



Progressive Web Apps (PWA)

Programación de aplicaciones para
dispositivos móviles

Curso 2024/2025



PWA: Introducción (1/2)

- PWA => *Progressive Web Apps*.
- Aplicaciones web (HTML, CSS, JS), pero con ciertas particularidades que le hacen parecerse a una app nativa:
 - Funcionamiento sin conexión: el usuario puede seguir consultando la web en el navegador, aunque haya perdido la conexión.
 - Uso de notificaciones nativas: muy útil cuando se trata de móviles.
 - [Acceso al hardware del dispositivo](#) (ej., [sensores](#), [cámara](#) y [GPS](#)).
 - Instalación sencilla: no se necesita descargar nada desde una *app store* (ej., Google Play o Apple Store).
 - Pueden utilizar toda la pantalla sin necesidad de ejecutarse desde la interfaz de usuario de un navegador.
 - Almacenamiento local.

PWA: Introducción (2/2)

- Aunque [PWA](#) sea técnicamente una página web reemplazan a las aplicaciones nativas.
- El mismo código sirve para una aplicación en escritorio, web, móviles y *tablets*.
- Son progresivas porque se diseñan con la estrategia “adaptación progresiva”: se diseña un núcleo básico fundamental (contenido y estructura básica) y se van añadiendo capas de funcionalidad sobre ese núcleo.
- Para implementar una PWA sencilla basta con hacer una **página web** y añadirle una serie de elementos.
- Una PWA bien hecha funcionará en las versiones más nuevas de los navegadores, pero también en sus versiones más antiguas. Aunque pierda funcionalidad podrá verse su contenido.

PWA: Manifiesto (1/3)


- El [manifiesto](#) es un archivo de texto con datos en formato JSON que describe la PWA.
- El navegador sabrá que se trata de una PWA cuando detecte que el archivo **index.html** apunta a un archivo de texto con formato JSON que representa el manifiesto (sección `<head>` del archivo). Ejemplo:


```
<!doctype html>
<html lang="en">
  <head>
    <link rel="manifest" href="manifest.json" />
    <!-- ... -->
  </head>
  <body></body>
</html>
```

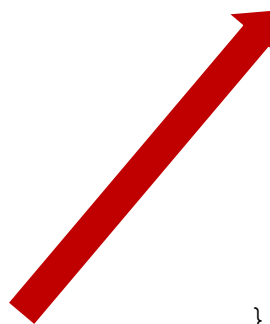
PWA: Manifiesto (2/3)

- Estructura de manifiesto (más info):

```
{  
  "short_name": "Nombre corto",  
  "name": "Nombre largo de la aplicación",  
  "description": "Descripción de la PWA",  
  "icons": [ ],  
  "start_url": "/",  
  "background_color": "#<color-fondo>",  
  "display": "standalone",  
  "scope": "/",  
  "theme-color": "#<color-tema-app>",  
  "orientation": "any"  
}
```

 Maskable icons

 “standalone”, para aspecto de app nativa.



Chrome, por ejemplo,
requiere dos tamaños:
192x192 px y 512x512 px.

PWA: Manifiesto (3/3)

■ Ejemplo de manifiesto:

