IIC3585-1

Web Assembly y PWA {

<Grupo 3>

}

DEMO


Editor de Fotos
Selecciona una imagen
Seleccionar archivo
Sin archivos seleccionados
Selecciona un filtro
Blanco y Negro
Desenfocar
Oscurecer
Visualizar bordes
aclarar
Restablecer
Original
Editada

Componentes

01 Web Assembly

02 PWA

03 Service Worker

04 Push Notifications

05 IndexedDB

Implementación de WASM

Implementación de WASM

Emscripten

Implementación de WASM

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <emscripten.h>

typedef struct {
    uint8_t r, g, b;
} RGB;
```

Funciones de entrada/salida

Definir tipos con tamaños

Manejo de memoria

Manipulación de strings

Implementación de WASM

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <emscripten.h>

typedef struct {
    uint8_t r, g, b;
} RGB;
```

Permite exportar funciones y trabajar con el entorno WASM

Implementación de WASM

01 Difuminar

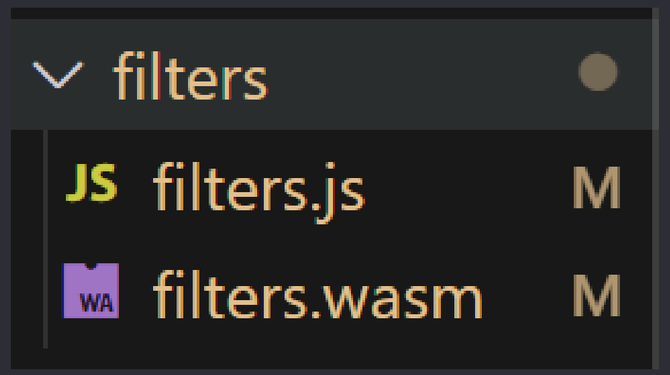
02 Aclarar

03 Oscurecer

04 Resaltar bordes

```
EMSCRIPTEN_KEEPALIVE
void apply_blur(RGB *image, int width, int height) {
```

Implementación de WASM



En main.js importamos el módulo de WASM llamando a filters.js

Podemos acceder a las funciones de C exportadas usando Module.cwrap()

Implementación de PWA

> public > {} manifest.json > ...

```
{
    "name": "Editor de Imágenes WASM",
    "short_name": "ImgWASM",
    "start_url": "./",
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#0f172a",
    "icons": [
      {
        "src": "/icon-192x192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/icon-512x512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ]
}
```

Manifiesto

Se define la información escencial de la PWA en maniest.json

```
import { defineConfig } from 'vite'
import { VitePWA } from 'vite-plugin-pwa'
export default defineConfig({
  plugins: [
    VitePWA({
      registerType: 'autoUpdate',
      //Usamos nuestro propio SW
      strategies: 'injectManifest',
      // La carpeta donde vive el SW:
      srcDir: 'src/pwa',
      // y el nombre final:
      filename: 'service-worker.js',
      injectManifest: {
        swSrc: 'service-worker.js'
      },
      manifest: {
        name: 'Editor de Imágenes WASM',
        short_name: 'ImgEditor',
        start_url: '/',
        display: 'standalone',
        background_color: '#ffffff',
        theme_color: '#0f172a',
        icons: [
```

Complemento Vite PWA

Se importa VitePWA

Definimos el uso de nuestro propio Service Worker

Llamamos el manifiesto

Service Worker

VitePWA + Workbox

Estrategia Cache First

```
import { precacheAndRoute } from 'workbox-precaching';
precacheAndRoute(self.__WB_MANIFEST);
```

Evento install → Precacheo → Funcionamiento offline

```
workbox: {

  globPatterns: ['**/*.{js,css,html,png,svg,wasm}'],
  runtimeCaching: []
},
```

Peticiónes al Backend

```
// ✅ Network only para las peticiones de la API
self.addEventListener('fetch', event => {
  // No hace nada para peticiones fetch, evita errores con POST/PUT
});
```

