UnaHur Anti-Social Net – Interfaz de Usuario

Objetivo

Desarrollar el FrontEnd en React para la red social "**UnaHur Anti-Social Net**", utilizando la API proporcionada por el TP de BackEnd.

La aplicación debe permitir que las personas usuarias puedan navegar publicaciones, agregar comentarios, registrarse, iniciar sesión y crear sus propios posteos.

Este trabajo incluye una simulación de login. No se requiere autenticación real ni JWT.

Funcionalidades requeridas

Inicio de Sesión (Login simulado)

- El sistema permite iniciar sesión con un `nickName` y una contraseña fija `"123456"`.
- Al iniciar sesión:
    - Se realiza un `GET /users` a la API para verificar si el usuario existe.
    - La contraseña se valida **localmente**.
    - Si es válido, se guarda el usuario en un contexto global (`useContext`) y se mantiene en `localStorage`.

Las rutas protegidas solo serán accesibles si hay un usuario logueado.

Registro de Usuario

- Formulario para crear un nuevo usuario.
- Validar previamente que el `nickName` no exista.
- Si es válido, enviar `POST /users`.
- Tras el registro, se puede redirigir al login o loguear directamente al usuario.

Home (Página de Inicio)

- Debe incluir un **feed de publicaciones recientes**, que muestre:
    - Descripción
    - Imágenes (si las hay)
    - Etiquetas
    - Cantidad de comentarios visibles
    - Botón “Ver más” → lleva a `/post/:id`
- Además del feed, el contenido de la página es **libre**. Se pueden incluir:
    - Banner de bienvenida

    - Sección “Sobre nosotros”
    - Slogans, frases destacadas, datos curiosos

Detalle de Publicación

- Vista accesible desde /post/:id
- Muestra:
    - Descripción completa
    - Imágenes
    - Etiquetas
    - Lista de comentarios visibles (filtrados por la API)
- Formulario para agregar un comentario nuevo:
    - Campo obligatorio
    - Envío mediante POST /comments
    - Componente controlado (useState)

Perfil de Usuario

- Vista protegida. Solo visible si el usuario está logueado.
- Muestra:
- El nickName del usuario actual
- Lista de publicaciones realizadas por ese usuario (consultadas a la API con su userId)
- Por cada post, debe mostrarse:
- Descripción
- Cantidad de comentarios visibles
- Botón “Ver más” → que lleva a la vista de detalle
- También debe haber un botón para cerrar sesión (logout)

Crear Nueva Publicación

- Vista protegida. Solo accesible si el usuario ha iniciado sesión correctamente.
- Formulario con los siguientes campos:
    - Descripción (obligatoria)
    - URLs de imágenes (opcional): uno o más campos para ingresar URLs de imágenes asociadas
    - Selección de etiquetas: lista obtenida desde la API
- Funcionamiento:
    - Al enviar el formulario:
        - Se hace un POST /posts con description, userId y tags
    - Si se ingresaron URLs de imágenes:
        - Por cada una, se hace un POST /postimages con url y postId
    - Al finalizar, redirigir al perfil o mostrar confirmación

Requisitos Técnicos

Tema	Aplicación
useState, useEffect	Manejo de estado y carga de datos
useContext	Gestión de usuario logueado
react-router-dom	Navegación entre vistas y rutas protegidas
Formularios controlados	Login, registro, comentarios, publicaciones
Fetch o axios	Consumo de API REST
CSS o framework	Diseño responsive libre (Bootstrap, Tailwind, etc.)
localStorage	Persistencia de sesión
Validaciones	Formularios con campos requeridos y feedback visual

Extras opcionales (Bonus)

- Filtro por etiquetas en la Home
- Publicaciones destacadas o aleatorias
- Scroll infinito o paginación
- Animaciones suaves y transiciones
- Alertas visuales (éxito o error)

Entrega esperada

- Repositorio en GitHub (público o compartido)
- README.md con:
    - Descripción del proyecto
    - Instrucciones para correr en local
    - URL de la API utilizada

Para quienes no cursan Estrategia de Persistencia

Si estás cursando solo la materia Construcción de Interfaces de Usuario, vas a trabajar con una copia funcional del BackEnd ya preparada.
Esta app de Node.js se usa como una **caja negra**: no es necesario modificarla ni entender cómo funciona internamente.
Solo deberás realizar los fetch necesarios desde React.
Repo de la API:
https://github.com/lucasfigarola/backend-api

Endpoints disponibles

Método	Endpoint	Uso
GET	/users	Lista completa de usuarios
GET	/users/:id	Obtener usuario por ID
POST	/users	Crear nuevo usuario
GET	/posts	Lista de publicaciones
GET	/posts/:id	Detalle de una publicación
GET	/posts?userId=xxx	Posts de un usuario específico
POST	/posts	Crear publicación
GET	/tags	Listado de etiquetas
GET	/posts/:id/comments	Comentarios visibles del post
POST	/comments	Crear comentario nuevo
GET	/postimages/post/:postId	Traer imágenes asociadas a un post
POST	/postimages	Asociar una imagen a un post

Requerimientos para el BackEnd (para los que están cursando Estrategia de Persistencia)

CORS

Para que el FrontEnd pueda comunicarse con la API local, el BackEnd debe habilitar CORS:

```
npm install cors
```

Y en index.js:

```
const cors = require('cors')
app.use(cors({ origin: 'http://localhost:5173' }))
```