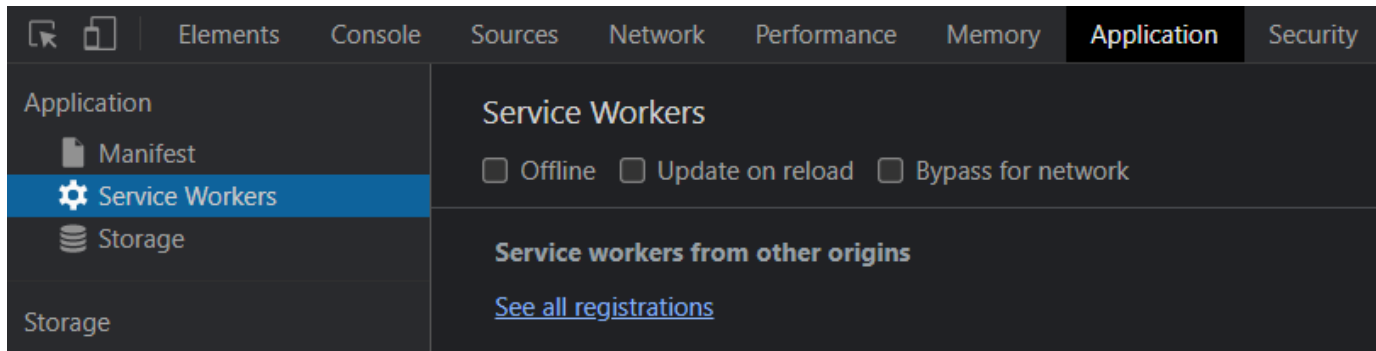
```
{
  "name": "Hello world",
  "short_name": "Hello",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#fdfdfd",
  "theme_color": "#db4938",
  "orientation": "portrait-primary",
  "icons": [
    {
      "src": "/images/icons/icon-192x192.png",
      "type": "image/png", "sizes": "192x192"
    },
    {
      "src": "/images/icons/icon-512x512.png",
      "type": "image/png", "sizes": "512x512"
    }
  ]
}
```

Ejemplos PWA:

- [HighTide](#)
- [2048](#)

PWA: Service Workers (1/8)

- [Service workers](#) son scripts escritos en JavaScript que se ejecutan en el navegador, independientemente de la página web a la que va vinculada.
- Funcionan como un proxy entre la red y la app.
- No tienen acceso al DOM ya que no tienen acceso a las páginas web.
- Funcionan **exclusivamente con HTTPS** (aunque si el dominio es localhost podría funcionar).
- Dada una página web, a través de la opción *Inspeccionar*, se pueden consultar los *Service Workers* activos en la pestaña *Application*.



PWA: Service Workers (2/8)

- Si se ha accedido por primera vez a una página web se puede registrar un *Service Worker* asociado a dicha página.
- De esta manera, cuando el *Service Worker* esté registrado, el navegador recordará lo que hay en ese dominio (la raíz de la URL), y lo usará cuando se vuelva a intentar acceder a ese dominio (**aunque no haya conexión**).
- Para registrar un *Service Worker* simplemente hay que ejecutar cierto código JavaScript en el fichero **index.html** de la app.

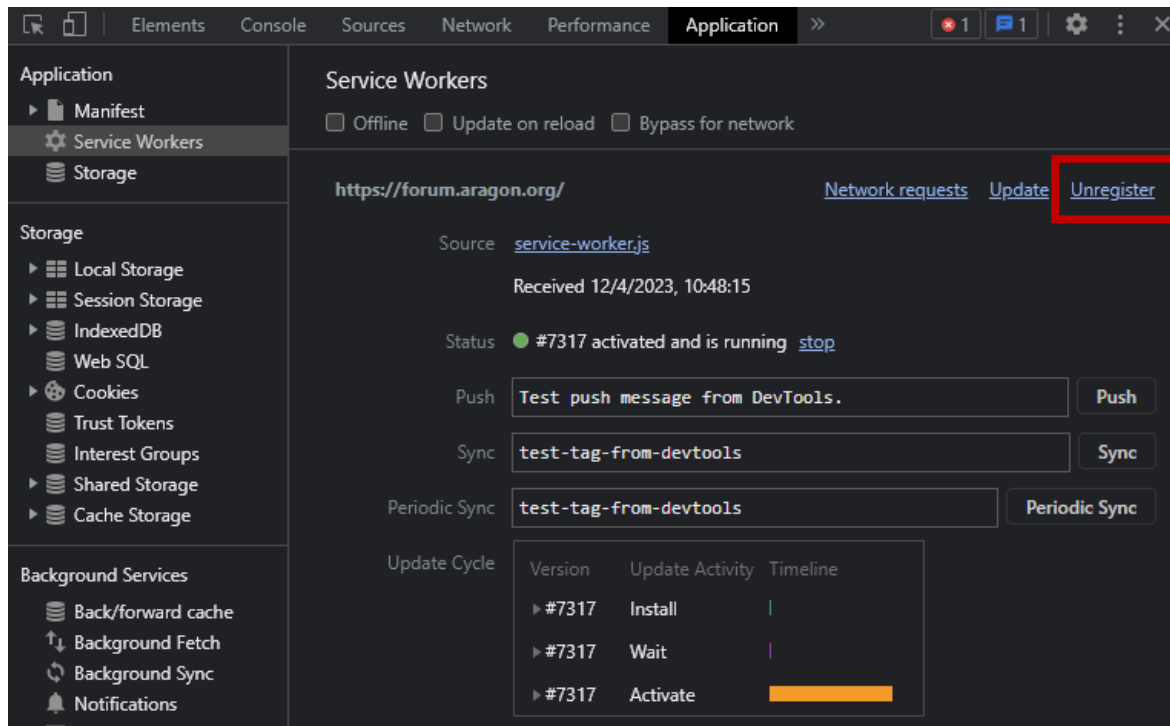
PWA: Service Workers (3/8)

