



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

IIC2513 - Sección 1 — Tecnologías y Aplicaciones Web

Tarea 1

Entrega

- **Fecha y hora:** Viernes 4 de abril del 2025, a las 22:00
- **Lugar:** Repositorio individual en la organización del curso en GitHub, main branch.
- **[Link de Invitación](#)**

Objetivos

- **Comprender** el funcionamiento de una API a partir de documentación.
- **Crear** un programa que pueda manejar requests y consumir una API
- **Producir** documentación efectiva y clara que permita el entendimiento del proceso realizado

Descripción

En esta tarea, explorarán cómo integrar y utilizar una API (Interfaz de Programación de Aplicaciones), una habilidad muy útil en el desarrollo de software actual. Frecuentemente, esto les permitirá aprovechar recursos preexistentes en lugar de desarrollar soluciones desde cero, optimizando así el uso de su tiempo y recursos. Esta práctica no solo les ahorra esfuerzo, sino que también les introduce a la amplia gama de herramientas y servicios disponibles en la nube, preparándolos para enfrentar desafíos similares en entornos profesionales.

Para este propósito, deberán hacer uso de la [API DummyJson](#), diseñada para simular la información de una red social. Esta API les suministra datos detallados sobre la aplicación, permitiéndoles crear un prototipo funcional de una aplicación como X (ex Twitter). Tendrán la oportunidad de interactuar con una gran variedad de usuarios y publicaciones (*posts*), junto con un pequeño manejo de inicio de sesión para los usuarios. La información obtenida de la API deberá ser presentada a los usuarios a través de una interfaz desarrollada en React, asegurando una experiencia interactiva y atractiva.

Especificaciones

Para esta tarea se espera que puedan recrear una red social utilizando componentes de react, a partir del código base entregado. Para lograr esto, se podrán utilizar la [documentación](#) de la API DummyJson, en donde podrán encontrar los endpoints que deben consumir para esta tarea.

Pages

Se les entregarán 4 archivos en la carpeta de pages, estos serán utilizados para mostrar las vistas de la página.

- **Index.jsx:** es la landing page de su aplicación. Aquí deberán colocar un mensaje de bienvenida a su página y mencionar qué cosas puede hacer un usuario. ¡Hagan volar su creatividad!

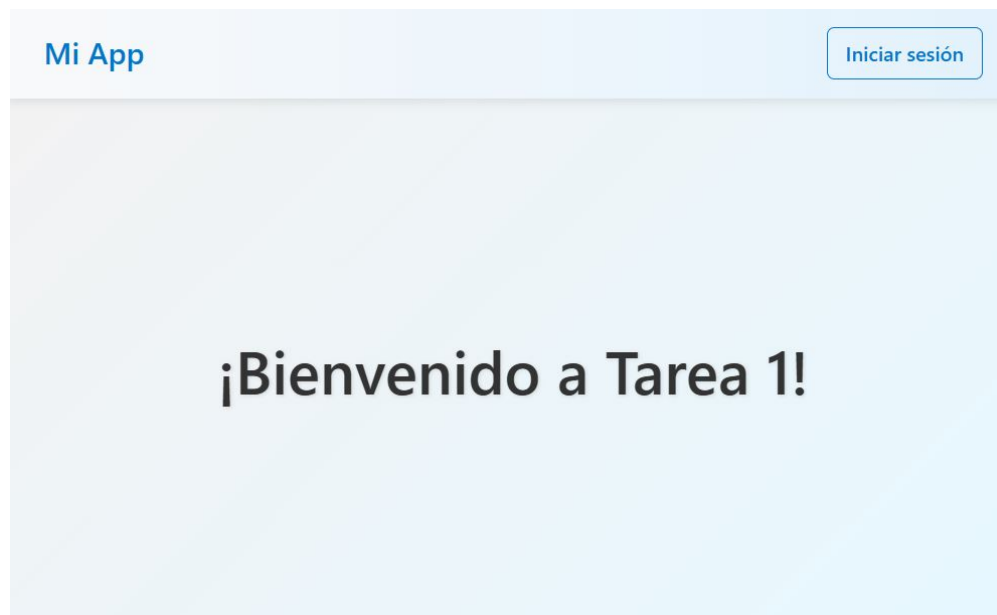


Figura 1: Ejemplo representativo de la página *Index*.

