

17 de septiembre del 2025

ING. Domingo Fraga Hernandez

PRACTICA 1

PROGRESSIVE LANDING PAGE

Equipo:

Rosendo Jesus Gonzales Espinoza

Leobardo Quintero Ruiz

Mario Antonio Mejia Valadez

Mario Mejia

UNIVERSIDAD TECNOLOGICA DLE NORTE DE COAHUILA | DESARROLLO Y GESTION DE
SOFTWARE

INDEX

PROGRESSIVE WEB APP	2
EXPLICACION DE LA APLICACIÓN.....	2
Explicacion por Mario Mejia Valadez.	5
EXPLICACION CON IMÁGENES DE LA PROGRESSIVE WEB APP.....	5
PAGINA WEB FUNCIONANDO CON SERVICE WORKER.....	10
QUE ES EL SERVER WORKER?	10
QUE SE NECESITA DEL SERVICE WORKER?	11
Explicacion por Leobardo Quintero Ruiz.....	12
CICLOS DE VIDA DE SERVER WORKER.....	12
PRACTICA REALIZADA	14
OBTENCION DE ARCHIVOS	14
Explicacion por Rosendo Jesus Gonzalez Espinoza.	18
FUNCIONALIDADES Y CAMBIOS EN LOS CODIGOS DEL SERVER WORKER.	18
PWA-NETWORK.JS	18
PWA-CONSTANTS.JS	18
PWA-INSTALLER.JS.....	19
PWA-SWA.JS.....	20
RESULTADOS:.....	22
Recursos de la pagina.	22
Consola.	23
servicio	24
Funcionamiento de la pagina.....	25
CONCLUSIONES.....	26

PROGRESSIVE WEB APP

Las **aplicaciones web progresivas** (mejor conocidas como **PWAs** por «**Progressive Web Apps**») son aplicaciones web que utilizan APIs y funciones emergentes del navegador web junto a una estrategia tradicional de mejora progresiva para ofrecer una aplicación nativa —como la experiencia del usuario para aplicaciones web multiplataforma. Las aplicaciones web progresivas son un patrón de diseño útil, aunque no son un estándar formalizado. Se puede pensar que PWA es similar a AJAX u otros patrones similares que abarcan un conjunto de atributos de aplicación, incluido el uso de tecnologías y técnicas web específicas. Este conjunto de documentos te dice todo lo que necesitas saber sobre ellas.

Para poder llamar PWA a una aplicación web, técnicamente hablando debe tener las siguientes características: Contexto seguro (**HTTPS**), uno o más Servicio Workers y un archivo de manifiesto.

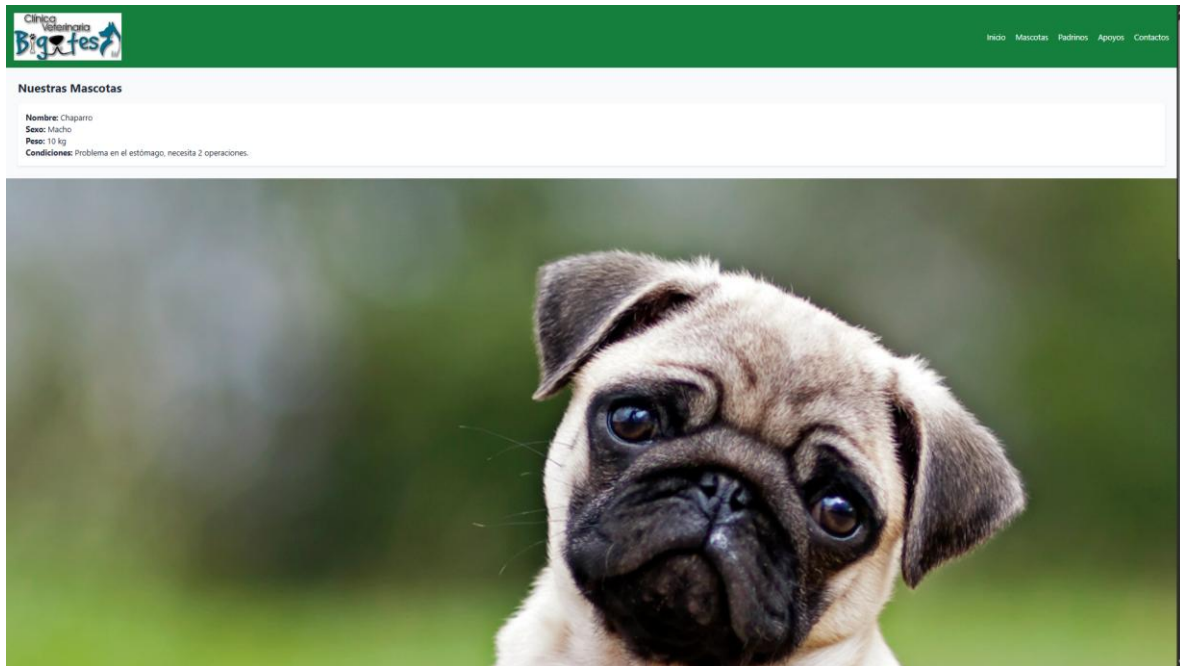
EXPLICACION DE LA APLICACIÓN

Esta progressive web app es de un refugio de animales con el cual puede observar mascotas en el momento, padrinos o donadores y apoyos, además de tener un formulario de contacto para mandar un mensaje.

Para esta ocasión utilizamos angular JS para la web app con diferentes vistas de cada uno de los apartados, anteriormente se iba a utilizar tailwind pero hubo problemas con el CDN (no sabíamos como aplicarlo en verdad).

Cada uno tiene un html distinto pero con el cual solo el header es el único que terminara de manera fija además de cada sección contar con sus tablas específicas.

Cada apartado tiene lo propio para que funcione adecuadamente.



