Guía para entender el código

`App.jsx` - Núcleo de la aplicación

Este componente define la estructura base de la app y el sistema de navegación usando React Router.

Estructura general:

- **Router**: Habilita la navegación entre rutas sin recargar la página.
- **Navbar**: Barra superior fija presente en todas las vistas.
- **Container**: Contenedor con margen superior para separar el contenido del borde.
- **Routes**: Sección donde se definen las rutas disponibles y los componentes que se renderizan según la URL.

Explicación por bloques:

Líneas	Descripción
1	Importa herramientas de rutas desde `react-router-dom`: `Router`, `Routes` y `Route`.
2	Importa `Container` desde MUI para dar estructura y espaciado al contenido.
3	Importa el componente `Navbar`, que se mostrará en todas las páginas.
5–8	Importa las páginas: `Home`, `About`, `Skills`, y `Certificates`.
10	Comienza la definición del componente principal `App`.
12	Envuelve toda la app con el `Router`, habilitando navegación entre páginas.
14	Renderiza `Navbar`, visible en todas las rutas.
17	`Container` de MUI con margen superior (`mt: 4`), para separar el contenido del borde superior.
19–24	Se definen las rutas con su componente correspondiente:

- `/` → `Home`

- `/about` → `About`

- `/skills` → `Skills`

- `/certificates` → `Certificates`

26 Cierre de `Router`.

28 Cierre del componente `App`.

30 Exportación del componente para poder usarlo en otros archivos.

Notas adicionales:

- `Router` debe envolver **todo** lo que tenga que ver con navegación, y lo estás haciendo perfecto.
- `Navbar` está **fuera** de las `Routes` porque debe mostrarse en todas las rutas (¡bien pensado!).
- Estás usando correctamente el componente `Container` de MUI para no dejar el contenido pegado arriba.
- Las rutas están claras y bien definidas. Muy bien estructurado.

`main.jsx` – Punto de entrada de la app

Este archivo **monta la aplicación en el DOM**, aplica el **tema personalizado** y el **reseteo de estilos globales** de MUI.

Explicación por líneas:

Líneas	Descripción
1	Importa React (necesario para trabajar con JSX y el modo estricto).
2	Importa `ReactDOM`, que se encarga de renderizar la app en el navegador.
3	Importa el componente principal `App`, que contiene toda la estructura.
4	Importa `CssBaseline`, un componente de MUI que aplica un reseteo de estilos CSS por defecto.
5	Importa el `CustomThemeProvider`, tu proveedor de tema personalizado para MUI.
7	Se llama a `ReactDOM.createRoot()` para iniciar la app en el elemento con id `root`.
8	Se activa `React.StrictMode`, una herramienta de desarrollo para detectar errores potenciales.
9	Se aplica el proveedor de tema (`CustomThemeProvider`) a toda la app.
10	`CssBaseline` resetea los estilos predeterminados del navegador.

11	Se renderiza el componente `App`, el núcleo de tu aplicación.
12–13	Cierre de los componentes y final del renderizado.

Notas adicionales:

Usar `React.StrictMode` no es obligatorio en producción, pero **te avisa de posibles malas prácticas** o funciones que se usarán de forma diferente en futuras versiones. ¡Buen hábito!

`ThemeContext.jsx` – Tema personalizado y modo claro/oscuro

Este archivo define el **contexto global** para controlar si la app está en **modo claro** o **modo oscuro**, usando Material UI. También provee ese tema a toda la aplicación.

Explicación por líneas:

Líneas	Descripción
1	Importa `React` y hooks (`createContext`, `useState`, `useMemo`) para manejar estado y optimizar renders.
2	Importa `createTheme` y `ThemeProvider` desde MUI para construir y aplicar el tema visual.
4	Crea un contexto llamado `ColorModeContext` con una función vacía por defecto. Esto se usará para cambiar el modo desde cualquier parte de la app.

