PWA

https://web.dev/learn/pwa/

https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

> https://github.com/suren-atoyan/react-pwa

https://www.npmjs.com/package/pwa-asset-generator

Service Worker

- Actúan como intermediarios entre nuestra aplicación web y el servidor del cual recibimos los datos.
- Son los encargados de ofrecernos una experiencia offline dentro de nuestras aplicaciones.
- Se trata de ficheros JS que no pueden acceder directamente al DOM ya que trabajan como un hilo aparte.
- No bloquean la UI.

Vite PWA

- Para poder instalar este plugin, lanzamos el comando

```
npm install -D vite-plugin-pwa
```

- Dentro del fichero de configuración de Vite modificamos la configuración del plugin

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import { VitePWA } from "vite-plugin-pwa";

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      // Opciones de autoupdate o prompt
      registerType: 'autoUpdate',
      // Comprobaciones sobre desarrollo. No funciona
      devOptions: { enabled: true },
      // includeAssets: ["**/*"],
      workbox: {
        // Incluimos todos los tipos de archivos assets fuera de
        globPatterns: ["**/*.{js,jsx,css,html,pdf,json}"],
        // Definimos los diferentes cachés en función de lo que
        runtimeCaching: [
          {
            // Se puede introducir una expresión regular
            urlPattern: ({ url }) => {
              return url.pathname.startsWith("/api/series");
            },
            // Cómo se maneja el cache
            handler: 'CacheFirst',
            options: {
              cacheName: 'api-cache-cambio1005',
              cacheableResponse: {
                statuses: [0, 200]
              }
            }
          }
        ]
      },
```

```
      manifest: {
        name: 'Series App',
        short_name: 'SeriesApp',
        description: 'Todas las series 10.05',
        icons: [
          {
            src: '/appIcon.png',
            sizes: '512x512',
            type: 'image/png',
            purpose: 'any'
          }
        ]
      }
    })
  ],
})
```

- La imagen la colocamos en la carpeta public.
- De esta manera, dentro del Chrome podemos instalar nuestra aplicación.

- Necesitamos colocar la siguiente configuración en vite-env.d.ts

```
/// <reference types="vite/client" />
/// <reference types="vite-plugin-pwa/client" />
```

- Además debemos registrar el SW dentro de **main.tsx**.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'
```

```
import { registerSW } from "virtual:pwa-register";

if ("serviceWorker" in navigator) {
  // && !/localhost/.test(window.location)) {
  registerSW();
}

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```