- **Login.jsx:** Esta página sirve para que una persona pueda iniciar sesión en una de las cuentas ya existentes en la aplicación web. Para efectos prácticos, en esta tarea no se considerará el manejo de crear nuevos usuarios.

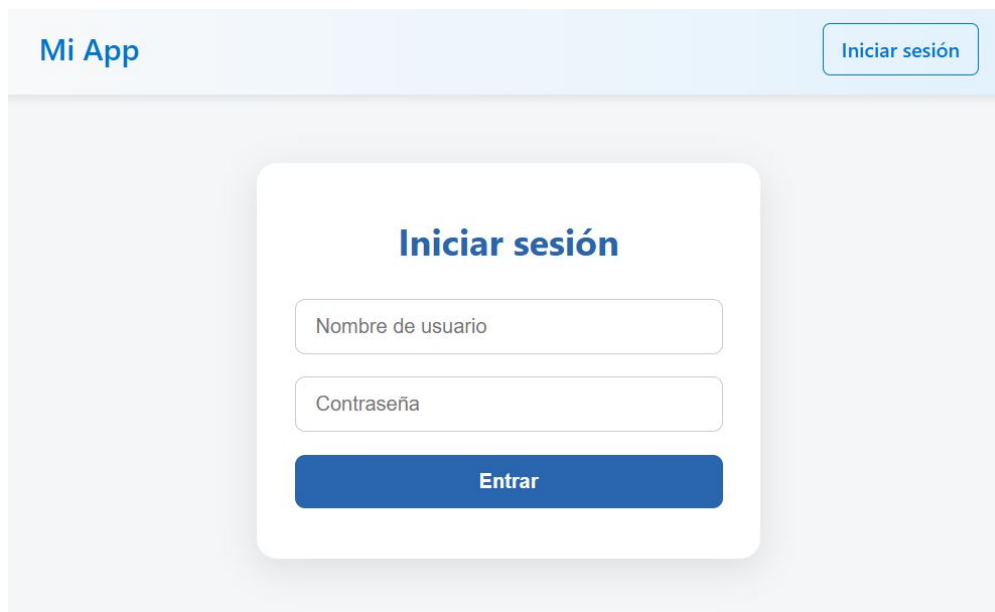


Figura 2: Ejemplo representativo de la página *Login*.

- **Profile.jsx:** Página que permite ver la información del usuario y a su vez, permite acceder a la edición de sus datos.