6	Define el componente `ThemeContextProvider`, que envolverá la app y le proporcionará el tema.
8	Declara el estado `mode`, que determina si el tema es `'light'` o `'dark'`. Por defecto comienza en `'light'`.
11–15	Define `colorMode` con `useMemo` para evitar que la función `toggleColorMode` se regenere en cada render. Esta función cambia el modo al opuesto.
18–22	Crea el objeto `theme` con `createTheme` de MUI, y le pasa el `mode` actual a la paleta. Este `theme` se actualiza automáticamente al cambiar el modo.
25–29	Retorna el proveedor de contexto `ColorModeContext` con la función `toggleColorMode`, y el `MuiThemeProvider` que aplica el tema a todos los componentes hijos (`{children}`).

Notas adicionales:

- Usar `useMemo` aquí es **clave para el rendimiento**, especialmente si tu app crece. Sin él, cada render volvería a crear el tema y la función, lo que puede causar renders innecesarios.
- Esta estructura está muy bien pensada para escalar. Podrías incluso añadir más personalizaciones al tema (`typography`, `breakpoints`, etc.) más adelante.

`Navbar.jsx` – Barra de navegación + botón de tema

Este componente representa la **barra de navegación superior** que aparece en todas las páginas. Contiene enlaces dinámicos y un botón para alternar entre **modo claro y oscuro**.

Explicación por líneas:

Líneas	Descripción
1	Importa `useContext` para acceder al contexto del tema.
2	Importa componentes de UI desde MUI: `AppBar`, `Toolbar`, `Button`, `Box`, `IconButton`.
3	Importa `NavLink` desde React Router para navegación con estilos activos.
4–5	Importa iconos de sol y luna para representar los modos oscuro y claro.
6	Importa el contexto `ColorModeContext` desde tu `ThemeContext`.
7	Importa el hook `useTheme` para saber si el tema actual es claro u oscuro.
9–13	Define un array con los ítems del menú: etiquetas y rutas de cada sección.

Componente `Navbar`

Líneas	Descripción
15	Comienza la función `Navbar`.
17	`theme` obtiene el tema actual de MUI (`light` u `dark`).

19	`colorMode` accede a la función `toggleColorMode` desde el contexto.

Renderizado:

Líneas	Descripción
21	Se renderiza un `AppBar` (barra de navegación fija en la parte superior).
22	Dentro del `AppBar`, se usa un `Toolbar` para alinear el contenido.
24	Se crea un `Box` con `flexGrow: 1` para empujar el botón de modo al lado derecho.
25–38	Se mapean los ítems de navegación (`navItems`) para crear un `Button` por cada uno: - `component={NavLink}` → navegación SPA con estilos activos. - `sx` → estilos inline con MUI (color blanco, margen, y subrayado en el activo). - `end={path === '/'}` → para que solo se active exacto en la home.
41	`IconButton` que llama a `toggleColorMode` al hacer clic.
43	Muestra el icono correspondiente según el modo actual (`dark` → sol, `light` → luna).

46–47	Cierre de `Toolbar` y `AppBar`.
48	Fin del componente `Navbar`.

Notas adicionales:

- El uso de `NavLink` te permite aplicar **estilos activos** automáticamente sin complicarte.
- `sx={{ '&.active': ... }}` está genial para marcar qué sección está activa. Súper pro.
- Y el botón de tema es **intuitivo y funcional**, ¡muy buen detalle para mejorar la UX!

`Home.jsx` – Página de bienvenida + datos personales con animaciones

Este componente es la **pantalla principal** de la app. Presenta a la "heroína" del portfolio con un avatar, descripción, y un botón que muestra u oculta sus datos personales, todo con animaciones gracias a **Framer Motion**.

Explicación por líneas:

Líneas	Descripción
1	Importa `React` y el hook `useState` para manejar el estado de visibilidad.
2	Importa componentes visuales desde MUI (`Box`, `Avatar`, `Typography`, etc.).

3–6	Importa iconos para los datos personales (email, teléfono, ubicación, etc.).
7	Importa `motion` y `AnimatePresence` de Framer Motion para las animaciones.
8	Importa la imagen del avatar desde los assets.

Lógica del componente `Home`

Líneas	Descripción
10	Se define el componente `Home`.
12	`showData`: estado booleano que controla si se muestran los datos personales.
15–20	Array con los datos personales, cada uno con su icono y texto correspondiente.