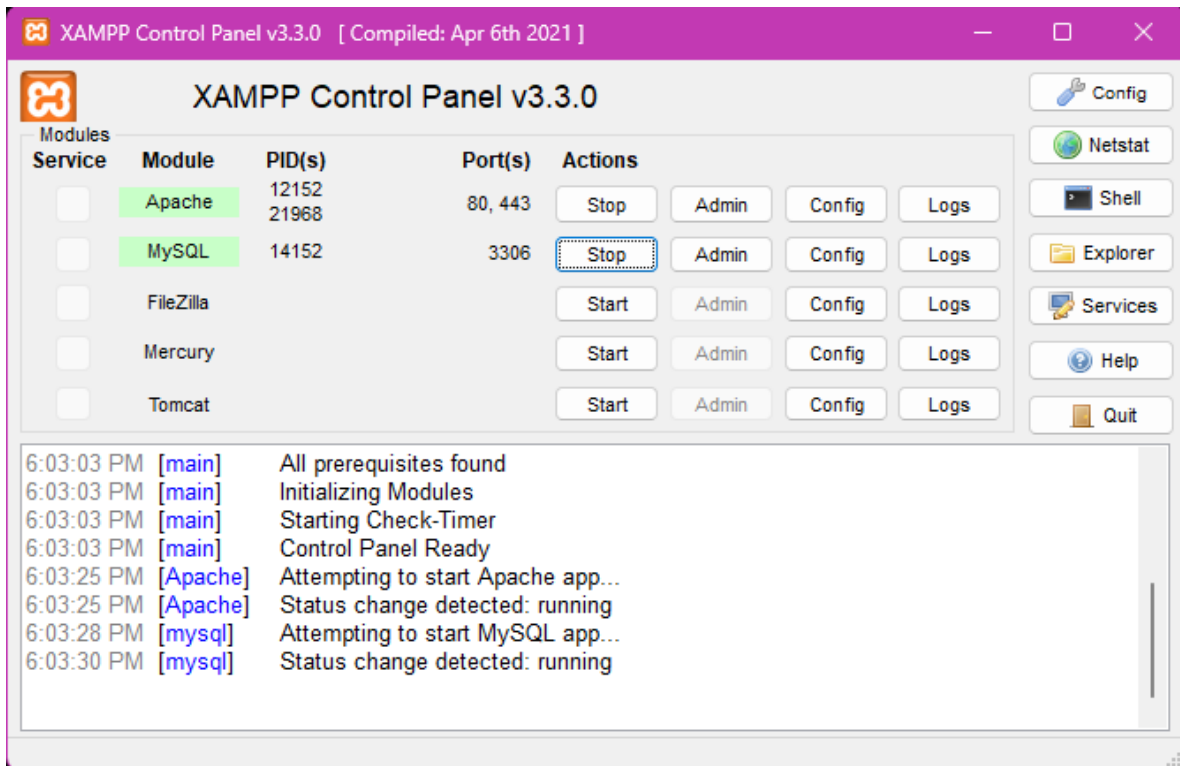
```
File Edit Selection View Go Run Terminal Help
index.html padinos.html contactos.html pets.html x hero.html apoyos.html home.html app.js
view > pets.html > section.p-4 > div.bg-white.shadow.rounded.p-4
1 <section class="p-4">
2 <div class="mb-2 font-weight bold">Nuestras Mascotas</div>
3 <div class="bg-white shadow rounded p-4">
4 <strong>Nombre:</strong> Chaparro</div>
5 <strong>Sexo:</strong> Macho</div>
6 <strong>Peso:</strong> 10 kg</div>
7 <strong>Condiciones:</strong> Problema en el estómago, necesita 2 operaciones.</div>
8 </div>
9 </div>
10 </section>
11
```

Y aquí esta la seccion de mascotas que yo la tengo nombrada como pets.html.

Asi como esta seccion son todas las demas secciones, todo esto se hace con el motivo de tener un control especifico de los index y no envolver todo en un mismo index.html.

Explicacion por Mario Mejia Valadez.

EXPLICACION CON IMÁGENES DE LA PROGRESSIVE WEB APP

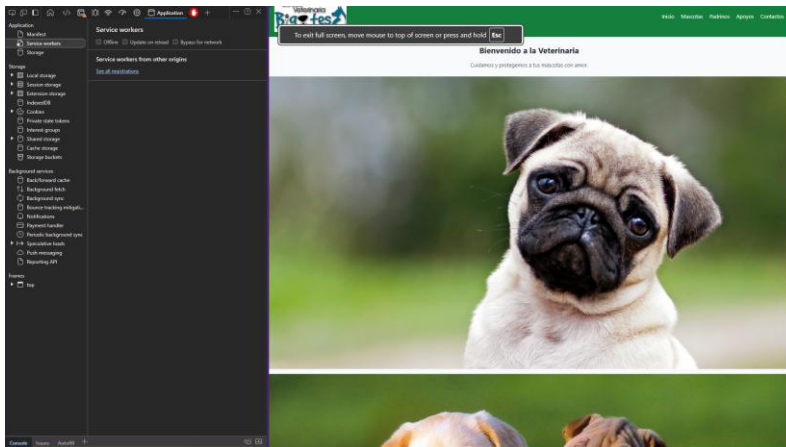


En este caso encendemos primero el localhost psrs acceder a la pagina web

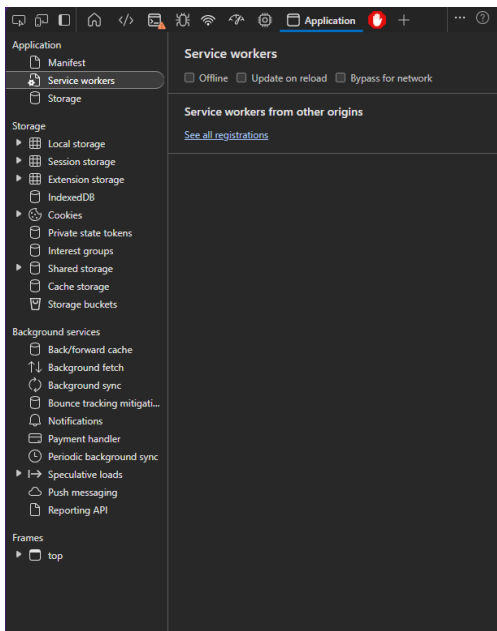
En el encabezado se muestran los liinks utilizados para la platilla de angular JS y abajo donde se muestran los links de las imágenes en google para el funcionamiento de esta practica.

[illegible]

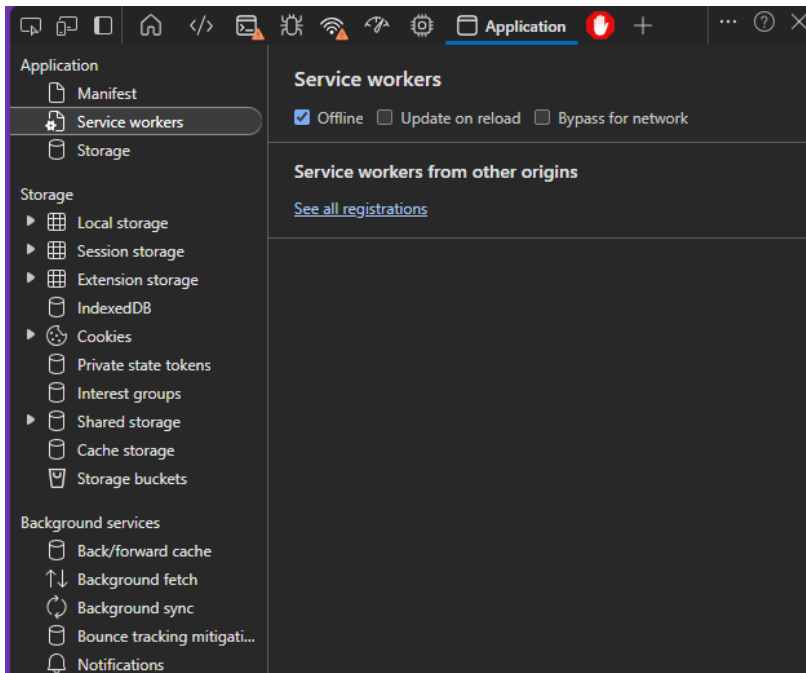
Una vez comprobado que tenemos elementos directamente de internet abrimos el DevTools para hacer la simulacion de desconexion. Para acceder a esta herramienta abrimos el devTools o inspeccionar codigo presionando F12 o click derecho y luego “Inspect”, una vez abierta la ventana de devTools nos dirigimos al apartado de aplicaciones que se encuentra en la parte superior del DevTools.



