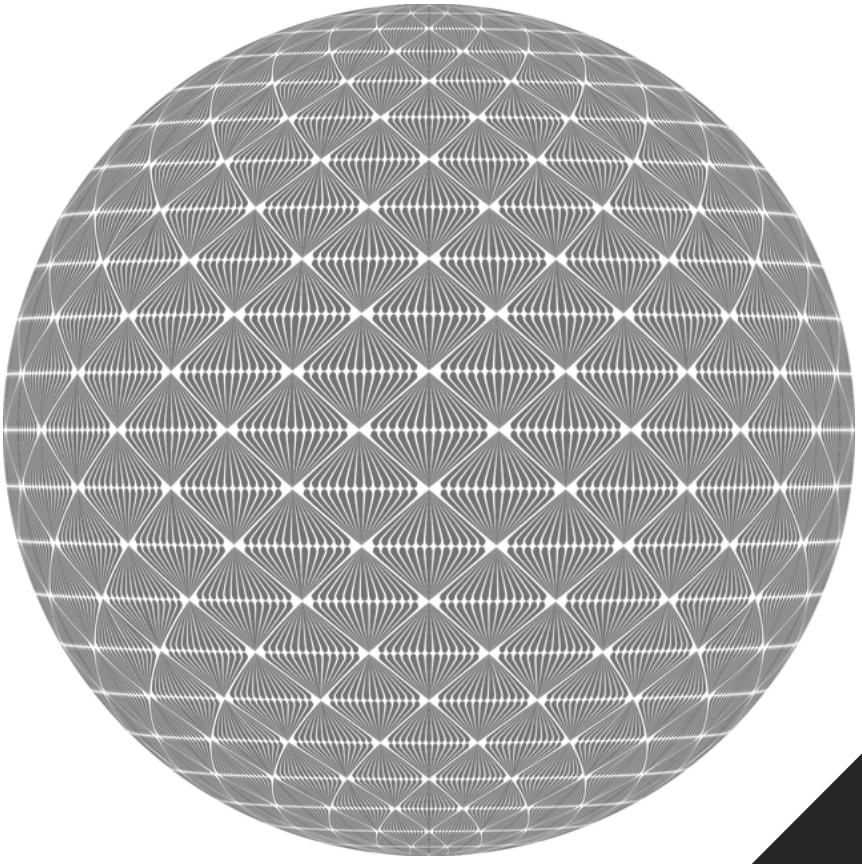


# APW

Rafael Aguayo Estrada
Gael Hernández Chávez
Ángel Jesús Reyes Huerta
Itzamara Mendoza Soto
Dianne García Vázquez

# Aplicaciones Web Progresivas (PWA)

## 1. Demostración práctica

Para comenzar, se realiza una demostración de una Aplicación Web Progresiva ya existente, con el objetivo de observar su funcionamiento real.

### 1.1 Apertura de una PWA conocida

Como ejemplo se utiliza Twitter Lite.

Abrir el navegador Google Chrome o Microsoft Edge.

Ingresar a la siguiente dirección web:

https://mobile.twitter.com

Al cargar el sitio, se puede observar que funciona como una aplicación optimizada para dispositivos móviles.

### 1.2 Instalación de la PWA

Una de las características principales de una PWA es que puede instalarse como una aplicación.

En la barra de direcciones del navegador aparece el ícono Instalar.

También se puede acceder desde el menú ⋮ y seleccionar la opción Instalar Twitter.

Una vez instalada, la aplicación se ejecuta de forma independiente, sin necesidad de abrir el navegador.

### 1.3 Ejecución en modo offline

Las PWA pueden funcionar sin conexión a internet.

Abrir las DevTools presionando la tecla F12.

Entrar a la pestaña Network.

Activar la opción Offline.

Recargar la aplicación.

Al realizar esta prueba, se observa que Twitter Lite sigue mostrando contenido básico, lo que demuestra que cuenta con funcionamiento offline.