Push Notifications

Push Service

Browser

Permitir Notificaciones

Subscription Object

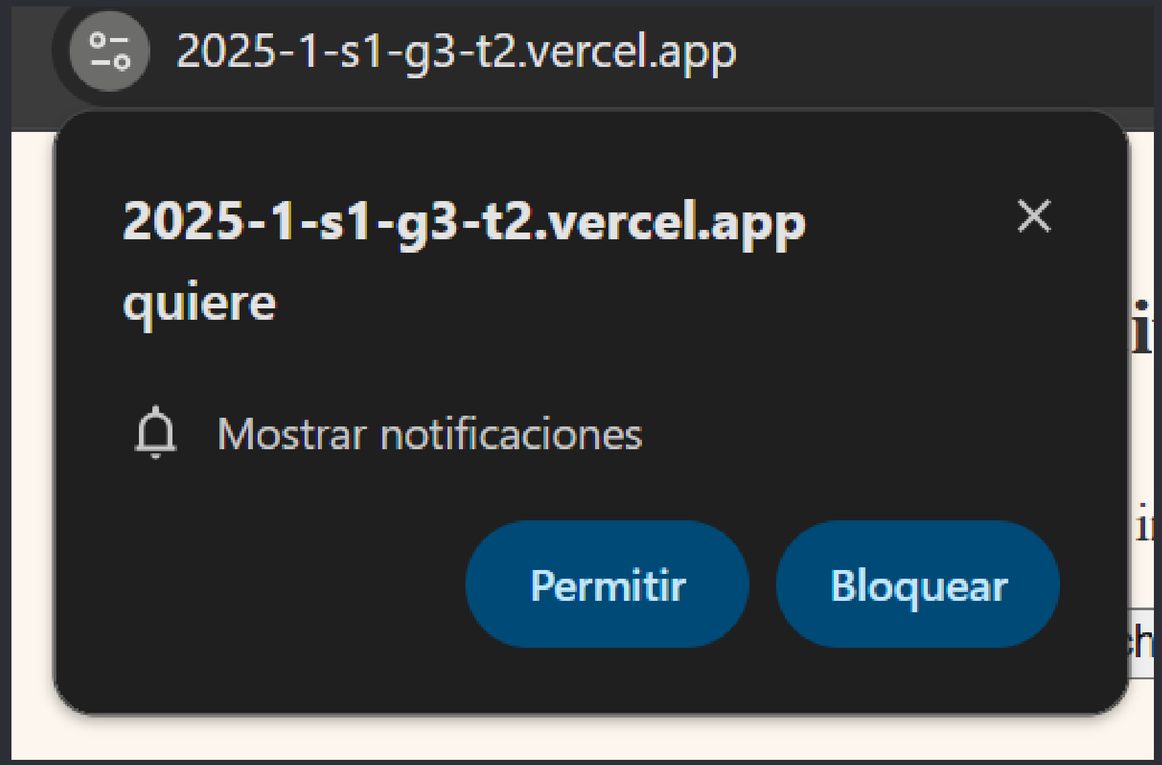
Subscription Object

Server

cliente

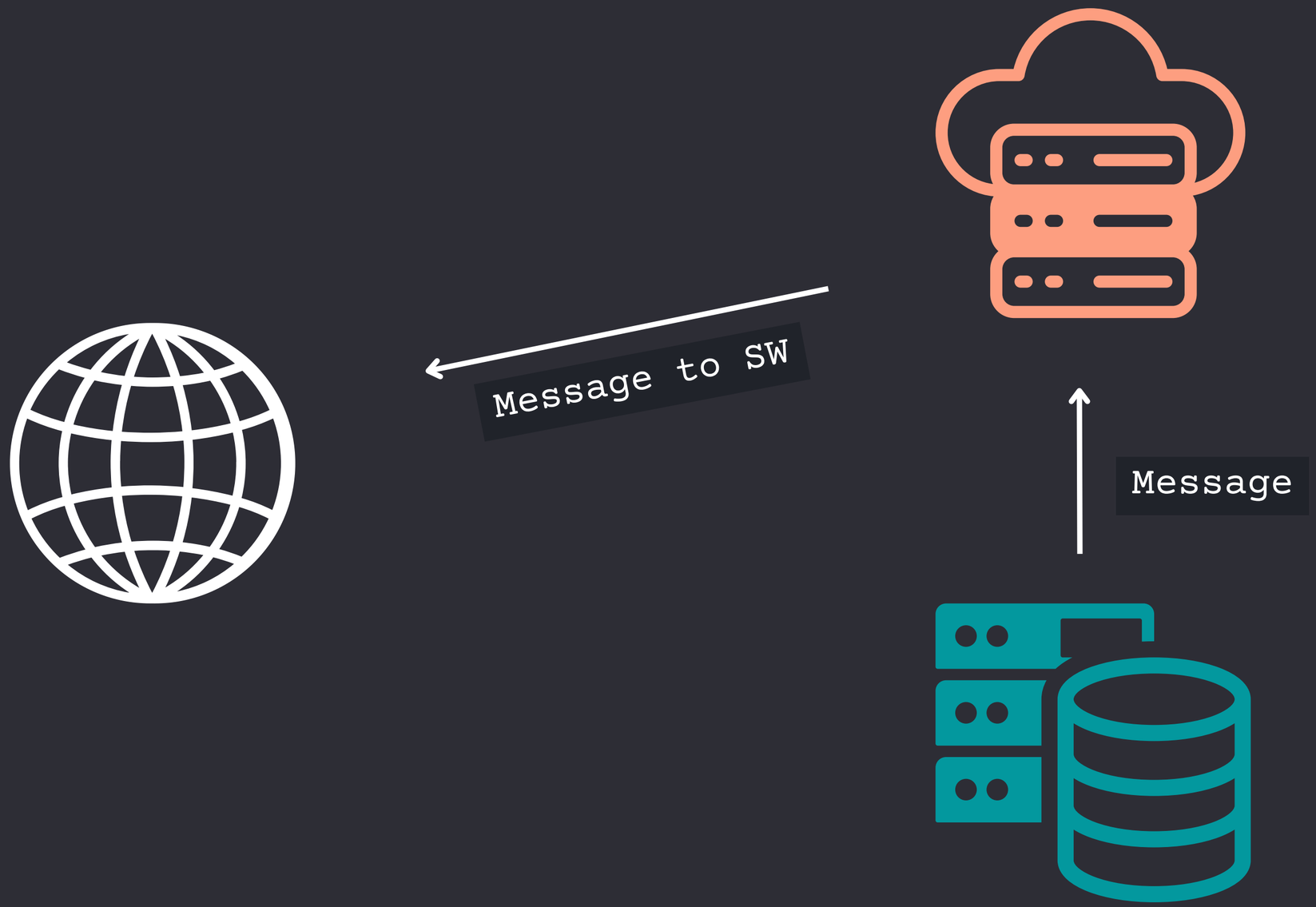
```
window.addEventListener("load", async () => {
  if ("Notification" in window) {
    if (Notification.permission !== "granted") {
      const permission = await Notification.requestPermission();
      if (permission !== "granted") return;
    }

    console.log("✅ Notificaciones activadas automáticamente");
    await subscribeUserToPush();
  }
});
```

cliente


2025-1-s1-g3-t2.vercel.app
2025-1-s1-g3-t2.vercel.app
quiere
Mostrar notificaciones
Permitir
Bloquear

cliente

```
export async function subscribeUserToPush() {
  try {
    const registration = await navigator.serviceWorker.ready;
    const subscription = await registration.pushManager.subscribe({
      userVisibleOnly: true,
      applicationServerKey: urlBase64ToUint8Array(publicKey),
    });

    console.log('Suscripción exitosa:', JSON.stringify(subscription));
    const response = await saveSubscription(subscription);
  } catch (err) {
    console.error('Error al suscribirse: ', err);
  }
}
```