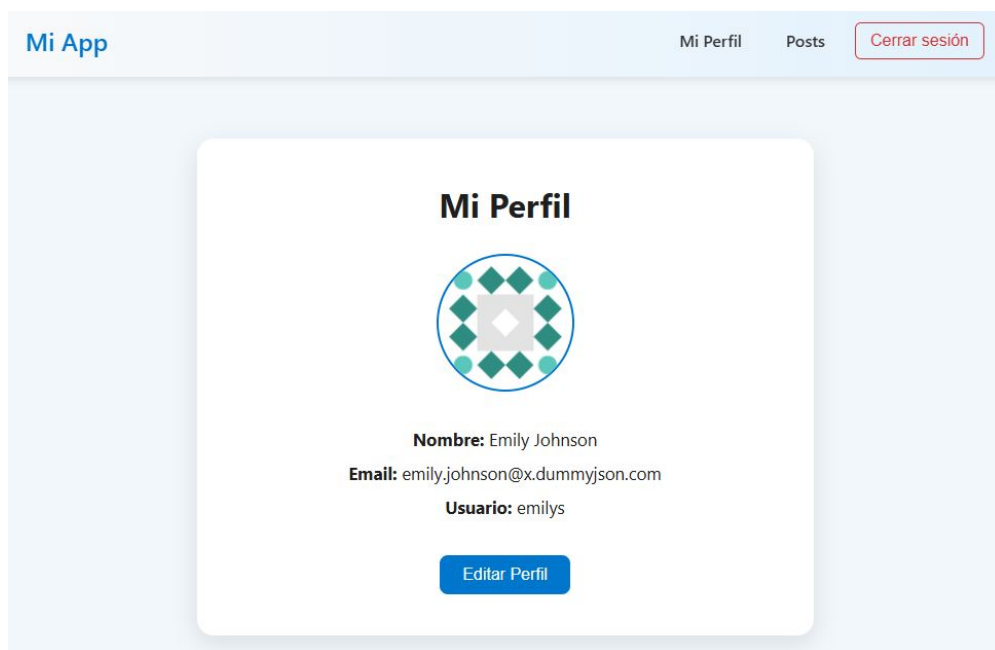


Figura 3: Ejemplo representativo de la página *Profile*.

- **Posts.jsx:** Esta es la página principal de su aplicación. Aquí deberán mostrar todas las tarjetas de los posts de la API utilizando paginación. Estas deben estar desplegadas de forma secuencial. Además, debe tener un botón que permita que los usuarios agreguen un nuevo post.

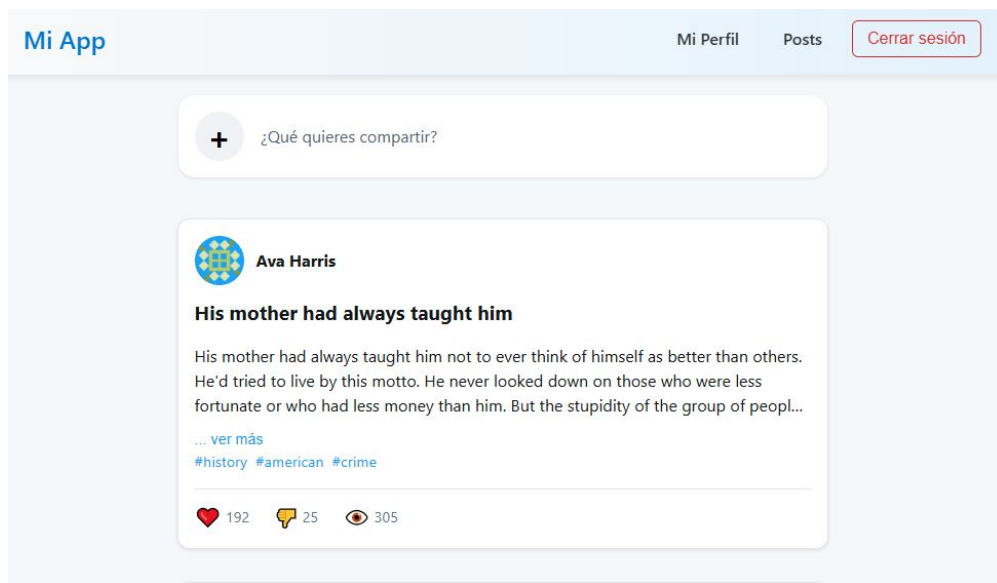


Figura 4: Ejemplo representativo de la página *Posts*.

Components

Su página debe contar mínimo con estos componentes.

- **Navbar:** Componente que permite la navegación entre las distintas vistas de la aplicación. Deberán considerar que esta Navbar se modifica según un usuario haya iniciado sesión o no.
- **CreatePostModal:** Componente utilizada para crear la información sobre un post. Este componente, al ser un formulario, debe incluir validación de datos.