### 1.4 Inspección de la PWA en DevTools

Para analizar cómo funciona internamente la PWA:

Abrir DevTools.

Entrar a la pestaña Application.

Revisar los siguientes elementos:

Manifest

Service Workers

Cache Storage

IndexedDB (si existe)

Estos elementos permiten comprender cómo la aplicación gestiona la instalación y el almacenamiento de datos.

## 2. Creación de una Aplicación Web Progresiva

Después de la demostración, se procede a crear una PWA desde cero.

### 2.1 Requisitos previos

Para desarrollar la PWA se necesita:

Google Chrome o Microsoft Edge

Visual Studio Code

Extensión Live Server

Una carpeta de trabajo vacía

## 3. Estructura del proyecto

La aplicación se organiza utilizando la siguiente estructura de archivos:

index.html

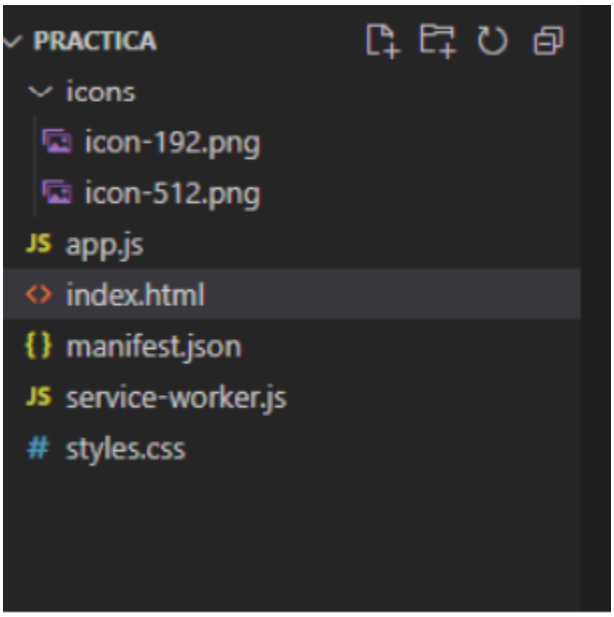
styles.css

app.js

manifest.json

service-worker.js

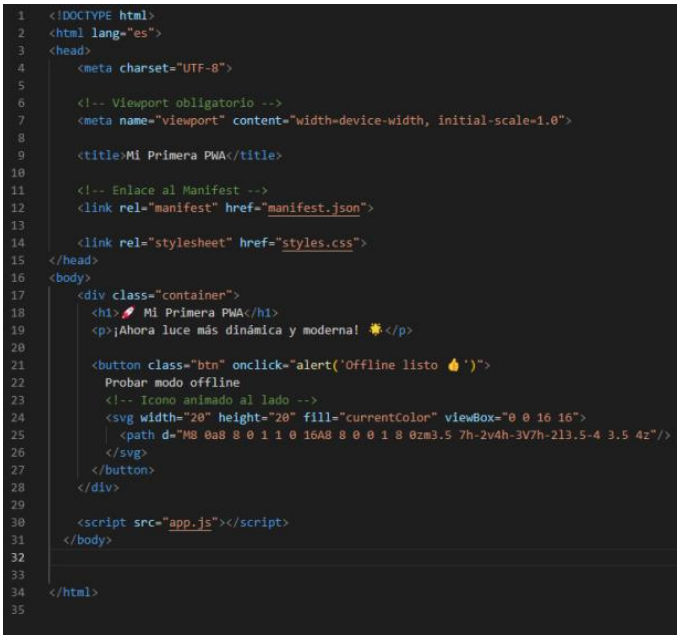
Esta estructura permite separar la interfaz, los estilos y la lógica de la aplicación.




## 4. Página principal (index.html)

El archivo index.html contiene la estructura principal de la aplicación y es el punto de entrada de la PWA.