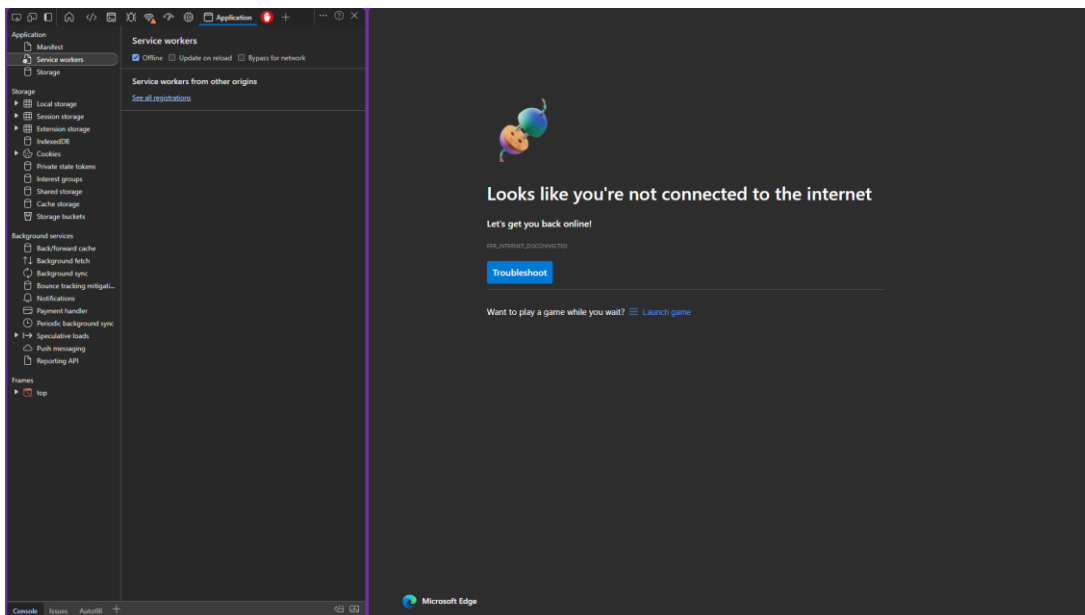
Se mostrara una ventana diferente a la del codigo y en esa seccion en la columna uno donde esta la opcion de “Services Worked” y se mostrara las opciones de “offline”



Lo presionamos y automaticamente estaremos simulando que no funciona.



Activamos la opcion y offline y automaticamente ya no funcionara los ni los links de las imágenes, ni el CDN de tailwind y los links de angular JS



PAGINA WEB FUNCIONANDO CON SERVICE WORKER

QUE ES EL SERVER WORKER?

Los service workers tienen sentido en muchos escenarios, pero básicamente nos permiten mantener un sitio web trabajando en segundo plano. Básicamente consisten en un script (archivo Javascript) que continúa su ejecución, independientemente de si una página web está abierta o no.

Algunas cosas interesantes de los Service Workers:

- Un Service Worker no puede acceder directamente al DOM, el Service Worker se comunica con las páginas que el controla a través de la interfaz `PostMessage`.
- Con un Service Worker puedes controlar cómo las peticiones de red de son manejadas
- Son capaces de implementar diversos sistemas de cacheo.
- Pueden persistir información a través del `indexedDB`.

QUE SE NECESITA DEL SERVICE WORKER?

- Soporte de navegador:

sólo se puede usar un service worker si el navegador permite registrarlo. La mayoría de los navegadores lo han implementado y el resto ha declarado ya su apoyo a la tecnología y su soporte está proceso de desarrollo. Puedes ver el soporte actual aquí. Algunos que soportan son Chrome, Edge, Opera, Firefox y safari (macOS).

- HTTPS:

Es necesario que el sitio web se entregue mediante un protocolo seguro para que el Service Worker pueda ser registrado.

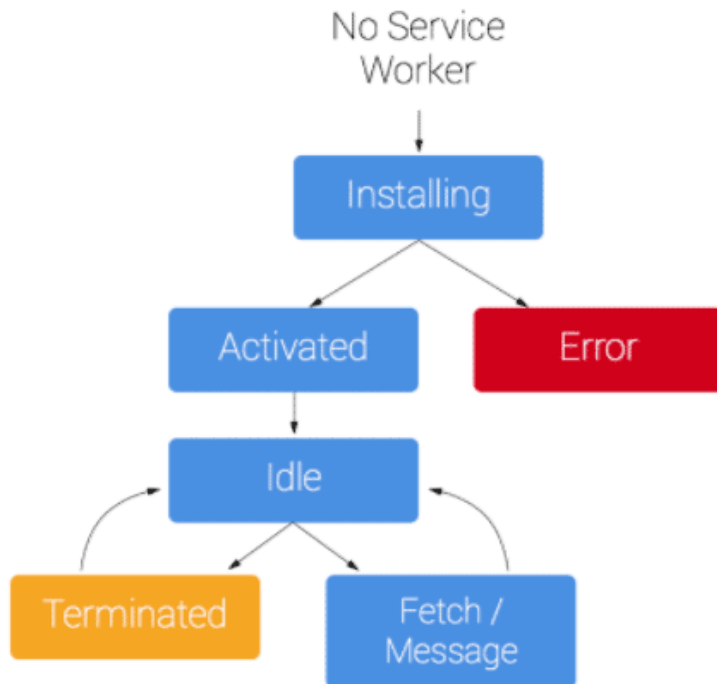
Nota:

Se puede utilizar localhost para el testeo de server worker ya que localhost es considerado un servidor seguro y podra registrar en el cache

Explicacion por Leobardo Quintero Ruiz.

CICLOS DE VIDA DE SERVER WORKER

Los Service worker tienen un ciclo de vida independiente de la Web. Entender el ciclo de vida de los Service Worker derivará en entregar una excelente experiencia a nuestros usuarios.



