Desarrollando un CRUD en ANGULAR

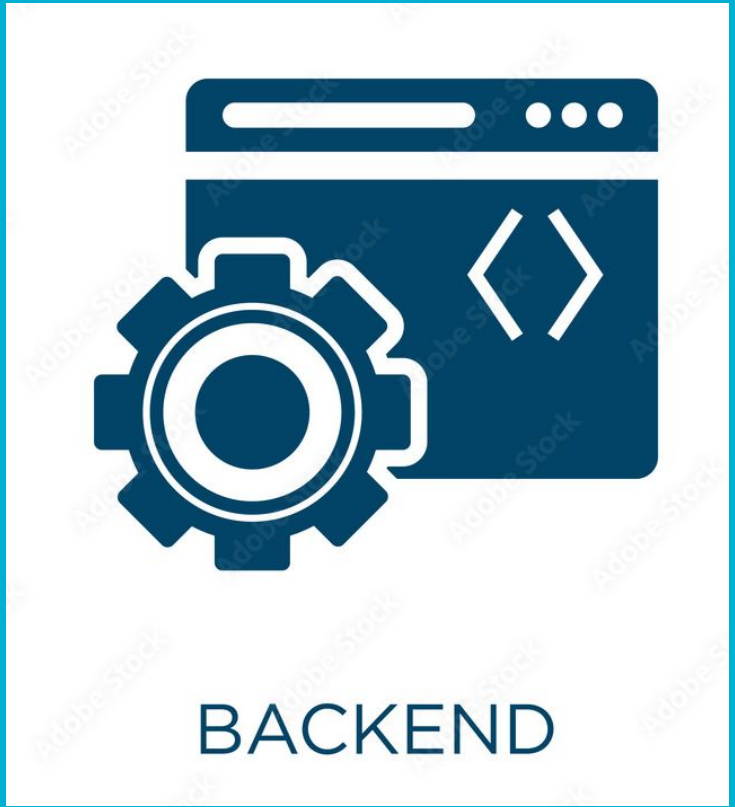
Parte 6
Service Workers

Mct. Esteban Calabria

Enunciado

En el laboratorio pasado hicimos un CRUD en angular que funciona con un backend.

Vamos a darle la posibilidad que funcione offline con un service worker



Service Worker

Los service workers son scripts que el navegador ejecuta en segundo plano, separados de la página web, permitiendo funcionalidades como el caché avanzado, las notificaciones push y el soporte para aplicaciones web sin conexión.

Actúan como un proxy entre la red y la aplicación, mejorando el rendimiento y la experiencia del usuario al proporcionar contenido incluso cuando no hay conexión a Internet.

Agregar un Service Worker en Angular

- Instalar el paquete de Service Worker
- Configurar el archivo ngsw-config.json
- Registrar el Service Worker en la aplicación
- Construir la aplicación en modo de producción
- Servir la aplicación desde un servidor compatible con HTTPS

Instalar el paquete de Service Worker

En tu proyecto de Angular existente agregar el soporte para PWA y serviceworker en el paquete @angular/pwa

```
ng add @angular/pwa
```

ng add es un comando específico de Angular CLI que no solo instala un paquete, sino que también realiza configuraciones adicionales automáticamente en tu proyecto, como añadir archivos y modificar configuraciones necesarias, mientras que npm install solo instala el paquete sin hacer configuraciones adicionales.

Configurar el archivo ngsw-config.json

- Este archivo se genera automáticamente y define cómo el Service Worker debe comportarse.
- Puedes modificarlo para incluir reglas de caché y otras configuraciones personalizadas.


```
{
  "$schema": "./node_modules/@angular/service-worker/config/schema.json",
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/index.html",
          "/manifest.webmanifest",
          "/*.css",
          "/*.js"
        ]
      }
    },
    {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
          "/assets/**",
          "/media/*.(svg|cur|jpg|jpeg|png|apng|webp|avif)"
        ]
      }
    }
  ]
}
```

Registrar el Service Worker en la aplicación

Angular ya incluye el registro en el archivo app.config.ts si usaste ng add @angular/pwa.

