernWeb. En caso de tener una propuesta alternativa se recomienda consultar con el departamento de arquitectura del sas

Cualquiera definición o stack tecnológico debe ser posible ejecutarlo dentro del ALM del SAS. Se recomienda validar la ejecución de pruebas en un entorno contenerizado.

## Configuración

- Por la técnica utilizada para la construcción del monorepo (a través de symlinks) la opción que se elijan debe tener en cuenta y activada la opción **preserveSymlinks**.
- El fichero **web-test-runner.config.mjs** es el que contiene la configuración mínima propuesta.
- Para ampliar la configuración consultar esta información

# 6. Actualización del proyecto

En esta sección se describe una guía básica para actualizar el proyecto a la versión actual del sas-cli.

### 6.1 Migración del proyecto con scaffolding a sas-cli 0.X.X

Hay que seguir los siguientes pasos para realización de la migración:

## Migración Monorepo

Se describen a continuación los pasos a dar para la migración del monorepo

#### Borrado de elementos del Monorepo

- carpeta **scaffolding**
- ficheros **configure.sh**, **sas-cli.js** y **package-lock.json**
- Opcional: carpetas **demo** y **playground**

## Web Component del Monorepo

#### Borrado de elementos del web component

- en la raíz del componente ficheros **tsconfig.json**, **web-dev-server.config.mjs** y **web-test-runner.config.mjs**
- en la raíz del componente un fichero con el siguiente patrón **<wc-area-nombrecomponente.ts>**. En caso de tener contenido se debe copiar al fichero index.ts localizado también en la raíz del componente
- la carpeta **.storybook**
- el fichero **.gitignore**, **.editorconfig**
- las devDependencies del **package.json**
- Aquellos componentes que sean librerías (que no tienen componentes visuales), eliminar la carpeta **stories** y el fichero **custom-element-json**

Recomendaciones migración de web component

- En caso de no necesitar crear ningún componente nuevo, se recomienda crear un ficticio que permita comprobar los cambios propuestos y adecuar los componentes actuales a dichos cambios.
- Revisar el **demo/index.html** para asegurarse de que las importaciones y el código se ajusta a los cambios una vez realizada la migración. Compararlo con el fichero propuesto. Verificar la carga del theme basándose en la propuesta y modificarla acorde al theme creado para el área o aplicación correspondiente.
- Revisar el código de las stories y adecuarlos a los cambios propuestos.

Migración de Lit 1.x a Lit 2.X

- La plantilla está construida sobre la base de Lit 2.X
- Es recomendable, no indispensable, que antes de hacer la migración se actualicen los componentes (src, testing y story) a la versión de Lit 2.x.
- Guía de actualización.

Notas

- Pueden convivir componentes en diferentes versiones de Lit, aunque no es lo más recomendable.
- Revisar, haciendo una búsqueda, los import del proyecto. Puede ocurrir que se esté apuntando a algún módulo de Lit 1.x y generar algún conflicto o error inesperado al compilarlo en el ALM o publicarlo en los entornos del SAS.

## Aplicación del Monorepo

Borrado de elementos de la aplicación

- La carpeta **.storybook**
- El fichero **custom_element.json** y **package-lock.json**

Recomendaciones migración de la aplicación

- En caso de disponer de una aplicación ya creada, se recomienda crear una ficticia que permita comprobar los cambios propuestos y adecuar los componentes actuales a dichos cambios.
- Revisar el **app/index.html** para asegurarse de que las importaciones y el código se ajusta a los cambios una vez realizada la migración. Compararlo con el fichero propuesto.
- Revisar la carga del theme basándose en la propuesta.

### 6.2 Actualización a sas-cli

- Instalación del sas-cli
- **Monorepo de componentes**
    - Limpieza del Monorepo
    - Inicializar del monorepo.
    - Lanzar storybook
- **Aplicación**
    - Construcción de la app

- Lanzar app

## 7. Lecciones aprendidas

- Al construir el monorepo, si se genera un node modules en dentro de un componente esto significa que hay alguna versión de alguno de las dependencias o devDependencies que no está alineada. Es recomendable alinear las versiones para evitar posibles conflictos o errores inesperados

## 8. Diagrama de uso