### **ACTIVIDAD 1**

## APLICACIÓN CRUD: GESTIÓN DE USUARIOS

#### PREPARACIÓN PREVIA

Tras crear el proyecto, será necesario instalar las siguientes dependencias, ya que trabajaremos con mui, y con rutas.

- npm install @mui/material @emotion/react @emotion/styled
- npm install @mui/icons-material
- npm install @fontsource/roboto
- npm install react-router-dom

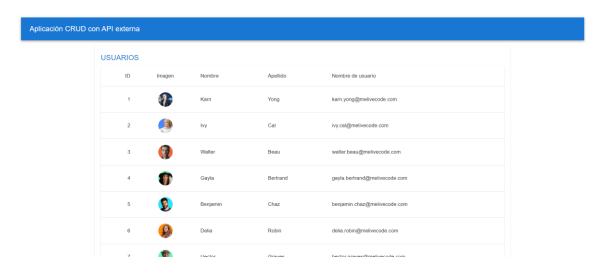
#### PRIMERA PARTE: APLICACIÓN QUE SÓLO MUESTRA DATOS DEFINIDOS EN EL CLIENTE

En esta primera versión crearemos una aplicación que constará de una barra de navegación (de formato libre, puedes echarle un vistazo a <u>AppBar de mui</u>) y un componente llamado **UserList**, que usará una tabla para mostrar información de unos clientes que tendremos almacenados en un array de objetos literales de este tipo (se proporcional archivo con los datos):

```
{
    "id": 1,
    "fname": "Karn",
    "lname": "Yong",
    "username": "karn.yong@melivecode.com",
    "avatar": "https://www.melivecode.com/users/1.png"
}
```

Este array servirá para inicializar una variable de estado "users", necesaria para realizar el resto de los ejercicios.

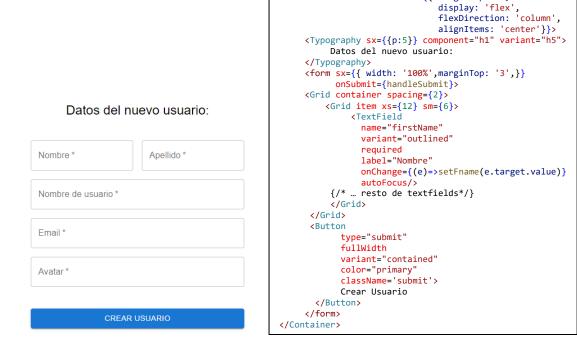
Para realizar la tabla puedes usar los componentes Table, TableBody, TableCell, TableContainer, TableHead, TableRow de <u>mui</u>.



#### SEGUNDA PARTE: CREAR DATOS DEFINIDOS EN EL CLIENTE

Para añadir un usuario, añadiremos un botón en la interfaz principal, sobre la tabla de usuarios y crearemos un componente específico **UserCreate**, que contendrá el formulario para añadir un usuario. Será necesario pasarle (con lifting o con contextos) la función de añadir usuario que básicamente modificará la variable de estado users, añadiendo el nuevo elemento. Una vez añadido volverá a la pantalla inicial con el listado ya actualizado.

<Container maxWidth="xs" sx={{ marginTop:'8',</pre>



Para trabajar con varias pantallas será necesario incorporar rutas en este punto.

Nuestra App podría ser algo así:

# TERCERA PARTE: LA APLICACIÓN HACE LAS MISMAS OPERACIONES, PERO USANDO UNA API (LOS DATOS PASAN A ESTAR EN EL SERVIDOR)

Vamos a actualizar nuestros componentes para que los datos residan en un servidor. La API que usaremos es https://melivecode.com/. Ten en cuenta que los 12 primeros registros no pueden ser borrados ni actualizados. Tus pruebas se realizarán a partir del registro número 13.

A continuación, nos falta por implementar los métodos de actualización y borrado. Será necesario añadir una nueva columna en nuestra tabla de usuarios para estas acciones:



- En el caso de la edición será necesario pasarle en la ruta el id, y acceder al mismo mediante el hook useParams. Este id será utilizado por el formulario para rellenar los datos del usuario a modificar. Recuerda usar método PUT en el fetch.
- Para el **borrado** la URL es <a href="https://www.melivecode.com/users/delete">https://www.melivecode.com/users/delete</a>, y es necesario pasarle en el fetch el id del usuario que se quiere borrar. Una vez realizado el borrado y si todo ha ido bien llamaremos el método que recarga los usuarios (ya implementado al hacer el listado).