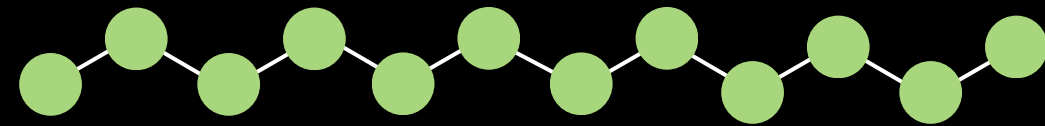


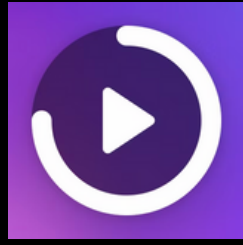
# TAREA 4

Diseño avanzado de aplicaciones web



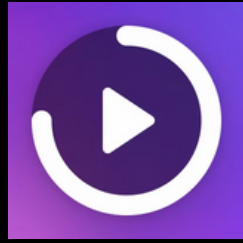
Grupo 3

# HELLO TIMO




- Crear, manejar y gestionar tus proyectos y tareas.
- Puedes crear nuevas tareas junto con un tiempo estimado de duración.
- Puedes ejecutar este tiempo estimado mientras trabajas en ella y Hello Timo te notificará del término de esta.

# HELLO TIMO



Todos

 Overview

Projects

☐ Account

**22 May 23**

Day ▼

Filter

< [ ] >

**22 Mon**

2 todos • 0h 15m

☒ Tarea de Diseño Avanzado de Aplicaciones Web

☐ No project

▶ 00:10:00 ⋮

- Tarea de Reconocimiento de Patrones

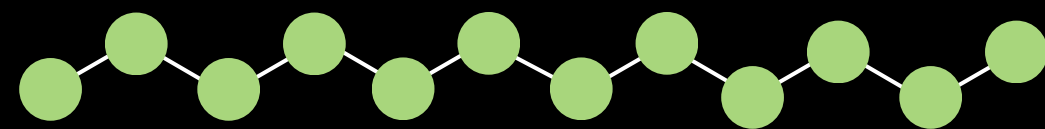
☐ No project

00:05:00

+ New todo



# DEMO



Grupo 3

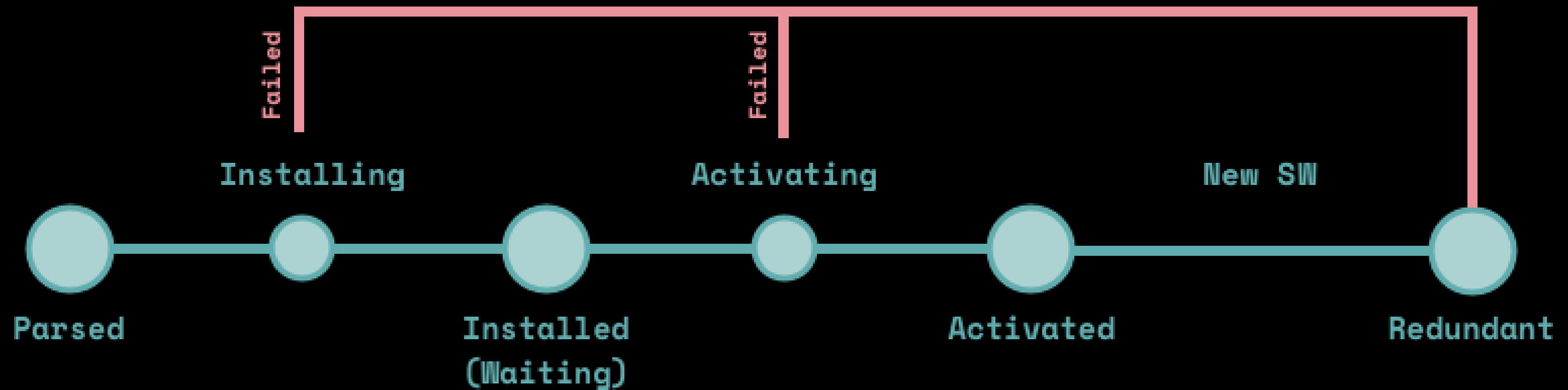
# MANIFEST



```
1  {
2    "name": "Hello Timo",
3    "icons": [
4      {
5        "src": "../images/icon.svg",
6        "sizes": "512x512"
7      }
8    ],
9    "short_name": "HelloTimo",
10   "start_url": "../index.html",
11   "display": "standalone",
12   "background_color": "#000000",
13   "theme_color": "#000000"
14 }
```

Informa al navegador sobre la PWA y cómo debe proceder cuando se instala en el escritorio o dispositivo móvil