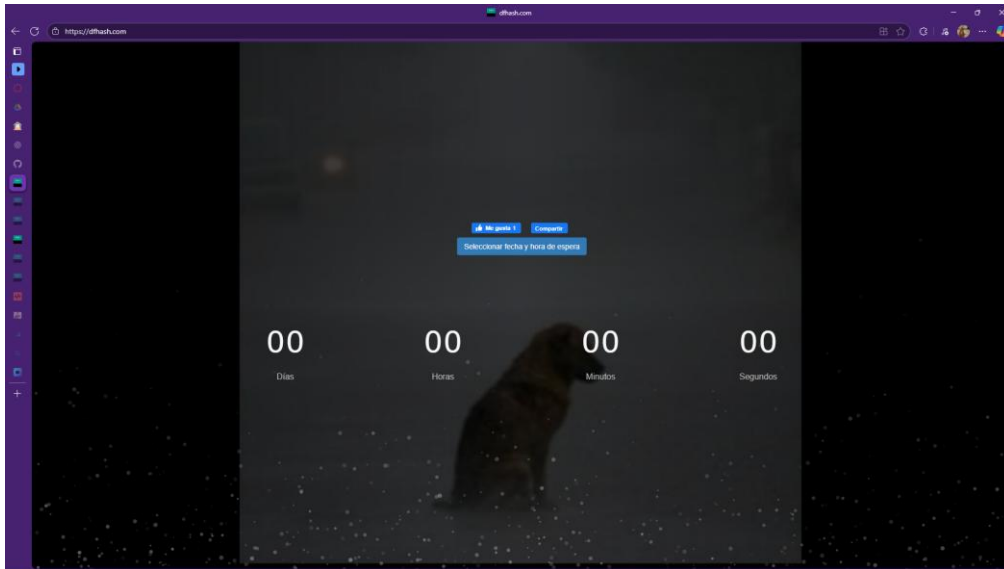
Install: Este es el primer evento que ocurre. Ocurre solo una vez por Service Worker. Si la promesa que llamas en este evento falla, el navegador no lo registra y no deja que el Service Worker tome control del cliente.

Activate: Una vez el Service Worker está controlando el cliente y está listo para usar y manejar eventos, se pasa al estado Activate. Podríamos entenderlo como que nuestro Service Worker está activo.

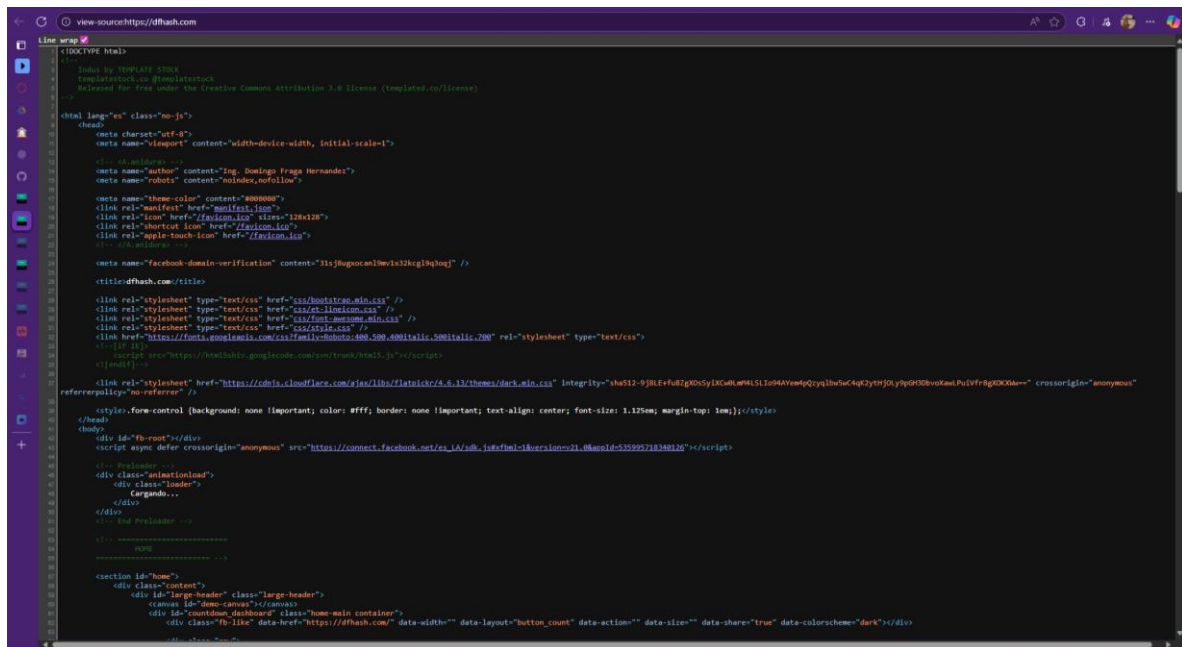
Terminated, Fetch: Una vez el Service Worker está en control de las páginas, podrá estar en uno de estos dos estados. Terminated se aplica para ahorrar memoria. Por su parte, si está haciendo manejo peticiones de red, su estado será fetch.

PRACTICA REALIZADA

OBTENCION DE ARCHIVOS

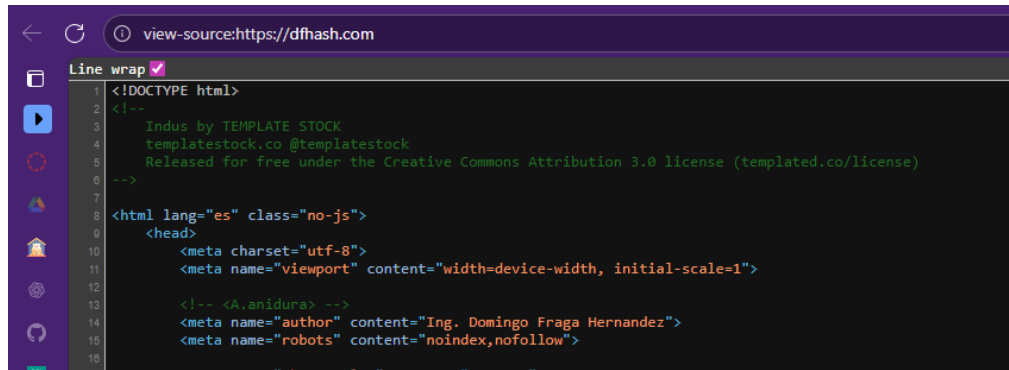


De primeras nos dirigimos a esta pagina para la obtencion de los archivos: dfhash.com, luego de ingresamos inspeccionamos el codigo fuente con Ctrl + U de la pagina para obtener los script de la funcionalidad del server worker.



