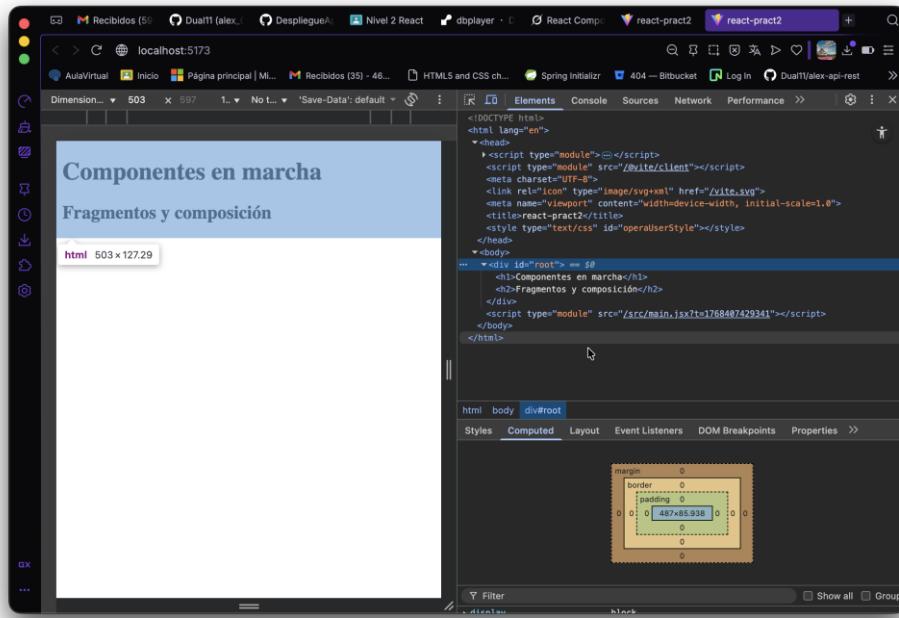


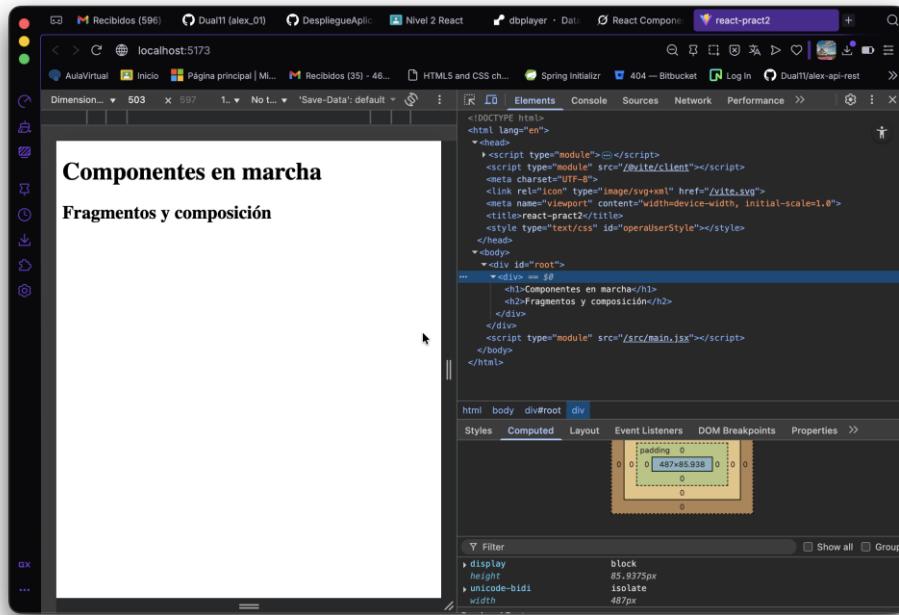
1- Cambiando el texto sin el div

Ha agregado el texto dentro del div root, con un div vacío.



2- He añadido el texto dentro de un div

La diferencia ahora es que, si ha añadido el texto en el div, ya que se lo hemos especificado.