# SERVICE WORKER - CICLO DE VIDA



FUENTE: [HTTPS://BITSOFCO.DE/THE-SERVICE-WORKER-LIFECYCLE/](https://bitsofco.de/the-service-worker-lifecycle/)

# SERVICE WORKER

```
1 window.addEventListener('load', () => {
2   registerSW();
3 });
4
5 // Register the Service Worker
6 async function registerSW() {
7   if ('serviceWorker' in navigator) {
8     try {
9       const reg = await navigator.serviceWorker.register('./serviceWorker.js');
10      console.log('Service worker registered! 🥳', reg);
11    }
12    catch (e) {
13      console.log('SW registration failed');
14    }
15  }
16 }
```

app.js

serviceWorker.js

```
1 self.addEventListener("install", function (e) {
2   e.waitUntil(
3     caches.open(staticCacheName).then(function (cache) {
4       return cache.addAll(filesToCache);
5     })
6   );
7 });
8
9 self.addEventListener('activate', async (event) => {
10   console.log('activating!');
11   const keys = await caches.keys();
12   keys
13     .filter((key) => key !== staticCacheName)
14     .map((key) => event.waitUntil(caches.delete(key)));
15 });
```

# CACHÉ



```
1  async function networkFirst(request) {  
2    const cache = await self.caches.open(staticCacheName);  
3    try {  
4      const response = await fetch(request);  
5      cache.put(request, response.clone());  
6      return response;  
7    } catch {  
8      return cache.match(request);  
9    }  
10 }  
11  
12 self.addEventListener("fetch", function (event) {  
13   if (event.request.method !== "GET") return;  
14   event.respondWith(networkFirst(event.request));  
15 });
```



# FIREBASE



## REALTIME DATABASE



```
1  const admin = require('firebase-admin');
2  const serviceAccount = require("../serviceAccountKey.json");
3
4  admin.initializeApp({
5    credential: admin.credential.cert(serviceAccount),
6    databaseURL: 'https://pwa-db-t4-default-rtdb.firebaseio.com/',
7  });
8
9  const express = require('express');
10 const { request } = require('express');
11 const app = express();
12 const port = 9000;
13 const cors = require('cors');
14
15 app.use(express.json());
16 app.use(cors());
```



```
1  app.get('/times', (req, res) => {
2    admin
3      .database()
4      .ref('times')
5      .once('value')
6      .then((snapshot) => {
7        const entries = Object.entries(snapshot.val());
8        const times = entries.map((entry) => {
9          return {
10             id: entry[0],
11             ...entry[1],
12           };
13        });
14        return times.sort((a, b) => {
15          return new Date(b.time) - new Date(a.time);
16        });
17      }).then((times) => {
18        res.header('Access-Control-Allow-Origin', '*')
19        res.send(times)
20      });
21  });
```

# NOTIFICACIONES 🔔



```
1  function showNotification(title, message) {  
2    return new Notification(title, {  
3      body: message,  
4      icon: './images/icon.svg',  
5    });  
6  }
```



```
1  function showNotification(title, message) {  
2    navigator.serviceWorker.ready.then((registration) => {  
3      registration.showNotification(title, {  
4        body: message,  
5        icon: './images/icon.svg',  
6        vibrate: [200, 100, 200, 100, 200, 100, 200]  
7      });  
8    });  
9  }
```

# NOTIFICACIONES



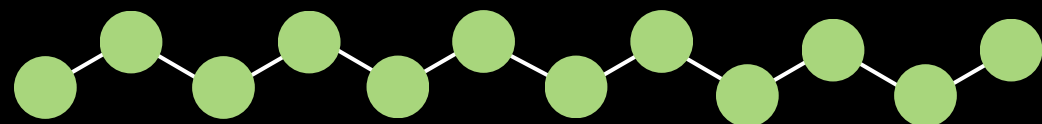
```
1 function showNotification(title, message) {  
2   return new Notification(title, {  
3     body: message,  
4     icon: './images/icon.svg',  
5   });  
6 }
```



```
1 function showNotification(title, message) {  
2   navigator.serviceWorker.ready.then((registration) => {  
3     registration.showNotification(title, {  
4       body: message,  
5       icon: './images/icon.svg',  
6       vibrate: [200, 100, 200, 100, 200, 100, 200]  
7     });  
8   });  
9 }
```

# APRENDIZAJES

- Una PWA es una forma rápida y fácil de tener aplicaciones en **todos los dispositivos**.
- Experiencia de uso similar a una aplicación nativa usando **solo Javascript**, sin la necesidad de aprender los lenguajes nativos.
- Mantenimiento y actualizaciones sincronizadas a todos los dispositivos. (No es necesario usar App Stores)





Grupo 3