- Ejemplo donde el *Service Worker* estará implementado en el fichero **sw.js** ([más info](#)):

```
<head>
  <script>
    // Comprobar si el navegador ejecuta service workers
    if ('serviceWorker' in navigator) {
      window.addEventListener('load', () => {
        navigator.serviceWorker.register('sw.js').then(reg => {
          console.log('Todo bien:', reg)
        }, function (err) {
          console.log('Fallo:', err)
        })
      })
    }
  </script>
</head>
<body>
  ...
</body>
```

PWA: Service Workers (4/8)

- Para borrar un *Service Worker* instalado por una página web hay que seleccionar la opción **Unregister** en la pestaña **Application > Sección Application > Service Workers**:

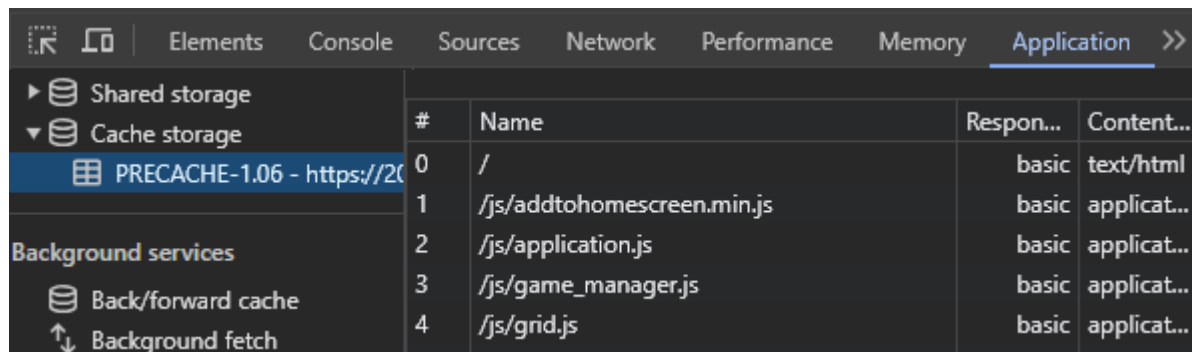


PWA: Service Workers (5/8)

- A través de un *Service Worker* se puede acceder al almacenamiento de la **caché de archivos del navegador**.
- Se pueden almacenar archivos (por ejemplo, index.html, index.js, app.js, logo.png y background.png) en la caché de archivos del navegador, que estarán disponibles para su acceso independientemente de que haya o no conexión.
- De esta manera, un *Service Worker* puede almacenar los archivos relativos a una página web, y los puede recuperar cuando se vuelva a la URL asociada a dicha página (dominio). Así, la aplicación web funcionará, aunque no haya conexión.
- Cuando haya una versión nueva de los archivos (se podrá saber cuándo haya conexión) se actualizará la caché.