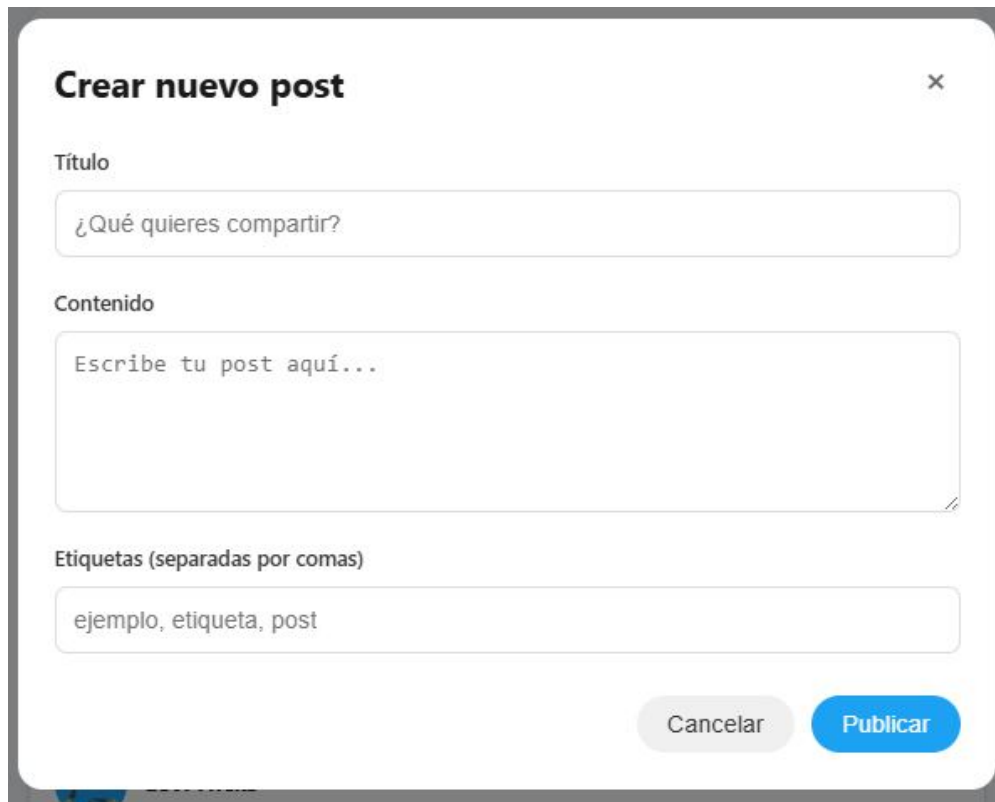
The image shows a modal window titled "Crear nuevo post" with a close button (X) in the top right corner. The form contains three input fields: "Título" with the placeholder text "¿Qué quieres compartir?", "Contenido" with the placeholder text "Escribe tu post aquí...", and "Etiquetas (separadas por comas)" with the placeholder text "ejemplo, etiqueta, post". At the bottom right of the modal, there are two buttons: "Cancelar" (light gray) and "Publicar" (blue).

Figura 5: Ejemplo representativo del componente *CreatePostModal*.

- **EditPostModal:** Componente utilizada para editar la información sobre un post. Esta al ser un formulario, debe incluir validación de datos.
- **InfoCard:** Este componente se encarga de mostrar la información detallada de un post, hasta un límite de 3 líneas de contenido.

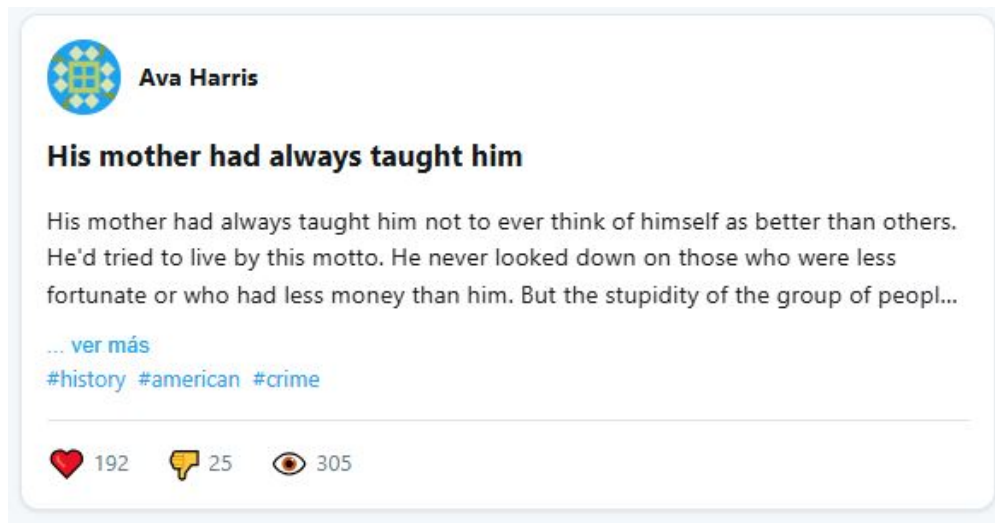


Figura 6: Ejemplo representativo del componente *InfoCard*.