Message to SW
Message

servidor

```
app.get("/send-notification", (req, res) => {
  const payload = {
    title: "🔔 Recordatorio",
    body: "Acuérdate de modificar tus imágenes antes de publicarlas",
    icon: "/icon-192x192.png",
    badge: "/icon-192x192.png"
  };
  enviarNotificacion(payload, res, "recordatorio");
});
```

```
try {
  await webpush.sendNotification(suscriptor, JSON.stringify(payload));
```

service-worker.js

```
self.addEventListener('push', event => {
  console.log('[SW] Push recibido');

  const data = event.data ? event.data.json() : {
    title: "🔔 Notificación",
    body: "¡Tienes una nueva notificación push!",
  };

  const options = {
    body: data.body,
    icon: '/icon-192x192.png',
    badge: '/icon-192x192.png'
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
});
```

Web-push {

01 Web Push Protocol

02 Gestión VAPID KEYS

03 Codifica y cifra los mensajes

04 Conexion push service del navegador

https://www.npmjs.com/package/web-push

}

IndexedDB

utils/db.js

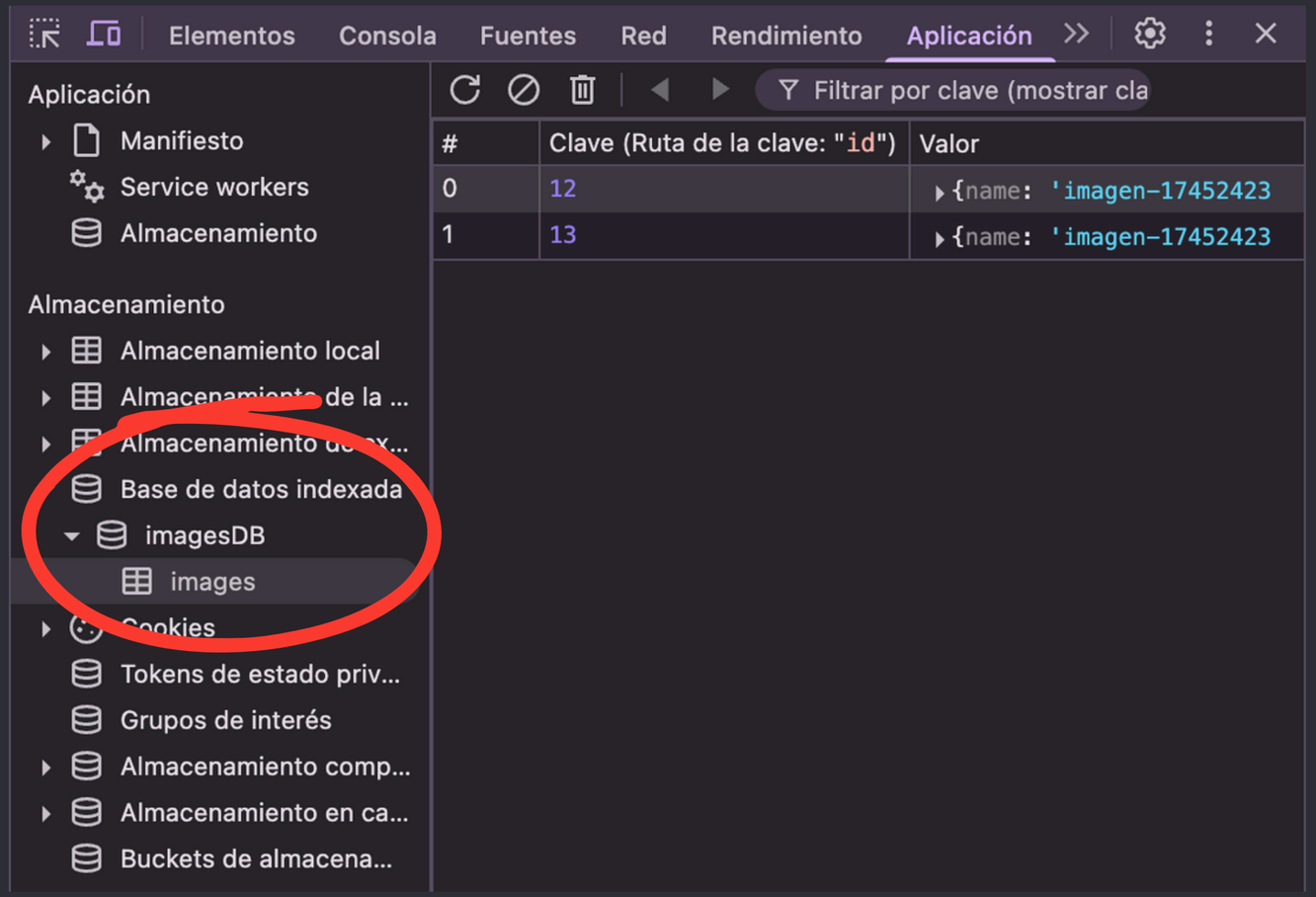
```
import { openDB } from 'idb';

export const initDB = async () => {
  return await openDB('imagesDB', 1, {
    upgrade(db) {
      if (!db.objectStoreNames.contains('images')) {
        db.createObjectStore('images', { keyPath: 'id', autoIncrement: true });
      }
    }
  });
};

export const saveImageToDB = async (name, blob) => {
  const db = await initDB();
  await db.add('images', {
    name,
    date: new Date(),
    blob
  });
};

export const getAllImages = async () => {
  const db = await initDB();
  return await db.getAll('images');
};

export const clearImagesFromDB = async () => {
  const db = await initDB();
  const tx = db.transaction('images', 'readwrite');
  await tx.objectStore('images').clear();
  await tx.done;
};
```