PWA: Service Workers (6/8)

- Se pueden consultar los recursos que se han almacenado en la caché desde la pestaña **Application > Sección Storage > Cache Storage**:

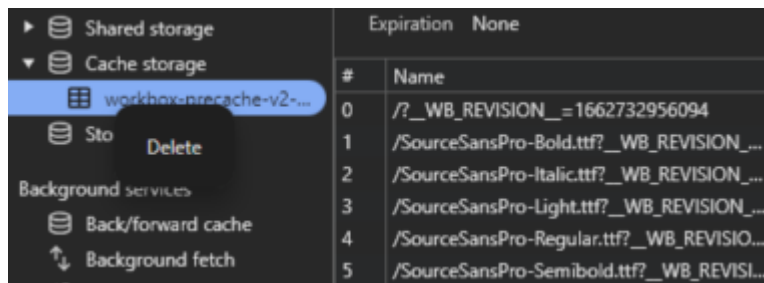


#	Name	Respon...	Content...
0	/	basic	text/html
1	/js/addtohomescreen.min.js	basic	applicat...
2	/js/application.js	basic	applicat...
3	/js/game_manager.js	basic	applicat...
4	/js/grid.js	basic	applicat...

- **NOTA:** Al tratarse de un **almacenamiento local al navegador**, los ficheros no estarán disponibles en ningún otro navegador.

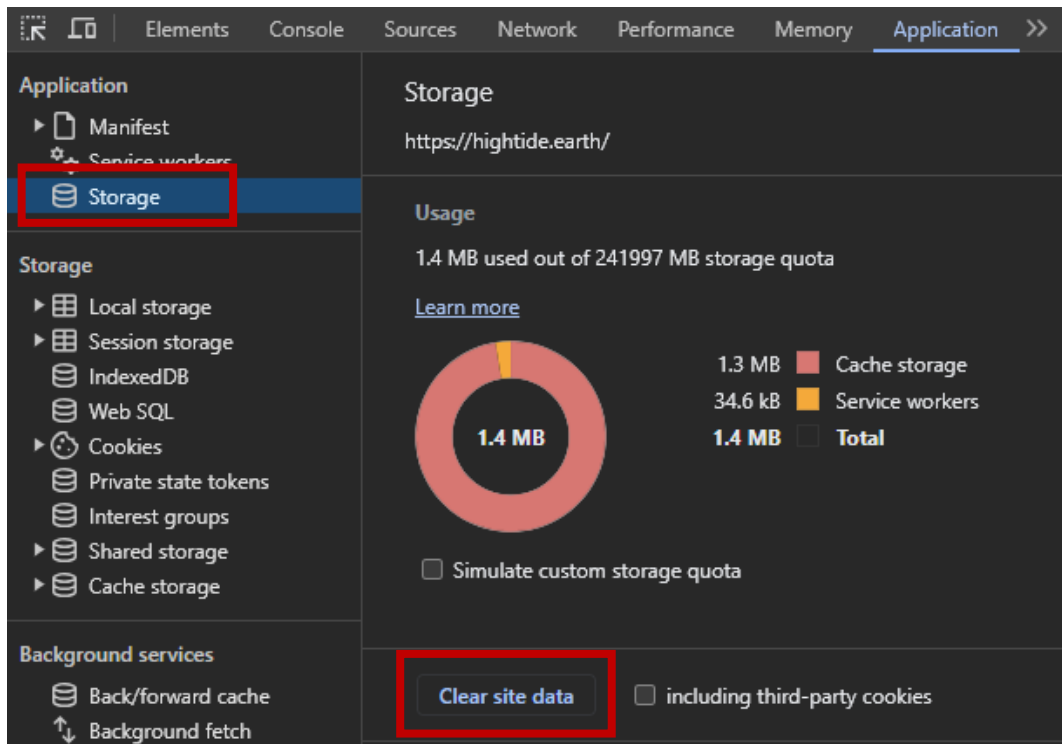
PWA: Service Workers (7/8)