Aquí tomaremos unos puntos importantes para la practica:

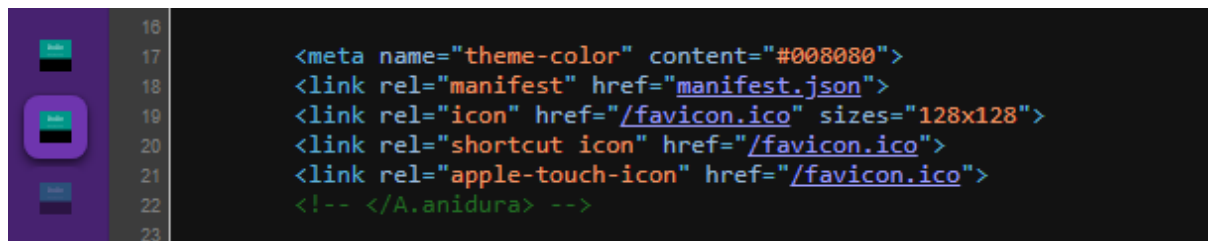
Tenemos que establecer el viewport para que pueda funcionar en dispositivos móviles de igual forma y que no tengamos muchas complicaciones a la hora de hacer una página responsive



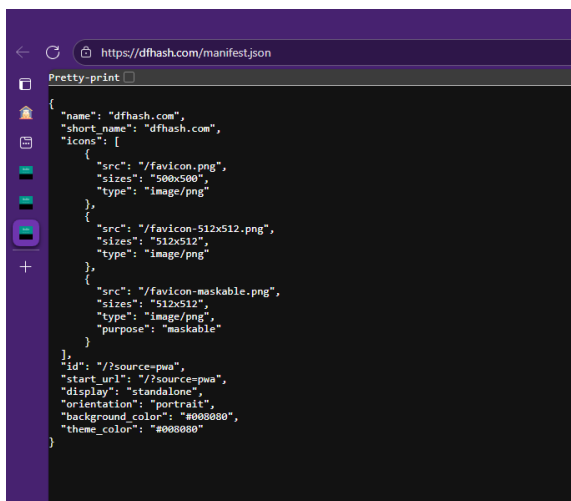
```
<!DOCTYPE html>
<!--
Indus by TEMPLATE STOCK
templatestock.co @templatestock
Released for free under the Creative Commons Attribution 3.0 license (templated.co/license)
-->
<html lang="es" class="no-js">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- <A.anidura> -->
<meta name="author" content="Ing. Domingo Fraga Hernandez">
<meta name="robots" content="noindex,nofollow">
```

Copiamos el viweport y lo pegamos en nuestro código index.html.

También tomaremos las etiquetas para utilizar los iconos además de tomar el manifiesto.json (Es muy importante el manifiesto).

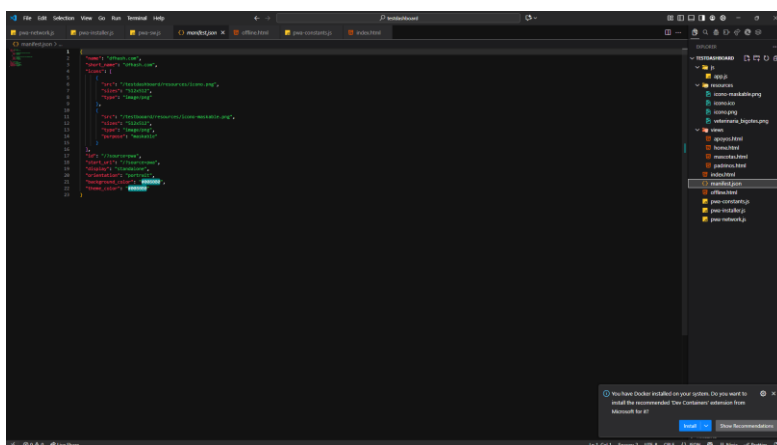


```
<meta name="theme-color" content="#008080">
<link rel="manifest" href="manifest.json">
<link rel="icon" href="/favicon.ico" sizes="128x128">
<link rel="shortcut icon" href="/favicon.ico">
<link rel="apple-touch-icon" href="/favicon.ico">
<!-- </A.anidura> -->
```



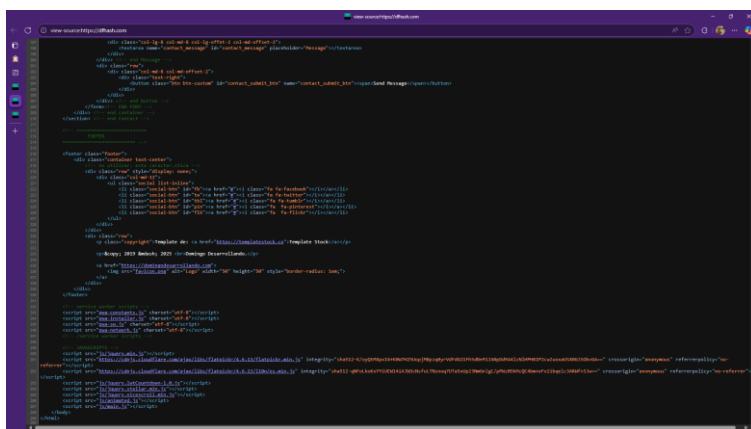
```
{
  "name": "dfhash.com",
  "short_name": "dfhash.com",
  "icons": [
    {
      "src": "/favicon.png",
      "sizes": "500x500",
      "type": "image/png"
    },
    {
      "src": "/favicon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    },
    {
      "src": "/favicon-maskable.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "maskable"
    }
  ],
  "id": "/?source-pwa",
  "start_url": "/?source-pwa",
  "display": "standalone",
  "orientation": "portrait",
  "background_color": "#008080",
  "theme_color": "#008080"
}
```


Este es el archivo con el cual manejaremos el manifest.json para la funcionalidad de la web app.



