
Progressive Web Application

Experiencias con ServiceWorkers

Grupo 4

Natalia Barra Marchant

Luis Chodiman Herrera

Mauricio Ortiz Angel

Nuestra aplicación

<https://quacker-g4.firebaseio.com>

Descripción del problema

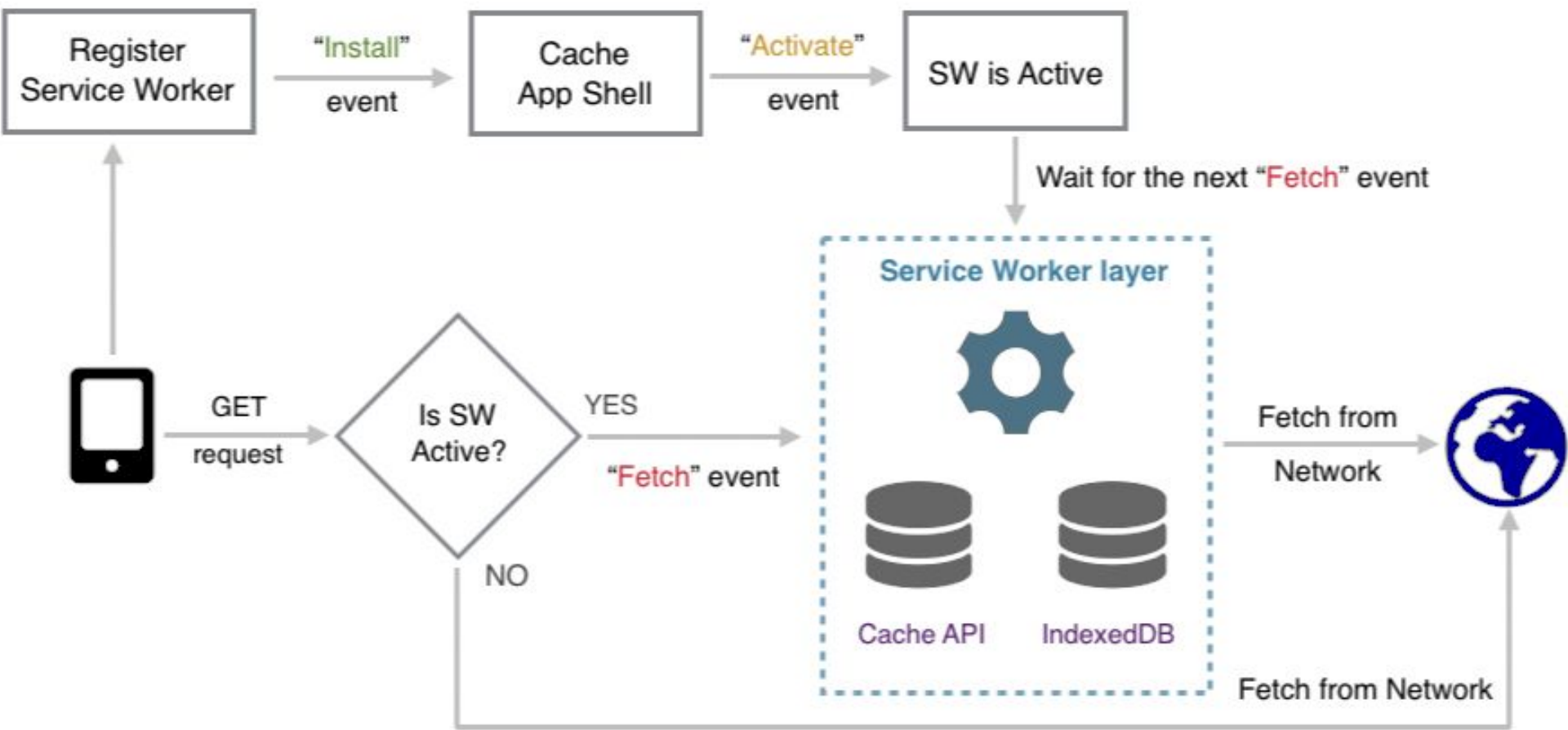
Experimentar con las tecnologías asociadas a las aplicaciones web progresivas (PWA), cumpliendo con los siguientes requisitos:

