Módulo 5
Desarrollo de aplicaciones Web Progresivas (PWA)

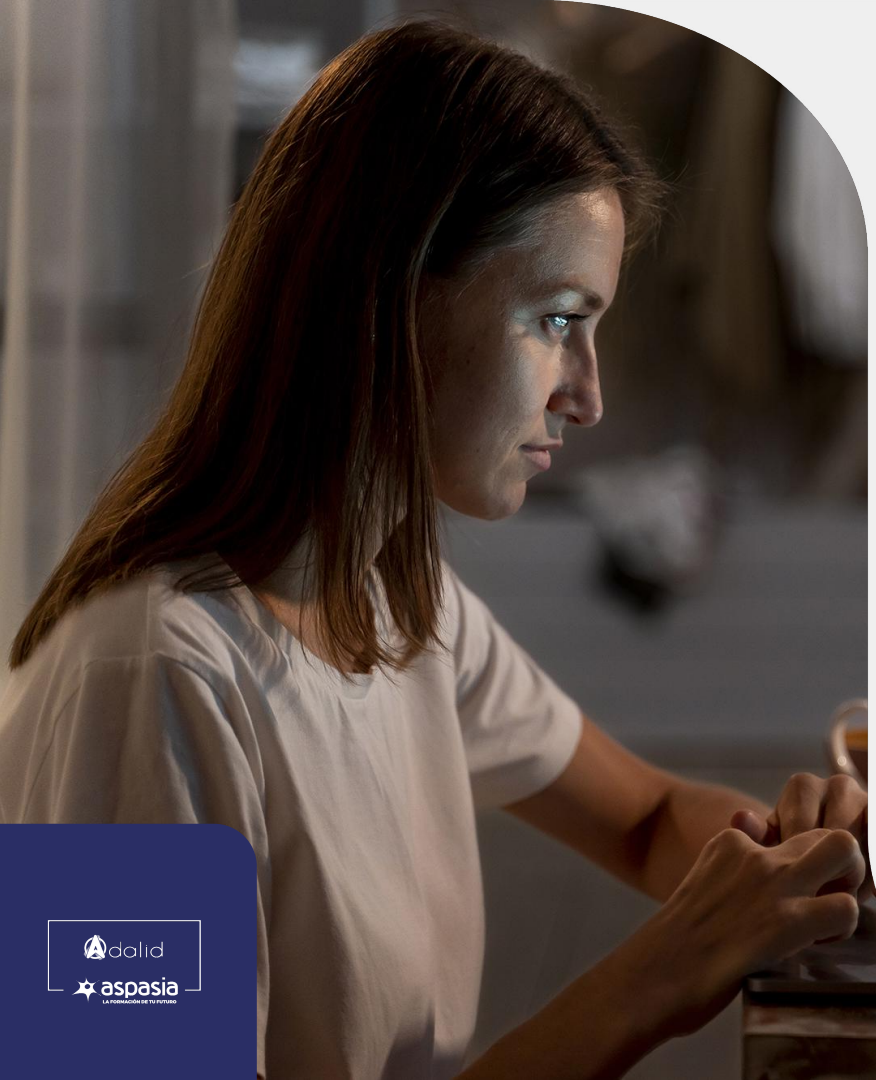
Introducción a las Aplicaciones PWA

Módulo 6

AE 1.2

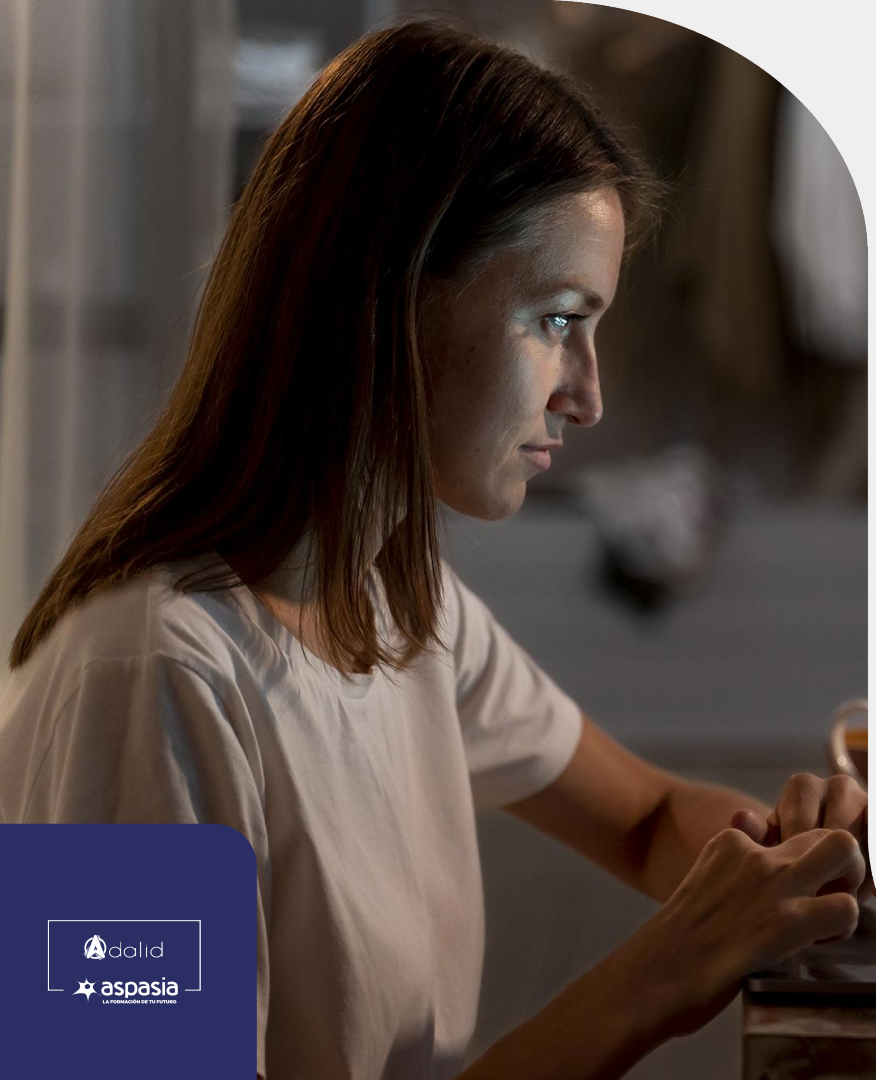
OBJETIVOS

Entender qué es una PWA, sus características, beneficios y