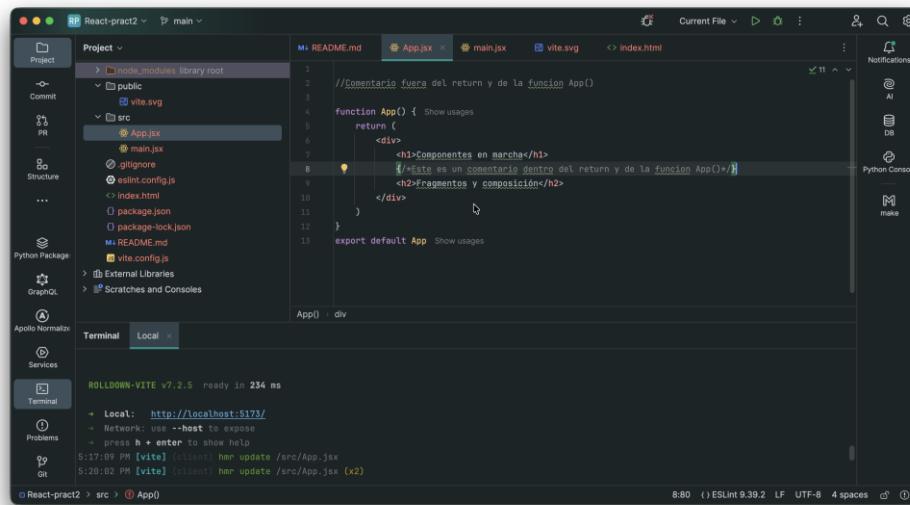


3- Renderizado: ¿dónde se “engancha” React?

`document.getElementById('root')`. Selecciona el elemento que está en el archivo index.html. Este div es el punto de entrada donde React "engancha" toda la aplicación. Luego, `ReactDOM.createRoot()` crea un root de renderizado en ese elemento y con `.render()` inserta el componente dentro de él, reemplazando el contenido existente.

4- Agregando comentarios

He agregado los comentarios y no los muestras



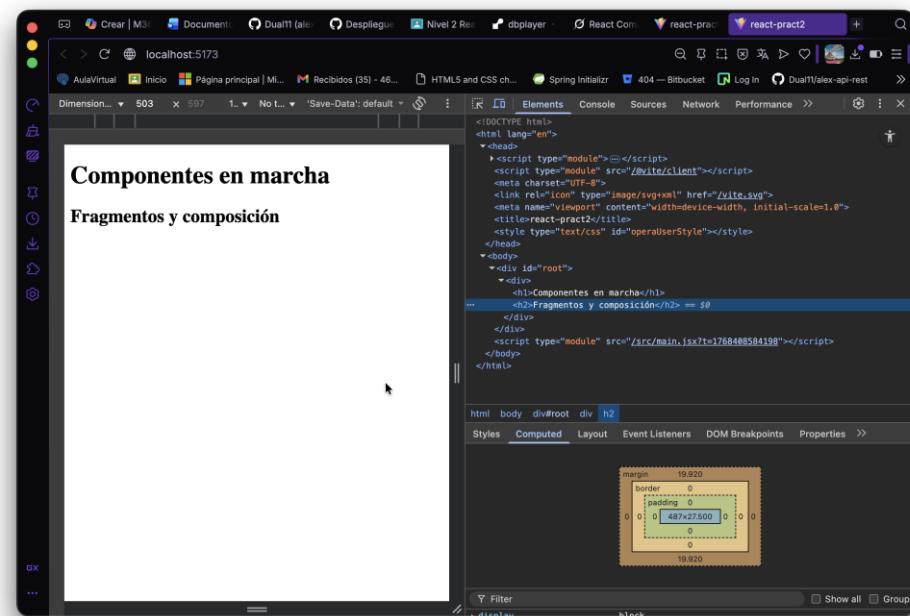
The screenshot shows a code editor with a file named `App.jsx` open. The code contains the following:

```

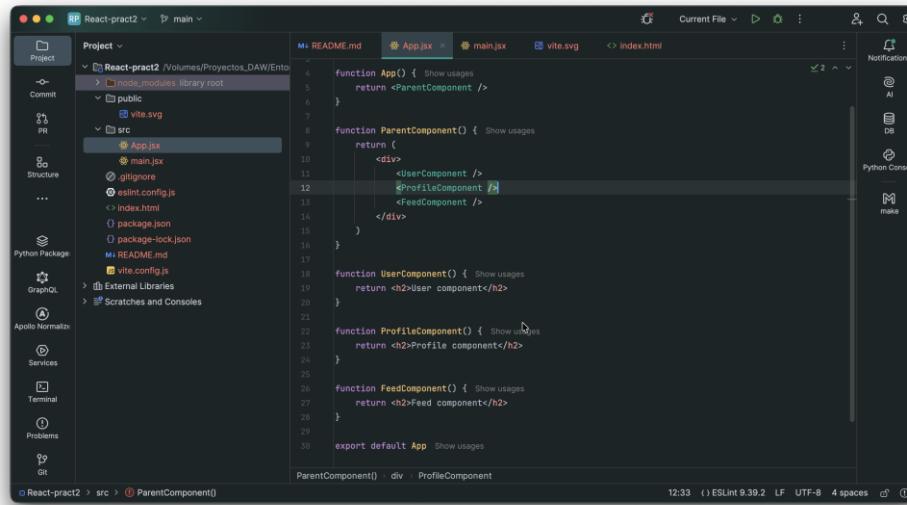
1 //Comentario fuera del return y de la función App()
2
3 function App() {
4   return (
5     <div>
6       <h1>Componentes en marcha</h1>
7       <!--Este es un comentario dentro del return y de la función App()-->
8       <h2>Fragmentos y composición</h2>
9     </div>
10   )
11 }
12
13 export default App

