PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC3585 - Sección 2 — Diseño Avanzado de Aplicaciones Web

Tarea 2

Actualización: 10 de abril de 2025

Entrega

- **Fecha y hora**: Lunes 21 de abril del 2025, a las 22:00
- **Lugar:** Repositorio grupal para la evaluación en GitHub del curso, rama `main`.

Objetivos

- **Emplear** las últimas herramientas y tecnologías para el desarrollo de aplicaciones web.
- **Construir** interfaces avanzadas con elementos modernos de HTML y CSS.
- **Integrar** principios de diseño y usabilidad para crear experiencias de usuario intuitivas y eficientes.
- **Evaluar** críticamente nuevas tecnologías y tendencias emergentes en el desarrollo de aplicaciones web.

Descripción

En esta tarea, deberán construir una **Aplicación Web Progresiva (PWA)** que implemente funcionalidades avanzadas del ecosistema web moderno, tales como notificaciones *push*, APIs nativas del navegador y capacidades *offline*.

La idea es que la aplicación cumpla con los requisitos mínimos para ser una PWA (archivo manifest, *service worker* y funcionamiento *offline*), además de incorporar **notificaciones *push* como funcionalidad obligatoria**, y al menos **una funcionalidad adicional** entre las que se detallan en la siguiente sección.

Cada grupo deberá escoger uno de los cinco temas propuestos como base para su aplicación. Esta aplicación deberá tener tanto un **front-end en HTML, CSS y JavaScript (sin framework)**, como un **back-end que permita gestionar sus propias notificaciones *push***. Se recomienda utilizar Vite para el entorno de desarrollo y Tailwind CSS para los estilos.

1. Especificaciones

La aplicación debe cumplir con los siguientes requisitos mínimos:

- **Debe ser una PWA**: Esto implica incluir un `manifest.json`, un *service worker* funcional y soporte *offline* al menos para una funcionalidad. La aplicación debe poder ser instalada tanto en el computador como en el celular.
- **Notificaciones *push***: La aplicación debe ser capaz de enviar notificaciones de escritorio y al centro de notificaciones del celular.

    - Las notificaciones deben ser entregadas mediante el uso de la **Push API** y/o la **Notifications API**.
    - La suscripción y el envío deben estar gestionadas por su propio back-end. Pueden elegir la tecnología que deseen para crear el back-end.

- **Una funcionalidad extra (obligatoria)**: Además de notificaciones, deberán implementar una funcionalidad adicional que aproveche capacidades avanzadas del navegador. Las APIs entre las que podrán elegir para implementar esta funcionalidad son:

    - **WebSockets**
    - **WebRTC**
    - **Server-Sent Events**
    - **Geolocation API**
    - **MediaDevices API**
    - **Web Share API**
    - **Device Orientation Events**

    ¡Sean creativos! Si utilizan al menos 2 Web APIs adicionales, tendrán un **bonus de 0,5 puntos**. **Importante:** sólo se contarán las APIs que tengan sentido como funcionalidades adicionales dentro del contexto de la aplicación.

- **Debe funcionar *offline***: Se espera que al menos el uso básico de la app sea posible sin conexión, mediante el uso de **Cache API**, **indexedDB** u otras soluciones que deseen utilizar.

- **Front-end sin frameworks**: No se permite el uso de frameworks como React, Vue o Angular. Sí está permitido el uso de **Vite** como empaquetador y el uso de **Tailwind CSS** para los estilos.

Temas disponibles

A continuación se proponen 5 temas que pueden desarrollar. Cada grupo debe elegir uno de estos como base para su aplicación:

1. **Mini red social**: Usuarios pueden postear, comentar y recibir notificaciones cuando hay actividad.
2. **Chat geolocalizado**: Una app de mensajería donde los usuarios solo pueden ver y chatear con otros cerca de su ubicación.
3. **Galería compartida**: Usuarios pueden subir y ver fotos o videos, compartirlos y recibir notificaciones cuando se suben nuevos archivos.
4. **App de emergencias vecinales**: Permite emitir alertas a vecinos cercanos, mostrando su ubicación en un mapa.
5. **Videollamada *express***: App para iniciar llamadas rápidas entre pares.