- Para borrar los datos de la caché y forzar la recarga de la página hay que situarse sobre el ítem de la caché y con el botón derecho del ratón seleccionar la opción **Delete**, en la pestaña **Application > Sección Storage > Cache Storage**:



PWA: Service Workers (8/8)

- Para borrar todos los datos hay que seleccionar la opción **Clear site data** en la pestaña **Application** > Sección **Application** > **Storage**:



PWA: Service Workers : Caché de archivos (1/5)


- Para realizar el almacenamiento en la caché de archivos, y así poder acceder a ellos posteriormente, se podría utilizar el [API de Service Workers](#).
- Sin embargo, resulta más útil hacer uso de la librería [workbox](#), ya que ahorra mucho trabajo vinculado a los *Service Workers*.

PWA: Service Workers: Caché de archivos (2/5)

- Activar la caché de archivos (a través de [workbox precaching](#)).
Ejemplo de fichero **sw.js**:

```
import {precacheAndRoute} from 'workbox-precaching';

precacheAndRoute([
  {url: '/index.html', revision: '383676'},
  {url: '/styles/app.css', revision: null},
  {url: '/scripts/app.js', revision: null},
  // ... otras entradas ...
]);
```



En este caso, no hay información de
revisión por lo que la caché no se
actualizará cuando haya modificaciones.

PWA: Service Workers : Caché de archivos (3/5)

- En **workbox**, la información sobre las revisiones es necesaria para que el navegador pueda saber qué archivos han cambiado.
- La gestión manual de las revisiones puede ser muy tediosa, por lo que workbox ofrece una aplicación de línea de comandos que se encargará de gestionar el valor de revisión: `workbox-cli`.

PWA: Service Workers : Caché de archivos (4/5)

- Instalar **node.js**:

```
# Instalar node.js (que viene con 'npm')  
cd ruta_a_la_carpeta_del_proyecto  
npm init --yes
```

- Instalar **workbox-cli**:

```
# Situados en la carpeta del proyecto  
npm install -D workbox-cli
```

- Iniciar el asistente de gestión de revisiones:

```
# Situados en la carpeta del proyecto  
npx workbox wizard
```

- Tras el proceso (aceptar opciones por defecto), se creará un archivo **sw.js** (permite cambiar el nombre en el proceso de creación). Si ya existiera un fichero con el nombre **sw.js** lo sobrescribe automáticamente.

PWA: Service Workers : Caché de archivos (5/5)

- El fichero **sw.js** tiene ya definido el *Service Worker* que gestiona la caché y las revisiones.
- Así, cuando se realicen cambios, la revisión cambiará y el navegador recargará el recurso. Hay que llamar a este comando cada vez que se realicen cambios:

```
# Para regenerar una versión, simplemente se utiliza la herramienta.  
# Esto asume que nuestra configuración se llama `workbox-config.js`  
# Hay que estar situados en la carpeta del proyecto.  
# Ejecutar este comando cada vez que se hayan producido cambios:  
npx workbox generateSW
```

- Si se realiza algún cambio, pero no se regenera el *Service Worker*, no se recargará en el navegador.

PWA: Ciclo de trabajo

- Tener que ejecutar el comando `generateSW` cada vez que se hace algún cambio para probar es tedioso.
- Una solución más rápida es **desarrollar la app como una aplicación web sin *Service Workers***, y convertirla a PWA cuando se desee desplegar la app en el entorno de (pre)producción.
- **Buenas prácticas:** Para el entorno de desarrollo se puede tener:
 - Un archivo `index.html` que haga referencia al *Service Worker*.
 - Un archivo `devel.html` que sea igual a `index.html`, pero que no haga uso del *Service Worker*.
- Así, se utilizará **`devel.html` en desarrollo**, pero se **desplegará con `index.html`**.