```

The code editor has a terminal tab at the bottom showing the command `ROLLDOWN-VITE v7.2.5 ready in 234 ms` and a local host URL `http://localhost:5173/`.



5- He creado los tres componentes



The screenshot shows a code editor interface with a sidebar containing project navigation and various development tools like AI, DB, and Python Console. The main area displays a file named `App.jsx` with the following content:

```

4  function App() { Show usages
5      return <ParentComponent />
6  }
7
8  function ParentComponent() { Show usages
9      return (
10         <div>
11             <UserComponent />
12             <ProfileComponent />
13             <FeedComponent />
14         </div>
15     )
16 }
17
18 function UserComponent() { Show usages
19     return <h2>User component</h2>
20 }
21
22 function ProfileComponent() { Show usages
23     return <h2>Profile component</h2>
24 }
25
26 function FeedComponent() { Show usages
27     return <h2>Feed component</h2>
28 }
29
30 export default App Show usages

```

The code defines three components: `UserComponent`, `ProfileComponent`, and `FeedComponent`, which are then used within the `ParentComponent`. The `App` component is the root component.

PREGUNTAS EXTRAS

1- ¿Qué es un componente en React?

Un componente en React es una pieza reutilizable e independiente de la interfaz del usuario, esto permite dividir la aplicación en pequeñas partes.

2- ¿Por qué usamos Fragmentos (<>...</>) en lugar de un <div>?

Los Fragmentos permiten devolver varios elementos hermanos sin añadir un nodo extra al DOM. Un añade un contenedor innecesario que puede romper estilos, semántica o layouts, mientras que `<>` es invisible en el HTML final.

3- ¿Qué papel tienen index.html y main.jsx en el renderizado?

`index.html` es la página base que carga el navegador y contiene el dónde React insertará la aplicación.

`main.jsx` es el punto de entrada de JavaScript: selecciona el elemento root con `document.getElementById('root')`, crea un root con `ReactDOM.createRoot()` y renderiza el componente principal () dentro de él.

4- Explica con tus palabras qué significa “componer componentes”.

Componer componentes significa construir la interfaz combinando componentes pequeños dentro de otros más grandes, como si fueran piezas de Lego. Por ejemplo, un componente “Página” puede contener “Header”, “Sidebar” y “Footer”, y cada uno de esos puede tener sus propios subcomponentes. Esto crea un árbol de componentes reutilizable, organizado y fácil de mantener.