main.js

```
import { saveImageToDB, getAllImages, clearImagesFromDB } from './utils/db.js';
```

```
saveBtn.onclick = async () => {
  if (!imageDataWorking) return;

  const exportCanvas = document.createElement("canvas");
  exportCanvas.width = scaledWidth;
  exportCanvas.height = scaledHeight;
  const exportCtx = exportCanvas.getContext("2d");
  exportCtx.putImageData(imageDataWorking, 0, 0);

  exportCanvas.toBlob(async (blob) => {
    if (blob) {
      await saveImageToDB(`imagen-${Date.now()}`, blob);
      showSavedImages();
      showNotification("✅ Imagen guardada correctamente");
    }
  }, "image/png");
};

clearBtn.onclick = async () => {
  await clearImagesFromDB();
  showNotification("🗑️ Imágenes eliminadas correctamente");
  showSavedImages();
};

async function showSavedImages() {
  const images = await getAllImages();
  savedList.innerHTML = "";
  images.forEach(item => {
    const url = URL.createObjectURL(item.blob);
    const img = document.createElement("img");
    img.src = url;
    img.alt = item.name;
    savedList.appendChild(img);
  });
}
```

DevTools


Elementos
Consola
Fuentes
Red
Rendimiento
Aplicación
Aplicación
Manifiesto
Service workers
Almacenamiento
Almacenamiento
Almacenamiento local
Base de datos indexada
imagesDB
images
Tokens de estado priv...
Grupos de interés
Almacenamiento comp...
Almacenamiento en ca...
Buckets de almacena...
Filtrar por clave (mostrar cla
| Clave (Ruta de la clave: "id") | Valor
0 | 12 | {name: 'imagen-17452423
1 | 13 | {name: 'imagen-17452423

IIC3585-1

Gracias {

<Grupo 3>

}