PWA: Almacenamiento local (1/2)

- Las PWA se pueden beneficiar del [API de almacenamiento local](#).
- Existe una variable denominada [localStorage](#), que constituye un objeto que permite guardar y leer valores.
- Estos valores se mantienen entre sesiones (a diferencia de [sessionStorage](#)), y cada dominio tiene su propio objeto.
- Añadir o modificar una clave (la clave siempre se guarda como un *string*):

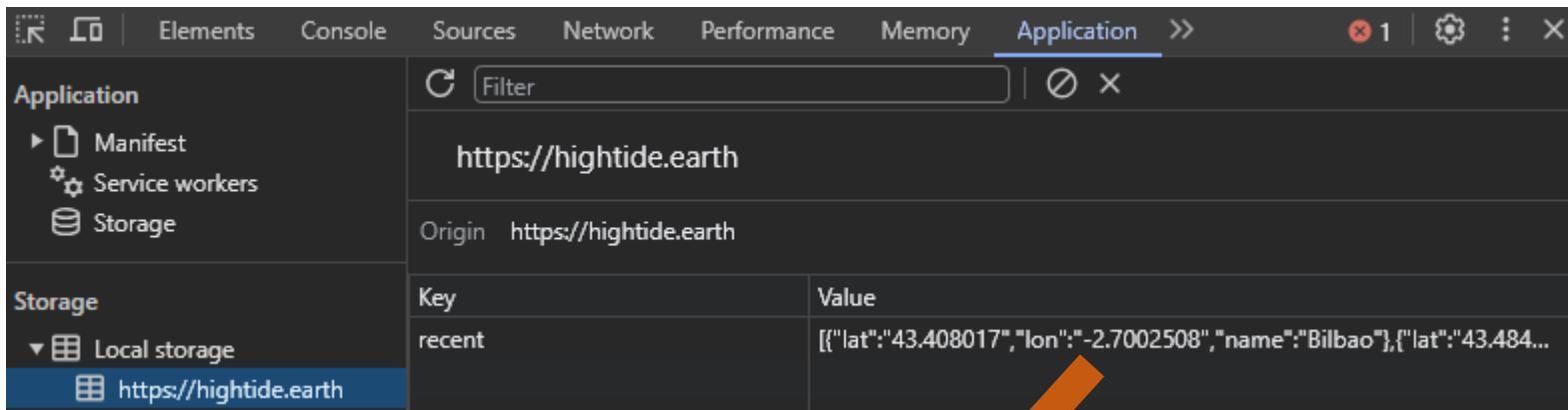
```
localStorage.setItem('miclave', 6)
localStorage.clave = 6

// Para acceder al valor.
localStorage.getItem('miclave')
localStorage.clave
```

- **Borrar una clave:** `localStorage.removeItem('clave')`
- **Borrar todo:** `localStorage.clear()`

PWA: Almacenamiento local (2/2)

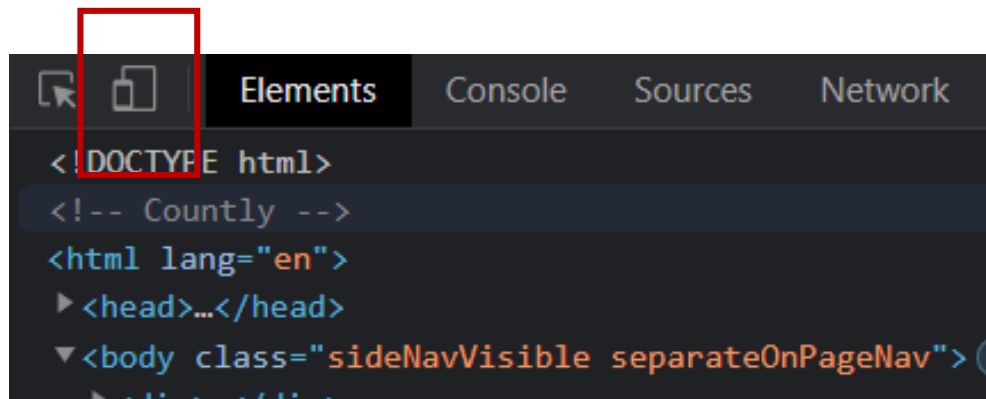
- Se puede consultar y modificar la lista de datos guardados, por dominio, desde la pestaña **Application** > **Sección Storage** > **Local Storage**:



```
> [{lat: "43.408017", lon: "-2.7002508", name: "Bilbao"},...]  
  ▶ 0: {lat: "43.408017", lon: "-2.7002508", name: "Bilbao"}  
  ▶ 1: {lat: "43.48490959999999", lon: "-5.2712248", name: "Gijón"}  
  ▶ 2: {lat: "43.427173", lon: "-2.8126769", name: "Bilbao"}
```