- **PostInfoModal:** componente diseñado para mostrar toda la información de un post, además de contener los botones para editar y eliminar dicho post. Además, notar que es la vista extendida de InfoCard. Junto con ello, destacar que el bonus de la tarea considera mostrar los comentarios de un post en este componente.



Figura 7: Ejemplo representativo del componente *PostInfoModal*.

- **ConfirmDeleteModal:** componente que se encarga de mostrar un mensaje de confirmación para eliminar el post.

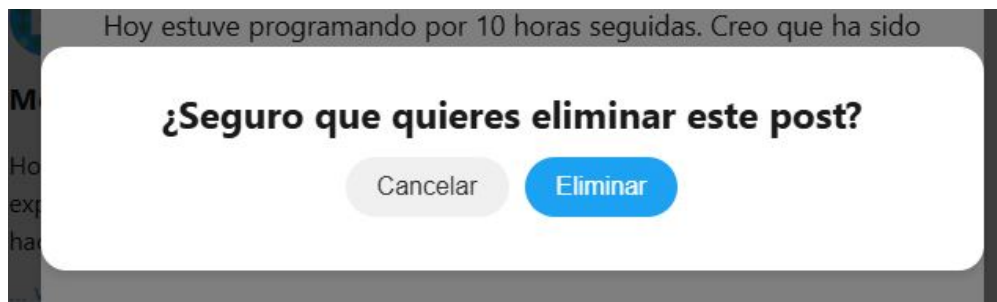


Figura 8: Ejemplo representativo del componente *ConfirmDeleteModal*.

- **EditProfileModal:** componente utilizado para editar los datos básicos de un usuario.

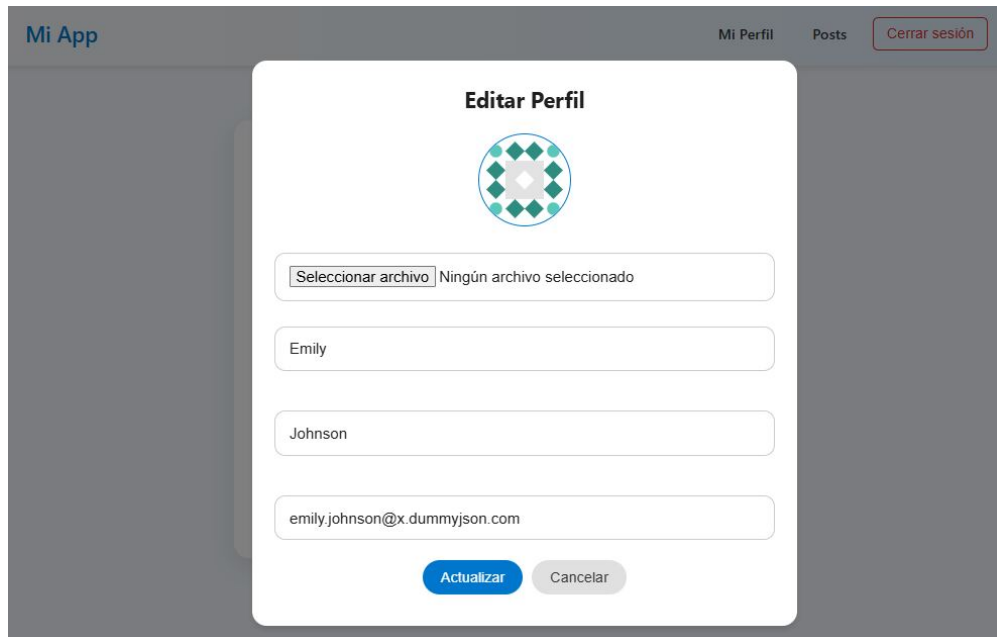


Figura 9: Ejemplo representativo del componente *EditProfileModal*.