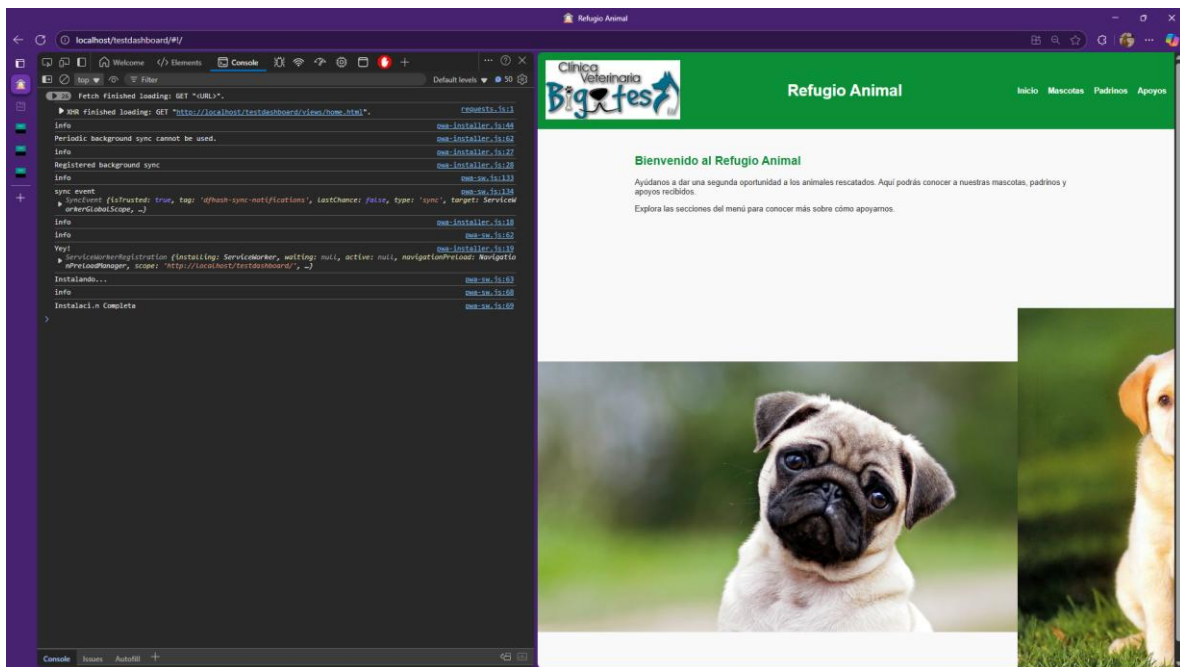
Dependiendo de como manejaras el icono cambiaremos las direcciones de los iconos agregando el nombre donde se encuentra el icono a partir de la carpeta donde esta la raiz del proyecto ademas de cambiar el nombre de la aplicación y el short name al del proyecto.

Now, obtendremos los archivos necesarios para la creacion de el service worwer y su funcionalidad que se encontrara en la misma pagina de dfhash.com al presionar Ctrl + U. y lo que obtendremos y necesitamos son especificamente los scripts que estah hasta abajo como en la imagen de abajo.



De preferencia todos los archivos relacionados con el service worker como el instalador y lo demas se tenga en la raiz del proyecto.

Nosotros presentamos un problema en el cual el archivo llamado pwa-sw.js no funcionaba en la raiz del proyecto sino que tenia que funcionar directamente en el localhost y asi resolvimos unos de los problemas principales, que no encontraba rutas y siempre presentaba “404 error not found” pero al final estableciendo correctamente las rutas de una mejor manera logramos obtener el resultado deseado



Que es que todos los archivos anteriores se descargaron en el cache y asi funcionar correctamente en la simulacion de offline.

Explicacion por Rosendo Jesus Gonzalez Espinoza.

FUNCIONALIDADES Y CAMBIOS EN LOS CODIGOS DEL SERVER WORKER.

Cada unos de los archivos obtenidos por la pagina dfhash.com que los archivos son:

- pwa-sw.js
- pwa-installer.js
- pwa-constants.js
- pwa-network.js

tiene una funcionalidad cada una que ayuda a que el archivo principal que es el que descarga de internet que es el pwa-sw.js funcione correctamente

PWA-NETWORK.JS

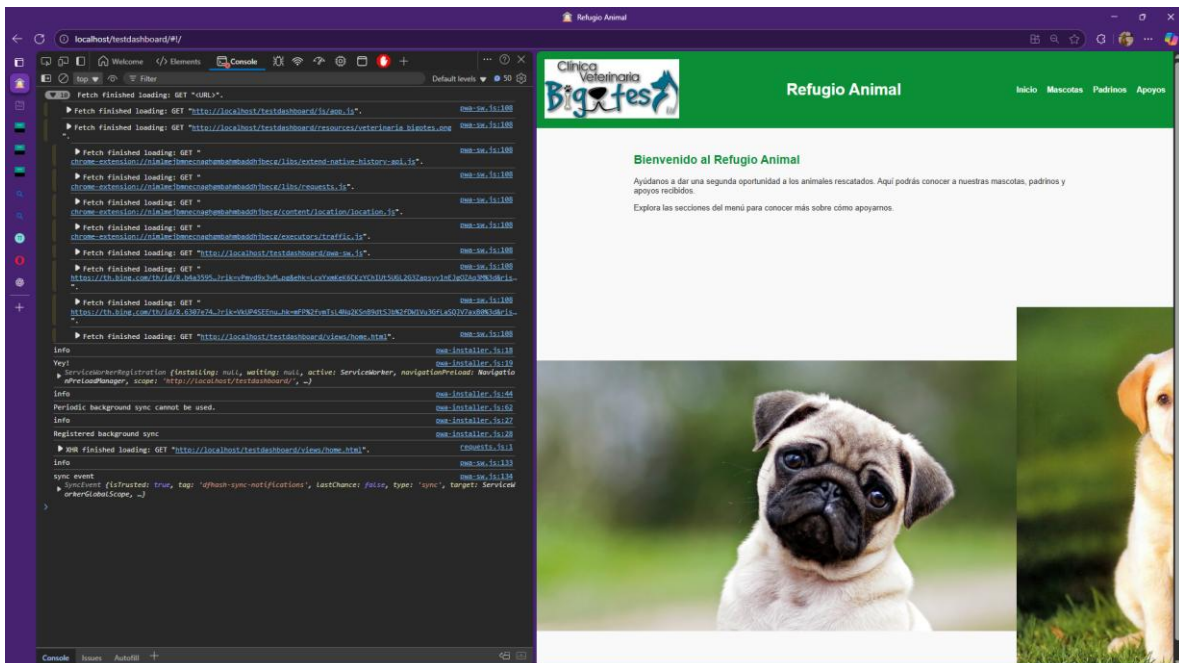
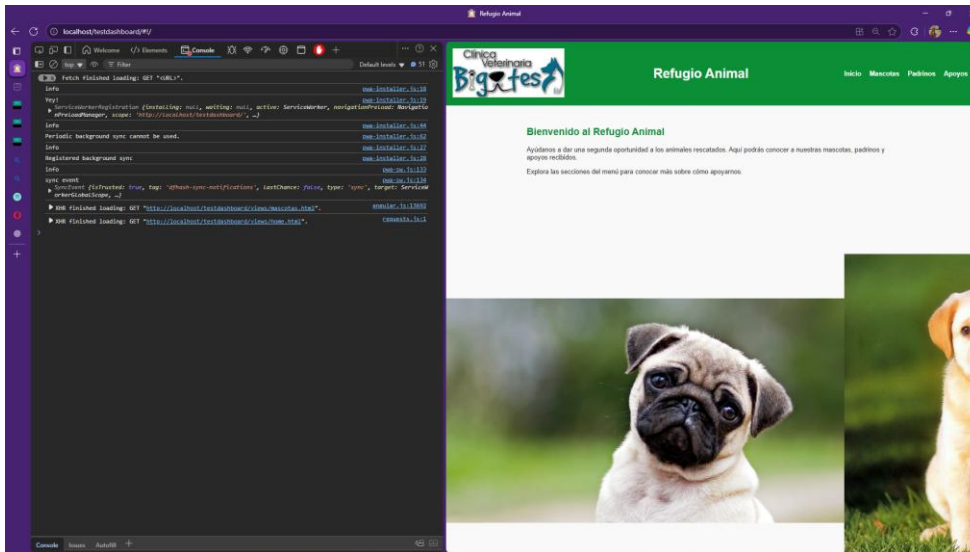
En este archivo no movemos nada realmente pero en pocas palabras es el archivo que tiene la verificacion de que estamos conectados a internet y permite ver si tiene la compatibilidad.