2. README

En el repositorio debe haber un documento `README.md` que especifique la distribución del trabajo entre los miembros del grupo. Además, deberán listar todas las Web APIs utilizadas y la funcionalidad con la que cada una está asociada.

Finalmente, deberán declarar y especificar el uso de IA, tal como se ejemplifica en las diapositivas de la primera clase del curso.

3. Presentaciones

Se seleccionará de forma aleatoria un número de grupos que deberá presentar su aplicación. La presentación deberá incluir:

- **Demo en vivo** de la aplicación funcionando tanto online como *offline*. Además, deben ser capaces de mostrar la instalación de la aplicación en el celular.
- **Distribución del trabajo** entre los miembros del grupo.
- **Justificación técnica** de las decisiones de arquitectura e implementación.
- **Diagramas** que expliquen el flujo de la aplicación, especialmente cómo se gestionan las notificaciones y el manejo *offline*.
- **Descripción de la API adicional implementada**, incluyendo problemas encontrados y cómo los resolvieron.

4. Rúbrica

A continuación, se describe el criterio de cada uno de los ítems de la rúbrica con la que se evaluará esta entrega.

- **Funcionalidad como PWA [2 puntos]:** Se evaluará que la aplicación esté correctamente configurada como una Progressive Web App, incluyendo un `manifest.json` válido y un Service Worker funcional. La aplicación debe poder instalarse y contar con al menos una funcionalidad disponible en modo *offline*.
- **Notificaciones Push [1.5 puntos]:** Se revisará que el flujo completo de suscripción y recepción de notificaciones en el navegador funcione correctamente. El envío de las notificaciones debe ser gestionado desde su propio back-end, sin depender completamente de plataformas externas.
- **Funcionalidad extra implementada [1 punto]:** Uso correcto y funcional de alguna de las APIs avanzadas presentadas en la sección anterior, considerando tanto su complejidad como su integración con la experiencia general de la aplicación.
- **Diseño y experiencia de usuario [1 punto]:** Se espera que la interfaz de la aplicación sea clara, **responsiva** para escritorio y móvil y que la navegación entre sus distintas vistas o componentes sea fluida y coherente.
- ***Gitflow* adecuado [0.5 puntos]**: Se evaluará el uso correcto de *gitflow* para el agregado de *features* en la tarea. En particular, se espera el uso de **PRs** y ***Code Review* entre los miembros del equipo**.

La nota asociada a la implementación $\mathbf{NI}$ se calculará a partir del puntaje obtenido en los ítems anteriores, de la siguiente manera:

$$\mathbf{NI} = 1 + \sum \text{Puntaje obtenido}$$

En el caso de que el grupo sea seleccionado para presentar, su nota en la tarea corresponderá a una ponderación de la nota $\mathbf{NI}$ y la nota de presentación $\mathbf{NP}$. En caso contrario, la nota de la tarea corresponderá solo a la nota $\mathbf{NI}$. Por lo tanto, la nota de la evaluación $\mathbf{T_1}$ se calculará como:

$$\mathbf{T_1} = \begin{cases} 0{,}75 \times \mathbf{NI} + 0{,}25 \times \mathbf{NP} & \text{si el grupo presenta} \\ \mathbf{NI} & \text{en otro caso (grupo no presenta).} \end{cases}$$

5. Integridad académica

Cualquier situación de **falta a la integridad académica** detectada en el contexto del curso (por ejemplo, durante alguna evaluación) tendrá como **sanción un 1**,**1 final en el curso**. Esto sin perjuicio de sanciones posteriores que estén de acuerdo a la Política de Integridad Académica de la Escuela de Ingeniería y de la Universidad, que sean aplicables al caso. Rige para este curso tanto la política de integridad académica del Departamento de Ciencia de la Computación (ver programa del curso) como el **Código de honor de la Escuela de Ingeniería**.

Dudas

Cualquier duda que se presente acerca del enunciado debes consultarla en las ***issues*** del repositorio del curso. Recuerden que como equipo docente estaremos atentos para poder ayudarlos :)