Asegúrate de que el código siguiente esté presente:


```
import { ApplicationConfig, isDevMode } from
'@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideServiceWorker } from
'@angular/service-worker';

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes),
provideServiceWorker('ngsw-worker.js', {
        enabled: !isDevMode(),
        registrationStrategy:
'registerWhenStable:30000'
    })]
};
```

Construir la aplicación en modo de producción

El service worker no funciona en modo desarrollo porque interfiere con la recarga dinámica de angular.

- Para probarlo tenemos que utilizar la aplicación en modo producción
- Esto asegura que los archivos del Service Worker se incluyan en el paquete de producción.
- Genera una carpeta dist que contiene todos los archivos necesarios para desplegar la aplicación.

```
ng build --prod
```

Servir la aplicación desde un servidor HTTPS

Los Service Workers requieren HTTPS debido a su naturaleza de seguridad.

Puedes usar http-server o cualquier otro servidor web para servir tu aplicación localmente con HTTPS.

Esto abrirá la aplicación en tu navegador en la dirección http://localhost:8080.

Probar la funcionalidad offline con las herramientas de desarrollador

```
npm install -g http-server


cd dist/tu-nombre-de-aplicacion


http-server -o
```

Agregar un Service Worker Customizado

- Programar el Service Worker Custom
- Registrarlo en lugar del que viene por defecto

Service Worker Custom

El Service Worker que viene integrado con Angular es muy cómodo de usar y simplifica enormemente el proceso de añadir funcionalidades offline a nuestras aplicaciones.

Sin embargo, en ocasiones, queremos tener control absoluto sobre lo que hacemos en el SW.

Para esos casos, podemos registrar y escribir nuestro propio SW personalizado para manejar funcionalidades específicas según necesidad.

Programar el Service Worker Custom

Instalación:

- Durante la fase de instalación, el Service Worker abre una caché llamada "demo" y añade archivos específicos como /, /index.html, polyfills.js, y main.js a la caché.
- La instalación se completa con el método skipWaiting, asegurando que el nuevo Service Worker tome control inmediato.

Activación:

- Durante la activación, el Service Worker simplemente imprime un mensaje en la consola indicando que está activado.

Intercepción de solicitudes de red:

- El Service Worker intercepta todas las solicitudes de red.
- Si hay conexión a Internet, responde con una solicitud de red y almacena la respuesta en la caché.
- Si no hay conexión, busca en la caché una respuesta coincidente y la devuelve si la encuentra, o realiza una solicitud de red si no la encuentra.

```
const cacheName = "demo";

self.addEventListener('install', function (event) {
    console.log('Service Worker installing.');
    event.waitUntil(
        caches.open(cacheName).then(cache => {
            return cache.addAll([
                `/`,
                `/index.html`,
                'polyfills.js',
                "main.js",
            ])
                .then(() => self.skipWaiting());
        })
    );
});

self.addEventListener('activate', function (event) {
    console.log('Service Worker activating.');
});
```

Metodo Fetch

```
self.addEventListener('fetch', function (event) {
    console.log('Service Worker fetching. ' + event.request.url);

    if (navigator.onLine) {
        event.respondWith(
            fetch(event.request).then(response => {
                return caches.open(cacheName).then(cache => {
                    cache.put(event.request, response.clone());
                    return response;
                });
            })
        );
    }
```

Continua luego…

Metodo Fetch (Cont)

```
    if (!navigator.onLine) {

        event.respondWith(
            caches.open(cacheName)
                .then(cache => cache.match(event.request, { ignoreSearch: true }))
                .then(response => {
                    if (response) { console.log('Found ', event.request.url, ' in cache');}
                    return response || fetch(event.request);
                })
        );
    }
});
```

Registrarlo en lugar del que viene por defecto

Se realiza en el archivo app.config.ts

provideServiceWorker('...):

> Registra un Service Worker personalizado (sw.js), habilitado siempre, y con una estrategia de registro que espera a que la aplicación esté estable durante 30 segundos antes de registrar el Service Worker.

```
import { ApplicationConfig, isDevMode } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideServiceWorker } from '@angular/service-worker';

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes),
    //provideServiceWorker('ngsw-worker.js', {
      provideServiceWorker('sw.js', {
        //enabled: !isDevMode(),
        enabled: true,
        registrationStrategy: 'registerWhenStable:30000'
        //registrationStrategy : 'registerImmediately'
    })]
};
```

Sigamos Trabajando...