PWA-CONSTANTS.JS

Es una configuracion entre pestañas y el localStorage del navegador.

PWA-INSTALLER.JS

En pocas palabras llama al archivo pwa-sw.js que es el que tiene todo lo que debe descargarse en segundo plano y da mensajes en la consola para dar señalamiento de que todo lo asignado a pwa-sw.js este descargado en el cache correctamente y todo lo hace en segundo plano



En estas imágenes observamos como todo a dado correctamente a pesar de que la consola marca un error aun asi logro hacer la peticion fetch para descargar todo lo necesario y todo lo que se ha indicado en el pwa-sw.js

PWA-SWA.JS

Este es un script que maneja todo lo que la pagina descargara, como el background del installer.js y todos los archivos seran almacenados en el cache, esto mismo hace que la experiencia fuera de linea sea posible.

```
61 self.addEventListener("install", function (event) {
62     console.log("info")
63     console.info("Instalando...")
64
65     // this happens while the old version is still in control
66     event.waitUntil(
67         caches.open(PRECACHE_NAME).then(function (cache) {
68             console.log("info")
69             console.info("Instalaci.n Completa")
70
71             return cache.addAll([
72
73                 "https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-animate.min.js",
74                 "https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-route.min.js",
75                 "https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js",
76
77                 "https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css",
78                 "https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.4/font/bootstrap-icons.css",
79                 "https://code.jquery.com/jquery-3.7.1.min.js",
80
81                 "https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css",
82                 "https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.4/font/bootstrap-icons.css",
83                 "https://code.jquery.com/jquery-3.7.1.min.js",
84                 "https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js",
85
86
87                 url(),
88                 url("manifest.json"),
89                 url("?source=pwa"),
90                 url("resources/icono.ico"),
91                 url("resources/icono.png"),
92                 url("resources/icono-maskable.png"),
93
94                 OFFLINEURL,
95                 url("pwa-constants.js"),
96                 url("pwa-installer.js"),
97                 url("pwa-network.js"),
98                 url("offline.html"),
99                 // url("pwa-sw.js"),
100                 // url("pwa-i-loud.png"),
101                 // url("pwa-i-zap.png"),
102             ])
103         })
104     )
105 })
```

Esta parte del script adjunta todos los recursos de la pagina web para ser almacenados en el cache.

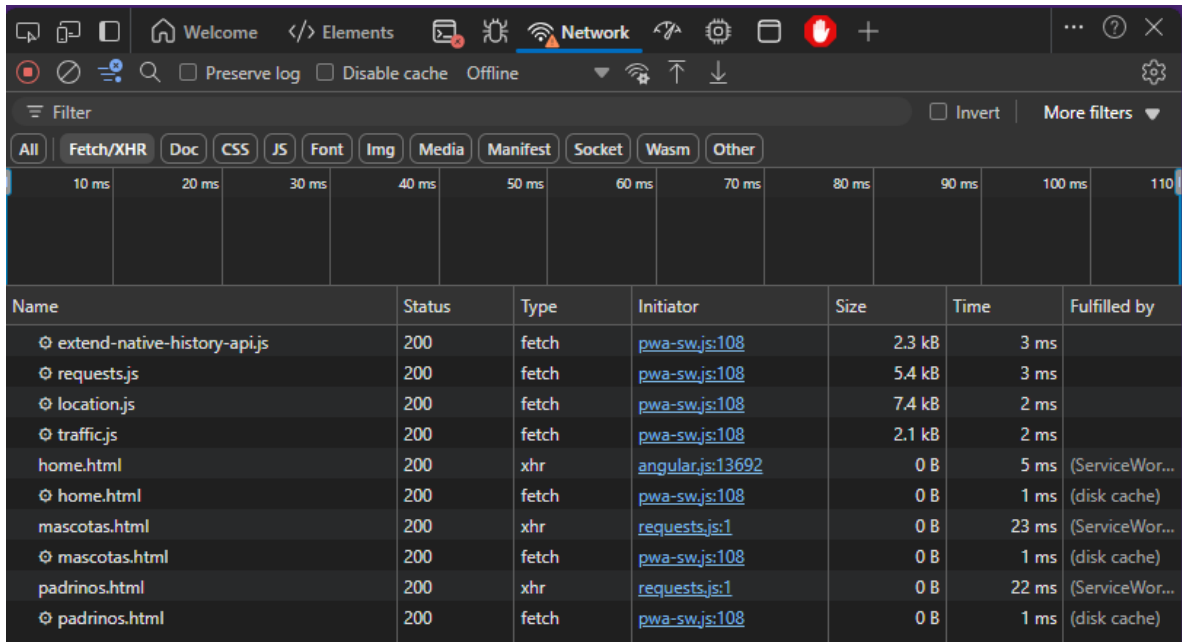
Dentro del script hay una configuracion adicional que presantara un mensaje dentro de la consola, para verificar el funcionamiento del script.

```
111 self.addEventListener("Fetch", function (event) {
112     event.respondWith(
113         caches.match(event.request)
114         .then(function (cachedResponse) {
115             return cachedResponse || fetch(event.request).catch(async function (err) {
116                 // Only call event.respondWith() if this is a navigation request
117                 // for an HTML page.
118                 if (event.request.mode === "navigate") {
119                     // catch is only triggered if an exception is thrown, which is
120                     // likely due to a network error.
121                     // If fetch() returns a valid HTTP response with a response code in
122                     // the 4xx or 5xx range, the catch() will NOT be called.
123                     console.log("error")
124                     console.error("Fetch failed; returning offline page instead.", err)
125
126                     var cache = await caches.open(PRECACHE_NAME)
127                     var cachedResponse = await cache.match(OFFLINE_URL)
128
129                     return cachedResponse
130                 }
131             })
132         })
133     ).catch(function (err) {
134         console.log("error")
135         console.error("Boo!", err)
136     })
137 })
138 })
```

La parte superior del código presenta la forma en que se declara la variable del cache para hacer uso del cache a la vez de que se declara la variable cachedResponse que hace que utilice los recursos guardados si esta fuera de linea

RESULTADOS:

Recursos de la pagina.