limitaciones, aprender a configurar el Manifiesto y el Service Worker, gestionar la caché y la red con diferentes estrategias, y optimizar una aplicación React para funcionar como una PWA segura y eficiente.



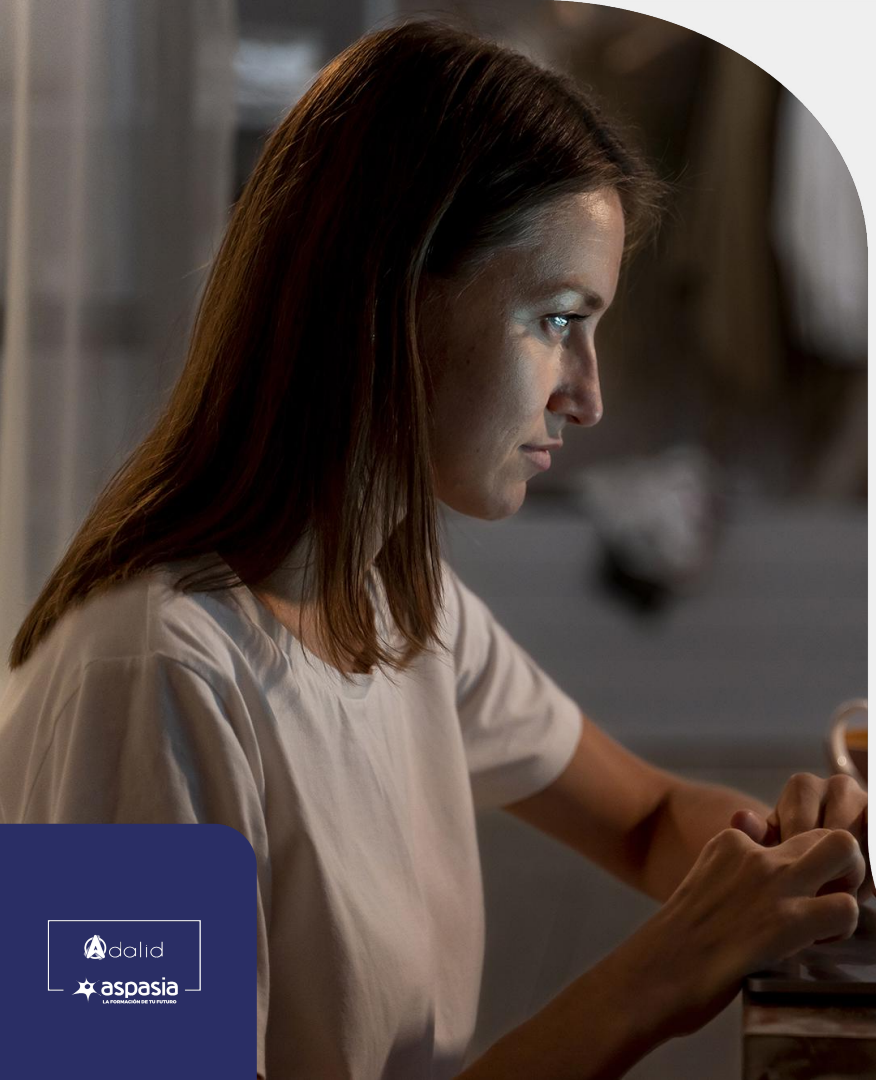
¿QUÉ VAMOS A VER?

