7-SSR

Server Side Rendering

```
# 1. Angular Universal
# 2. Despliegue con Node Express
# 3. Variantes: shell y pre-rendering
```

```
As a: web user,
  I want: get content ready
  so that: I can interact asap

As a: sop spider robot,
  I want: get content inn html
  so that: I can index it
```

1 Angular Universal

1.1 Vuelta al servidor

> Angular nació para vivir en el navegador.

- Para **quitarle carga al servidor** generando el contenido dinámico en el navegador en base a plantillas.
- Para enviar por la red primero la aplicación y después los datos, **ahorrando transferencia** durante un uso continuado.
- Para mejorar la **experiencia del usuario** al no percibir recarga de página durante la navegación dentro de la aplicación.

Ideal en entornos de intranet o aplicaciones de gestión de uso intensivo.

Problemático para uso esporádico o indexable públicamente.

1.2 Para mejorar el SEO

> El contenido se genera durante la ejecución del JavaScript en el navegador.

- Los **robots** no tienen nada significativo que indexar.
- Las **redes sociales** no encuentran cabeceras para mejorar la presentación de enlaces.

Hay que enviar el contenido ya generado.

Pero sin perder la experiencia de usuario durante la ejecución.

1.3 Para mejorar la experiencia en la primera visita

> Para mostrar contenido antes hay que descargar y ejecutar la aplicación.

- Los **usuarios** ven una página vacía demasiado tiempo.
- El **peso de la descarga inicial** es desproporcionado a pesar de *lazy loading*.

Hay que enviar el contenido ya generado.

Descargar la aplicación en segundo plano.

2 Despliegue con Node Express

2.1 Add Express Engine

```
ng add @nguniversal/express-engine --clientProject shop
```

2.2 Scripts de compilado y despliegue

```
{
  "start:ssr": "npm run build:ssr && npm run serve:ssr",
  "build:ssr": "npm run build:client-and-server-bundles && npm run
compile:server",
  "build:client-and-server-bundles": "ng build --prod && ng run
shop:server:production --bundleDependencies all",
  "compile:server": "webpack --config webpack.server.config.js --progress --
colors",
  "serve:ssr": "node dist/server"
}
```

2.3 Control de rutas

```
http://localhost:4000
http://localhost:4000/rates
```

3 Variantes: shell y pre-rendering

3.1 Shell para mejora de experiencia inicial

- Muestra un contenido instantáneo mientras descarga la app.

- Mejora la experiencia de usuario en la primera visita.
- De cara al SEO, sólo indexa el contenido inicial.

> Adecuado para aplicaciones de usuario registrado, pero con un portal de bienvenida indexable y rápido.

https://next.angular.io/guide/app-shell

```
ng g application admin --inlineStyle --inlineTemplate --routing --prefix=ab-admin
--skipTests --no-interactive
ng g appShell --clientProject=admin --universalProject=uadmin --dry-run --no-
interactive
```

3.2 Pre renderizado de toda la aplicación

- Se trata de volver no sólo al servidor web, si no al servidor de ficheros.
- La idea es invocar repetidamente al SSR y almacenar el HTML resultante en ficheros físicos.
- En producción, los robots y los usuarios recibirán ya esas copias pre generadas.
- Para mantener el sistema actualizado se necesita regenerar frecuentemente los ficheros

> Adecuado para blogs y sitios que no puedan o no quieran tener un servidor web corriendo.

angular Prerender

> **Blog de apoyo:** Velocidad y SEO con el SSR de Angular Universal
>
> > By Alberto Basalo

> Next:

Internacionalización y puesta en producción

Traducciones

Adaptaciones culturales de tiempo y moneda

Otras consideraciones para aplicaciones en producción.