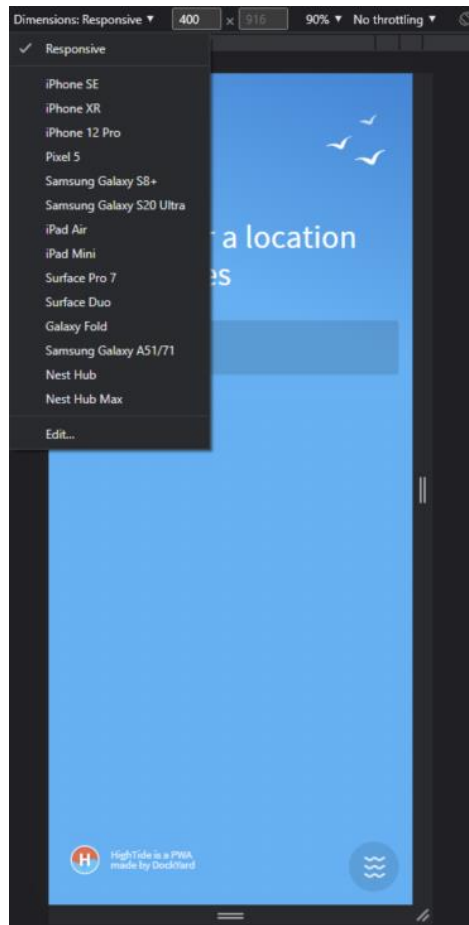
PWA: Visualización en modo dispositivo móvil (1/2)

- Para probar la aplicación en un teléfono móvil o *tablet*, basta con cargar la URL en un navegador (por ejemplo, Chrome) instalado en el dispositivo móvil.
- El navegador Chrome dispone de una [utilidad](#) al inspeccionar la página, que permite ver la aplicación con el aspecto de un dispositivo móvil:



PWA: Visualización en modo dispositivo móvil (2/2)

- Una vez pulsado el modo dispositivo móvil, la página web se adapta para que se puedan elegir las opciones del dispositivo que se está simulando:



Enlaces de interés

- <https://web.dev/progressive-web-apps/>
- <https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/>
- <https://support.google.com/chrome/answer/9658361?co=GENI&E.Platform%3DAndroid&hl=en-GB>
- [Maskable icons](#)
- [PWA con aspecto de app](#)

Comparativa apps Nativas/Web/PWA

App	Descripción	Instalación	Acceso a las características del dispositivo	Dependencia de conexión	Entrega de actualizaciones
Nativa	Implementada para una plataforma específica (ej., iOS o Android).	Se instala directamente en el dispositivo a través de una <i>app store</i> .	Acceso total. Posibilidad de personalizar la configuración del sistema.	Algunas apps pueden tener dependencia de conexión, pero otras no.	Dependiente de la normativa de la <i>app store</i> .
Tecnologías Web	Se ejecuta en un servidor remoto y se accede a través de la interfaz de un navegador.	No requiere instalación en el dispositivo (URL en un navegador).	Acceso limitado.	La app no puede funcionar si no tiene conexión.	Se realizan de manera inmediata para los usuarios.
PWA	La(s) página(s) del sitio web tiene el aspecto de una interfaz nativa.	No se instala en el dispositivo móvil. Para poner en forma nativa hay que hay que visitar la web e instalarla desde ahí.	Acceso limitado.	Permite grabar funcionalidad para el modo offline (es decir, interfaz y contenido disponible gracias al almacenamiento en caché).	Automatizada sin necesidad de pasar por una <i>app store</i> .