- Introducción a las Aplicaciones PWA.
- Qué es unaPWA(Progressive Web Application).
- Características de una PWA (Progresiva,responsiva, adaptable, segura, independiente).
- Beneficios de una PWA.
- Limitaciones de una PWA.
- Diferencias entre una PWA, una aplicación Web tradicional y una aplicación Nativa.
- Arquitectura y componentes de una PWA (Service Workers, Manifiesto, Shell de la aplicación).



¿QUÉ VAMOS A VER?

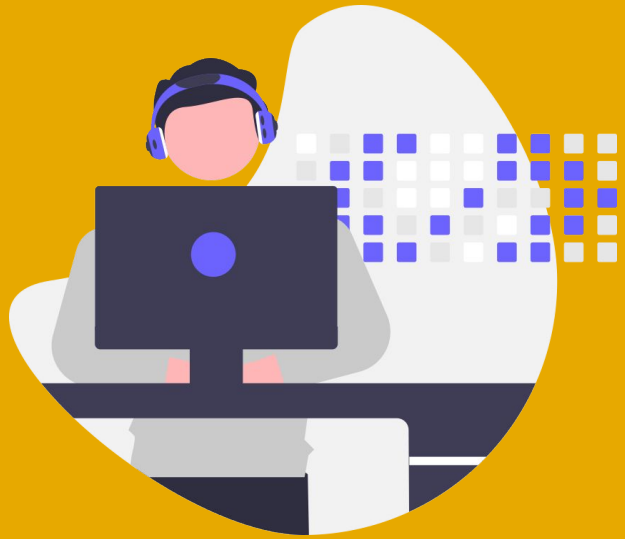
- Frameworks que soportan el desarrollo de PWA's.
- El Manifiesto.
- Qué es el Manifiesto.
- Para qué se usa el Manifiesto.
- Estructura de un archivo de Manifiesto.
- El Service Worker.
- Qué es el ServiceWorker.
- Para qué se usa el Service Worker.
- Ventajas de usar un Service Worker.
- Descripción general de un Service Worker (API asíncrona, API basada en eventos, precaching, aislamiento del hilo principal).



¿QUÉ VAMOS A VER?

- Ciclo de vida de un Service Worker.
- Cómo interactúa el Service Worker con el caché y el acceso a la red.
- Cómo configurar Service Worker para ReactJs.
- Funcionamiento de una PWA con HTTPS.
- Estrategias de almacenamiento en caché de Service Worker.
    - Stale-While-Revalidate.
    - Cache-first.
    - NetworkFirst.
    - CacheOnly.
    - NetworkOnly.
    - CacheAndNetwork.

Pongamos a prueba lo aprendido 😊!!!



Adalid aspasia

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

En este ejercicio, vamos a convertir un proyecto creado con REACT https://github.com/adalid-cl/ESPECIALIZACION_FRONTEND_M6_AE1 en una Progressive Web App (PWA) funcional, con instalación en dispositivos.

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

1. Clona el repositorio si aún no lo tienes:

```
git clone https://github.com/adalid-cl/ESPECIALIZACION_FRONTEND_M6_AE1.git
cd ESPECIALIZACION_FRONTEND_M6_AE1
npm install
```

- Instala las dependencias necesarias:

```
npm install vite-plugin-pwa --save-dev
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

2. Configurar Vite para Soporte PWA

Añade el plugin de PWA en la configuración de Vite en el archivo **vite.config.js**:

- Permite registrar el Service Worker automáticamente.
- Define el **nombre, iconos y comportamiento** de la PWA.

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

export default defineConfig({
  plugins: [
    react(),
    VitePWA({
      registerType: 'autoUpdate',
      manifest: {
        name: 'Mi PWA',
        short_name: 'PWA',
        description: 'Aplicación PWA con React y Vite',
        theme_color: '#ffffff',
        start_url: '/',
        display: 'standalone',
        icons: [
          {
            src: '/icons/icon-192x192.png',
            sizes: '192x192',
            type: 'image/png'
          },
          {
            src: '/icons/icon-512x512.png',
            sizes: '512x512',
            type: 'image/png'
          }
        ]
      },
      devOptions: {
        enabled: true
      }
    })
  ]
});
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

3. Agregar el Archivo de Manifiesto

- **Ubicación:** public/manifest.json
- Los iconos ya se encuentran en la carpeta **public/icons/** pero sus tamaños no son los adecuados, sin embargo funcionara.