Consideraciones Importantes

A continuación, se presentan algunas directrices **importantes** para el desarrollo de la tarea:

- **Restricciones de Librerías:** Está **estrictamente prohibido** el uso de cualquier librería externa o componentes precontruidos (bootstrap, Tailwind CSS, Bulma o parecidos). El incumplimiento de esta regla resultará en una calificación automática de 1.0, sin posibilidad de re-corrección.
- **Pulls en GitHub:** No deben subir el directorio `node_modules`. Además, solo se revisará en la **rama main**. El incumplimiento de esta regla resultará en la reducción en un 50% de su nota final, sin posibilidad de re-corrección.
- **Herramienta de petición HTTP:** Solamente se permitirá el uso de `axios` como librería externa para la realización de solicitudes HTTP a la API. Tampoco se permitirá el uso de `fetch`.
- **Diseño y Código Base:** El código base proporcionado tiene un carácter orientativo y no es obligatorio seguirlo en su totalidad. Sin embargo, todos los `components` y `pages` deben estar correctamente implementados y funcionales. Se permite plena libertad en cuanto al diseño visual, siempre que se cumplan los requisitos funcionales de la tarea.

Interfaz gráfica

Para esta parte, tendrán que mostrar visualmente los resultados de las *requests* anteriormente descritas. Para ello, les proveeremos un código base en React que podrán encontrar en sus respectivos repositorios.

Documentación del proceso

Este ítem tiene como propósito:

- Explicar qué logra hacer su programa y también aquello que no lograron implementar.
- Indicar apropiadamente el proceso para levantar tu aplicación, incluyendo que comandos se deben correr.
- Dar a entender cómo funciona cada parte de su programa, es decir, cómo funciona cada *endpoint* implementado y qué consideraciones tuvieron que tener para que estos funcionaran correctamente.
- Indicar consideraciones particulares de su trabajo (en caso de haber).

Además, en sus respectivos repositorios encontrarán una serie de preguntas teóricas acerca de su tarea y del consumo de APIs en general. Deberán responder estas preguntas de forma más completa posible.

Para un mejor entendimiento sobre cómo escribir la documentación, se les hará entrega de una plantilla con la estructura que debe seguir, su descripción y las preguntas que deben responder.

Entregables

Cada entrega deberá incluir los siguientes archivos:

- `README.md` con la documentación de tu programa. Debes mencionar brevemente aquello que pudiste implementar y también lo que no. Debes indicar cómo levantar tu aplicación y responder las preguntas indicadas en el *template* que te facilitamos. Para finalizar, debes explicar claramente cómo funciona cada implementación de tu programa.
- Aplicación en React desarrollada en base al material entregado. Se deben considerar todos los directorios necesarios para poder levantar y correr el programa.

Rúbrica

A continuación, se describe el criterio de cada uno de los ítems de la rúbrica con la que se evaluará esta entrega. La nota se calcula al 50 % considerando un total de **34.5** puntos.