Render del componente:

Contenedor principal animado

Líneas	Descripción
22–28	`Box` animado con Framer Motion (desvanecido y caída suave al cargar).

Avatar

Líneas	Descripción
30–47	`Avatar` circular con estilos personalizados: borde, sombra, hover con zoom, etc.

Textos

Líneas	Descripción
50–52	`Typography` con saludo y título llamativo.
54–63	Subtítulo con descripción breve y estilo refinado (cursiva, centrado, etc.).

Botón para alternar visibilidad

Líneas	Descripción
66–69	Botón que cambia el estado `showData` y actualiza su texto dinámicamente.

Sección animada de datos personales

Líneas	Descripción
72	`AnimatePresence` permite animar la entrada/salida del contenido condicional.
73–92	`Paper` animado que aparece/desaparece con efecto suave. Contiene los datos personales.

Título de sección con efecto hover

Líneas	Descripción
95–112	`Typography` con decoración inferior que se expande al pasar el mouse. Muy visual.

Lista de datos

Líneas	Descripción
115–133	Se mapea `dataItems` para mostrar cada dato con su icono e info. Tiene estilo hover y separación visual.

| **135–137** | Cierre de `AnimatePresence`, `Box` y del componente `Home`. |

Notas adicionales:

- `AnimatePresence` + `motion.div` = para cualquier animación de entrada/salida. Te queda top visualmente.
- Estás usando bien `sx` para mantener el estilo limpio y concentrado.
- `Avatar` con hover escalado + sombra = detalle de diseño que suma un montón.

`About.jsx` – Experiencia profesional

Este componente muestra una lista de experiencias relevantes como desarrolladora web, con una breve intro y detalles presentados en una tarjeta con íconos e info.

Explicación por líneas:

Líneas	Descripción
1	Importa componentes desde MUI: contenedor, tipografía, tarjeta (`Paper`), lista y sus ítems.
2–5	Importa iconos representativos para cada experiencia (`Code`, `React`, `JS`, `Storage`).

Lógica del componente `About`

Líneas	Descripción
7	Comienza la función `About`.
9–25	Array `experiences` con la información de cada experiencia: - `icon`: icono con color. - `title`: título de la experiencia. - `desc`: descripción más detallada.

Render del componente:

Líneas	Descripción
27	`Box` como contenedor principal con padding (`p: 3`).
29	Título principal (`Typography` con `variant="h4"`).
32	Texto introductorio corto.
35	`Paper` que actúa como tarjeta contenedora de la lista. Elevación 3 = sombra suave.
36–44	`List` que recorre el array `experiences` con `map`: - `ListItem` para cada entrada. - `ListItemIcon` con el icono. - `ListItemText` con el título (primary) y descripción (secondary).

| **45–47** | Cierre del componente: `Paper`, `Box`, y `About`. |

Notas adicionales:

- Puedes añadir animaciones suaves (como `Fade` de MUI o `motion.div`) para que los ítems aparezcan con efecto.
- Si tuvieras muchas experiencias, podrías dividirlas en secciones o poner tabs (`Tabs` de MUI).
- Los iconos están bien elegidos, aunque podrías usar alguno más "directo" si encuentras uno específico para React (`SiReact`, por ejemplo, desde `react-icons`).

`Skills.jsx` – Habilidades técnicas

Este componente muestra las **competencias técnicas** de la heroína del portfolio, con iconos representativos e indicadores de nivel usando `LinearProgress`.

Explicación por líneas:

Líneas	Descripción
1	Importa componentes de MUI: `Box`, `Grid`, `Card`, `CardContent`, `Typography`, y `LinearProgress`.
2	Importa iconos desde `react-icons/fa` para representar visualmente cada habilidad.

Lógica de datos

Líneas	Descripción
4–10	`skills`: array de objetos con: - `name`: nombre de la habilidad. - `icon`: icono representativo con color y tamaño.

- `level`: nivel expresado en porcentaje para la barra de progreso.