The screenshot shows the Chrome DevTools Network tab. At the top, there's a toolbar with icons for opening, refreshing, and other actions. Below the toolbar, there's a filter bar with tabs for 'All', 'Fetch/XHR', 'Doc', 'CSS', 'JS', 'Font', 'Img', 'Media', 'Manifest', 'Socket', 'Wasm', and 'Other'. The 'Fetch/XHR' tab is selected. Below the filter bar, there's a timeline view showing the loading of resources over time. The timeline is divided into segments representing different resources. Below the timeline, there's a table with the following columns: Name, Status, Type, Initiator, Size, Time, and Fulfilled by.

Name	Status	Type	Initiator	Size	Time	Fulfilled by
⊗ extend-native-history-api.js	200	fetch	pwa-sw.js:108	2.3 kB	3 ms	
⊗ requests.js	200	fetch	pwa-sw.js:108	5.4 kB	3 ms	
⊗ location.js	200	fetch	pwa-sw.js:108	7.4 kB	2 ms	
⊗ traffic.js	200	fetch	pwa-sw.js:108	2.1 kB	2 ms	
home.html	200	xhr	angular.js:13692	0 B	5 ms	(ServiceWor...
⊗ home.html	200	fetch	pwa-sw.js:108	0 B	1 ms	(disk cache)
mascotas.html	200	xhr	requests.js:1	0 B	23 ms	(ServiceWor...
⊗ mascotas.html	200	fetch	pwa-sw.js:108	0 B	1 ms	(disk cache)
padrinos.html	200	xhr	requests.js:1	0 B	22 ms	(ServiceWor...
⊗ padrinios.html	200	fetch	pwa-sw.js:108	0 B	1 ms	(disk cache)

Muestra los recursos que fueron cargados de la pagina.

Consola.

```
info pwa-installer.js:18
Yey! pwa-installer.js:19
▼ ServiceWorkerRegistration {installing: null, waiting: null, active: ServiceWorker, navigationPreload: Navigat
  ionPreloadManager, scope: 'http://localhost/testdashboard/', ...} ⓘ
  ► active: ServiceWorker {scriptURL: 'http://localhost/testdashboard/pwa-sw.js', state: 'activated', onstatech
  ► backgroundFetch: BackgroundFetchManager {}
  ► cookies: CookieStoreManager {}
    installing: null
  ► navigationPreload: NavigationPreloadManager {}
    onupdatefound: null
  ► paymentManager: PaymentManager {userHint: ''}
  ► periodicSync: PeriodicSyncManager {}
  ► pushManager: PushManager {}
    scope: "http://localhost/testdashboard/"
  ► sync: SyncManager {}
    updateViaCache: "imports"
    waiting: null
  ► [[Prototype]]: ServiceWorkerRegistration
◀────────────────────────────────────────────────────────────────────────────────────────────────────────────────▶
info pwa-installer.js:44
Periodic background sync cannot be used. pwa-installer.js:62
info pwa-installer.js:27
Registered background sync pwa-installer.js:28
```

Esta muestra un mensaje que muestra el registro de instalacion.

servicio.

Service workers

☒ Offline ☐ Update on reload ☐ Bypass for network


http://localhost/t...

Network requests

Update

Unregister

Source

pwa-sw.js  4
Received 9/17/2025, 2:28:04 PM

Status

#538 activated and is running

Stop

Push

Test push message from DevTo

Push

Sync

test-tag-from-devtools

Sync

Periodic sync

test-tag-from-devtool:

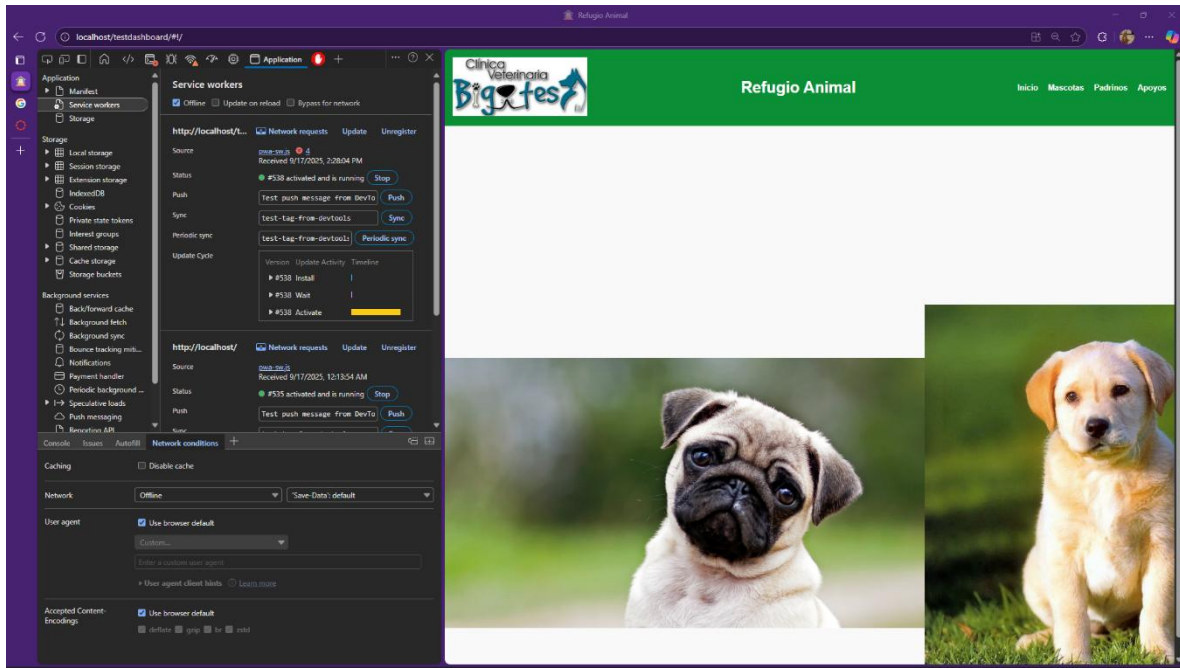
Periodic sync

Update Cycle

Version	Update Activity	Timeline
▶ #538	Install	
▶ #538	Wait	
▶ #538	Activate	<div></div>

La forma en que se desactiva el uso del internet para ver la funcionalidad.

Funcionamiento de la pagina.



Este muestra el resultado que funciona el service worker.

CONCLUSIONES.

Aprendimos lo esencial del manejo de la aplicación de manera online y offline y con el PWA y la funcionalidad de cada una y aprendimos un poco del ciclo de vida del PWA ademas de entender a detalle el metodo ademas de utilizar de mejor manera los DevTools de los navegadores.