- Funcionalidad Offline
 - Notificaciones Push
 - Instalable y Fullscreen
-

Aplicación desarrollada

“Twitter” con funcionalidad limitada: creación de posts

- Base de datos (backend): Firebase
 - Caché estático y dinámico
 - Sincronización en segundo plano
 - Caché IndexedDB
 - Google Cloud Functions (API, posts storage)
-



Base de datos: Firebase

The screenshot displays the Firebase console interface. On the left is a dark sidebar with navigation options: Project Overview, Desarrolla (with sub-items Authentication, Database, Storage, Hosting, Functions, ML Kit), Calidad (Crashlytics, Performance, Test Lab), Estadísticas, and Spark. The main area has a blue header with the project name 'quacker-g4', a link to documentation, and a user profile. Below the header, the 'Database' section is active, showing 'Realtime Database'. A sub-header contains tabs for 'Datos', 'Reglas', 'Copias de seguridad', and 'Uso'. The 'Datos' tab is selected, displaying a tree view of the database structure. The root node is 'quacker-g4', which contains a child node 'posts'. The 'posts' node is expanded, showing a list of 12 data points, each with a unique key and a plus icon for expansion.

quacker-g4

- posts
 - Ld4YYQ0w1OeSGKuTmI7
 - Ld4YYzMiiXLF2EryN3J
 - Ld4bMJnxfL5EnaJlbFq
 - Ld4bgdlaLhKSwgG_F-m
 - Ld4bhGv3eHI4h2fOU1t
 - Ld4bmB1-swALEYv6GSK
 - Ld4bmMF10R1QlxWjKNU
 - Ld4bmVd3MvgR8ylBaf0
 - Ld4bmYbO_DXoA3rb-hY

Caché

- Estático: archivos base de la aplicación (html, js, css, images) -> app shell
 - Dinámico: elementos creados durante el funcionamiento de la app (comentarios) y otros
 - Ej: Help page
-

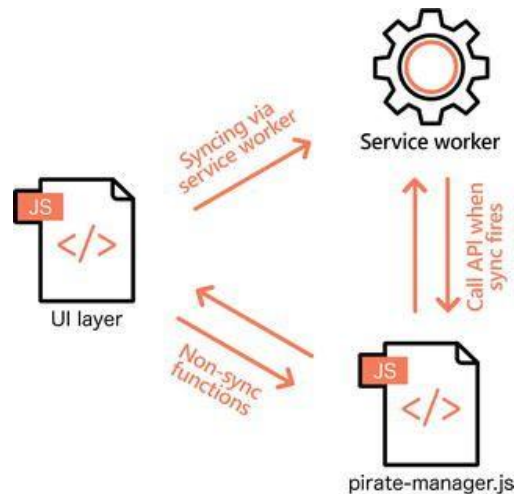
IndexDB

2 Storages

- posts: guardamos lo recibido del fetch de la API (Firebase)
 - sync-posts: guardamos temporalmente para enviar al estar online
-

Sincronización en segundo plano

- Si se hace un post estando offline, aplicación la “guarda” y la envía al backend cuando se vuelva a conectar



Google Cloud Functions

```
exports.storePostData = functions.https.onRequest((request, response) => {
  cors(request, response, () => {
    admin
      .database()
      .ref("posts")
      .push({
        id: request.body.id,
        title: request.body.title,
        location: request.body.location,
        image: request.body.image
      })
      .then(() => {
        response
          .status(201)
          .json({ message: "Data stored", id: request.body.id });
      })
      .catch(err => {
        response.status(500).json({ error: err });
      });
  });
});
```

Funcionalidades pendientes

- Acceder al hardware del dispositivo mobile (GPS, cámara)
 - Push Notifications desde la API usando PUB/SUB
-