También se enlazan los archivos de estilos, JavaScript y el archivo manifest.json.



```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">

    <!-- Viewport obligatorio -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Mi Primera PWA</title>

    <!-- Enlace al Manifest -->
    <link rel="manifest" href="manifest.json">

    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
      <h1>🚀 Mi Primera PWA</h1>
      <p>¡Ahora luce más dinámica y moderna! ✨</p>

      <button class="btn" onclick="alert('Offline listo 👍')">
        Probar modo offline
        <!-- Icono animado al lado -->
        <svg width="20" height="20" fill="currentColor" viewBox="0 0 16 16">
          <path d="M8 0a8 8 0 1 1 0 16A8 8 0 0 1 8 0zm3.5 7h-2v4h-3V7h-2l3.5-4 3.5 4z"/>
        </svg>
      </button>
    </div>

    <script src="app.js"></script>
  </body>


</html>

```

```css
/* Variables de color */
:root {
    --bg-gradient: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
    --card-bg: #ffffff;
    --text-color: #222;
    --primary-color: #2575fc;
    --secondary-color: #6a11cb;
    --btn-hover: #1b5bc1;
}

/* Reset básico */
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

/* Fondo dinámico */
body {
    font-family: "Poppins", sans-serif;
    background: var(--bg-gradient);
    color: var(--text-color);
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    padding: 20px;
    animation: backgroundShift 10s ease infinite;
}

/* Animación de fondo */
@keyframes backgroundShift {
    0% { background-position: 0% 50%; }
    50% { background-position: 100% 50%; }
    100% { background-position: 0% 50%; }
}

/* Tarjeta principal */
.container {
    background: var(--card-bg);
    padding: 35px 45px;
```

## 5. Archivo manifest.json

El archivo manifest.json es un archivo en formato JSON que permite definir cómo se comporta la PWA al instalarse.

Sus funciones principales son:

Definir el nombre de la aplicación

Configurar los iconos

Establecer colores y modo de visualización

Permitir que el navegador trate la web como una aplicación


## 6. Importancia del viewport

El meta viewport es fundamental para las PWA, ya que:

Permite que la aplicación sea responsive

Mejora la visualización en dispositivos móviles

Es un requisito para que la PWA pueda instalarse correctamente

```
<!-- Viewport obligatorio -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

## 7. Relación entre los componentes de una PWA

Una PWA está compuesta por diferentes elementos que trabajan en conjunto:

HTML: interfaz principal de la aplicación

Manifest: instalación y apariencia

Service Worker: funcionamiento offline y caché

JavaScript: registro y control de la aplicación

Sin el archivo HTML, la PWA no puede existir.

## 8. Registro del Service Worker

El archivo app.js se encarga de registrar el Service Worker, el cual permite:

Manejar la caché

Habilitar el modo offline

Mejorar el rendimiento de la aplicación

app.js > ...

```
1  if ('serviceWorker' in navigator) {
2      window.addEventListener('load', () => {
3          navigator.serviceWorker.register('service-worker.js')
4              .then(reg => console.log('Service Worker registrado'))
5              .catch(err => console.error('Error al registrar SW', err));
6      });
7  }
8
```

```
const CACHE_NAME = 'pwa-cache-v3';
const FILES_TO_CACHE = [
    '/',
    '/index.html',
    '/styles.css',
    '/manifest.json',
    '/app.js',
    '/icons/icon-192.png',
    '/icons/icon-512.png'
];

// Instalación
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open(CACHE_NAME)
            .then(cache => cache.addAll(FILES_TO_CACHE))
    );
});

// Activación
self.addEventListener('activate', event => {
    console.log('Service Worker activo');
});

// Fetch (modo offline)
self.addEventListener('fetch', event => {
    event.respondWith(
        caches.match(event.request)
            .then(response => response || fetch(event.request))
    );
});
```

## 9. Prueba Offline

1. Ejecuta el proyecto con Live Server
2. Abre DevTools → Network
3. Marca Offline
4. Recarga la página

**La PWA sigue funcionando**

```
Live reload enabled.                                    index.h
Service Worker registrado                                   ap
Service Worker activo                                   service-worker
```