- **Navbar [2 puntos]:** Se debe incluir la navbar en todas las vistas. Esta debe contener el logo y nombre de la red social que están creando. Además, debe cambiar según haya o no un usuario logeado, mostrando un botón de Iniciar Sesión o uno de Cerrar Sesión según corresponda.
- **Página Index: [1 pts]:** Página de bienvenida a su red social, en donde muestren en logo en grande y un mensaje creativo de su aplicación.
- **Página Login: [3 pts]:** Formulario que permita ingresar a la aplicación través del nombre de usuario y contraseña. En caso de que el usuario ingresado no exista, debe notificarse con un pop up.
Cuando la API ha permitido el acceso al usuario, se debe guardar el token retornado por esta en el localStorage.
El formulario debe cumplir con estar centrado tanto vertical como horizontalmente.
- **Página Profile: [1.5 puntos]:** Vista debe mostrar la imagen de perfil del usuario, junto con su nombre, apellido, email y nombre de usuario. Además, debe poseer un botón para editar.
- **Página Posts: [2.5 puntos]:** Página principal de la aplicación, que debe mostrar todos los posts. Esta debe cumplir con:
 - Posts alineados horizontalmente
 - Usar componente InfoCard para mostrar la información del post.
 - Utilizar paginación, mostrando página actual y número total de páginas.
 - Botón para crear más posts
- **CreatePostModal [3.5 puntos]:** Este componente es el formulario para crear un post, y debe cumplir con:
 - Tener campos para título, contenido y etiquetas.
 - Validación de atributos no vacíos (a excepción de etiquetas).
 - Pop up en caso de éxito o error.
 - Utilizar el endpoint para crear un post.
- **InfoCard [2.5 puntos]** Este componente es aquel que se muestra en la página principal y debe realizar lo siguiente:
 - Mostrar atributos del post y relacionados (User name, user image, title, body, tags, likes, dislikes, views)
 - El contenido (body) debe mostrarse colapsado en caso de que supere más de tres líneas de cuerpo. La información completa debe poder verse en el componente PostInfoModal.
 - Mostrar etiquetas separadas por un espacio ' ' y con un símbolo '#' adelante. (Ej: #Web #Groot)
- **PostInfoModal [4 puntos]:** Este componente muestra toda la información del post de forma expandida. Además, debe cumplir con:
 - Contener todos los atributos del post mencionados anteriormente.
 - Mostrar todo el contenido del post.
 - Poseer botones de editar y eliminar post, que solamente se deben visualizar si es que el post es del usuario que ha iniciado la sesión. Es decir, solo el usuario que creó un post puede editarlo o eliminarlo.

- **ConfirmDelete [1.25 puntos]:** Este componente se debe mostrar tras presionar el botón de eliminar. Una vez apretado, se debe mostrar un mensaje que solicite confirmar la decisión, para posteriormente ejecutar la acción.
- **EditPostModal: [2.75 puntos]:** Se espera que este componente sea un modal que incluya un formulario para poder editar un post. Este debe
 - Validar que no se incluyan campos vacíos (excepto las etiquetas).
 - Tener cargados los datos a editar en el formulario.
 - Utilizar endpoint para editar un post.
- **EditProfileModal: [3.5 puntos]:** Este componente debe diseñarse para poder actualizar los datos del usuario, y debe:
 - Mostrar imagen del usuario
 - Permitir el cambio de imagen de un usuario
 - Permitir el cambio los atributos nombre, apellido y email.
 - Utilizar el endpoint para editar un usuario
 - Mostrar mensaje de error o éxito tras realizar la solicitud.
- **Escritura de documentación [6 puntos]:** Se espera que se explique de manera clara, ordenada y suficiente cada parte del programa. Se debe notar un claro entendimiento acerca de cómo se consume una API y de cómo se realizó (y cómo se corre) el código entregado. Además, se esperan respuestas claras y completas explicando las preguntas teóricas asociadas a la tarea.

¡Consideración importante! Como buena práctica de diseño, todos los modals deben tener un botón para salir de este mismo.

Bonus

Como en toda buena red social, es fundamental que los usuarios puedan interactuar entre sí a través de comentarios. Para simular esta funcionalidad y optar a un bonus de 0.3 décimas en la nota final de la tarea, deberás implementar lo siguiente:

Comentarios [3 décimas]:

- Deberás crear un componente para los comentarios, en el cual se muestren los atributos pertinentes a estos.
- En el componente PostInfoModal debes mostrar los comentarios del post.
- Debes poder agregar y eliminar un comentario de un post de otra persona. Ojo que solo debes poder eliminar los que te pertenecen a tí como usuario.
- Deberás crear una vista llamada 'Mis comentarios' en la que se muestren todos los post en los cuales has realizado comentarios. Esta pestaña debe estar en la navbar.

Para obtener las 3 décimas, debes cumplir con todos los aspectos mencionados del bonus. Cabe destacar que para realizar este bonus, los endpoints se encuentran en el modelo 'Comments' de la API DummyJson.

Dudas

Cualquier duda que se presente acerca del enunciado debes consultarla en las [issues](#) del repositorio del curso. Recuerda que como equipo docente estaremos atentos para poder ayudarte :)

Notas

No se responderán dudas por correo.

La política de atraso podrán encontrarla en el programa del curso