Render del componente:

Líneas	Descripción
12	Comienza la función `Skills`.
14	Contenedor general con padding (`Box sx={{ p: 3 }}`).
16	Título centrado (`Typography` con `align="center"` y `variant="h4"`).
19	`Grid container` para organizar las tarjetas responsivamente.
21–33	Se recorre el array `skills` y se crea un `Grid item` por cada habilidad: - `xs={12}`: ocupa todo el ancho en móvil. - `sm={6}`: media pantalla en tablet. - `md={4}`: un tercio en escritorio.
23	Cada `Card` usa `display: flex` para alinear el icono con el contenido.
25	El `Box` del icono tiene margen derecho para separación visual.
27–28	Se muestra el nombre de la habilidad y una barra de progreso (`LinearProgress`) con el nivel.

| **34–36** | Cierre del `Grid`, `Box`, y del componente `Skills`. |

Notas adicionales:

- Podrías animar la barra de progreso usando `useEffect` para que "cargue" desde 0 al montar el componente 💡.
- Si añades muchas skills, considera un sistema de categorías o tabs para agruparlas (frontend, backend, herramientas...).
- ¿Quieres hacerlo más visual aún? Cambia el color de la barra en función del nivel (por ejemplo: verde si es +80, naranja si es 50–79, etc.).

`Certificates.jsx` – Certificados obtenidos

Este componente muestra los certificados alcanzados por la heroína del portfolio, usando tarjetas interactivas con animaciones, y un modal para ampliar cada certificado.

Explicación por líneas:

Líneas	Descripción
1	Importa `useState` desde React para manejar estados del componente.
2	Importa componentes de MUI: `Box`, `Typography`, `Card`, `CardMedia`, `CardContent`, `Grid`, `Modal`, `IconButton`.
3	Importa `motion` y `AnimatePresence` desde `framer-motion` para animaciones suaves y condicionales.
4	Importa el icono de cierre (`CloseIcon`) para el modal.

Lógica de datos

Líneas	Descripción
6–9	Importa las imágenes de certificados desde la carpeta `assets/certificates`.
11–20	`certificates`: array de objetos que contienen: - `title`: nombre del certificado - `image`: ruta a la imagen del certificado

	- `details`: información adicional como fecha, lugar, institución
23–24	`collapseVariants`: objeto con dos estados de animación (`open` y `closed`) para controlar el despliegue/ocultación de detalles.

Lógica del componente

Líneas	Descripción
26	Comienza la función `Certificates`.
28	`expandedCard`: guarda el índice de la tarjeta actualmente expandida.
30	`openModal`: controla si el modal está abierto.
32	`modalImg`: almacena la imagen a mostrar en el modal.
35–37	`toggleExpand`: alterna el estado expandido de una tarjeta al hacer clic.
40–42	`handleOpenModal`: abre el modal y carga la imagen correspondiente.
45–47	`handleCloseModal`: cierra el modal y limpia la imagen.

Render del componente

Líneas	Descripción
49	Contenedor principal con margen superior (`mt: 5`).
51–53	Título "Certificados" centrado usando `Typography`.
56	`Grid container` con espaciado (`spacing={3}`) para organizar las tarjetas.
57–84	Mapea el array `certificates` y genera una tarjeta (`Card`) por cada uno: - `motion.div`: anima la tarjeta al hacer hover (escala). - `onClick`: alterna la expansión de la tarjeta.
64–68	`CardMedia`: muestra la imagen del certificado. Al hacer clic, abre el modal sin expandir la tarjeta (`e.stopPropagation()`).
69–71	`CardContent` con el título del certificado.
73–83	Si la tarjeta está expandida, muestra los detalles con animación (`motion.div`) dentro de `AnimatePresence`.

Modal de imagen ampliada

Líneas	Descripción
86–115	`Modal` de MUI que se activa con `openModal`. Centrado en pantalla con `Box`.
94–98	`IconButton` con `CloseIcon` para cerrar el modal. Posicionado arriba a la derecha.
100–109	Imagen ampliada del certificado, estilizada con borde redondeado, sombra, y tamaño responsivo.

Notas adicionales:

- Puedes animar la apertura del modal con un `Fade` o `Zoom` si quieres un toque más dinámico.
- Si tienes muchos certificados, podrías agruparlos por tipo o año con filtros o tabs.
- Considera agregar un botón de descarga o link externo si los certificados están disponibles online.