```
{
  "name": "Mi PWA",
  "short_name": "PWA",
  "description": "Aplicación PWA con React y Vite",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#ffffff",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

4. Enlazar el Manifiesto en index.html

- Permite que los navegadores detecten la PWA y muestren la opción de instalación.
- Agrega esta línea dentro de <head>:

```
<link rel="manifest" href="/manifest.json">
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

5. Crear y Configurar el Service Worker

- **Ubicación:** public/sw.js
- **install:** Guarda en caché los archivos esenciales para el modo offline.
- **fetch:** Intercepta peticiones y responde con la caché si es posible.

```
const CACHE_NAME = "pwa-cache-v1";
const urlsToCache = ["/", "/index.html", "/main.js",
"/icons/icon-192x192.png"];

self.addEventListener("install", event => {
  event.waitUntil(
    caches.open(CACHE_NAME).then(cache => {
      return cache.addAll(urlsToCache);
    })
  );
});

self.addEventListener("fetch", event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

6. Registrar el Service Worker en main.jsx

- **Ubicación:** src/main.jsx
- **Registra el Service Worker** cuando la aplicación se carga.
- **Muestra errores en la consola** en caso de problemas.

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";

if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/sw.js")
    .then(() => console.log("Service Worker registrado"))
    .catch(error => console.log("Error en Service Worker",
error));
}

ReactDOM.createRoot(document.getElementById("root")).render
(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

7. Implementar Estrategias de Caché

- Modifica **sw.js** para usar **Stale-While-Revalidate**
- Muestra la **versión en caché primero** (si existe).
- Luego, busca una **versión más actualizada en la red**.

```
self.addEventListener("fetch", event => {
  event.respondWith(
    caches.open(CACHE_NAME).then(cache => {
      return cache.match(event.request).then(response
=> {
        const fetchPromise =
fetch(event.request).then(networkResponse => {
          cache.put(event.request,
networkResponse.clone());
          return networkResponse;
        });
        return response || fetchPromise;
      });
    })
  );
});
```

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

8. Probar la Instalación y el Funcionamiento Offline

- Ejecuta la aplicación

```
npm run dev
```

- **Abre DevTools (F12) → Application → Service Worker**
  - Confirma que el Service Worker está activado.
  - Detén el servidor local y verifica si la PWA sigue funcionando.

Ejercicio Guiado: Convertir el Proyecto Existente en una PWA

9. Construir y Desplegar la PWA

- Ejecuta el siguiente comando para compilar el proyecto:

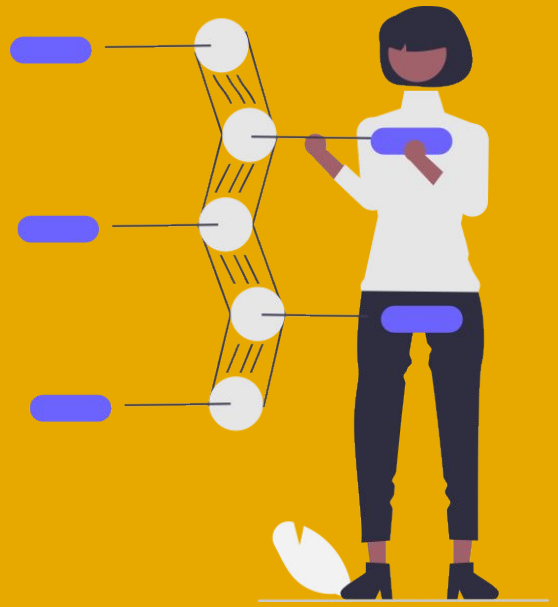
```
npm run build
```

- Luego, sirve el contenido estático generado con:

```
npm install -g serve
serve -s dist
```

- Para producción, sube el contenido de la carpeta **dist/** a un servidor con **HTTPS**.

Resumen de lo aprendido 🧑‍🎓



Adalid aspasia

Resumen de lo aprendido

- **Entendiste qué es una PWA**, sus características, beneficios y diferencias con apps web y nativas.
- **Aprendiste a usar el Manifiesto**, configurándolo para personalizar e instalar la PWA.
- **Exploraste los Service Workers**, su ciclo de vida y su rol en la caché y el rendimiento.
- **Implementaste estrategias de almacenamiento en caché**, optimizando la velocidad y funcionalidad offline.

GRACIAS POR TU ATENCIÓN

Nos vemos en la próxima clase

Chile
Adalid
aspasia
LA FORMACIÓN DE TU FUTURO