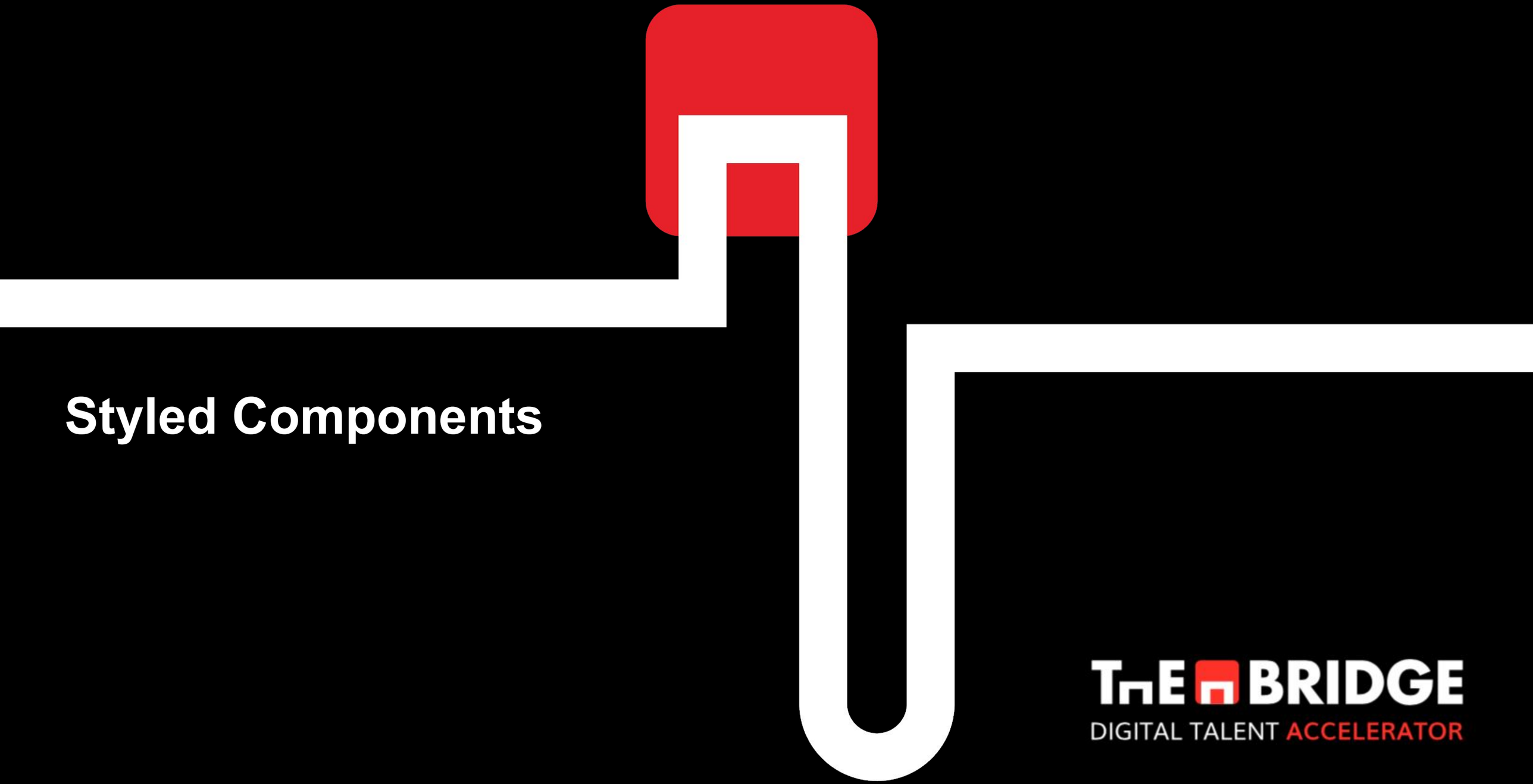
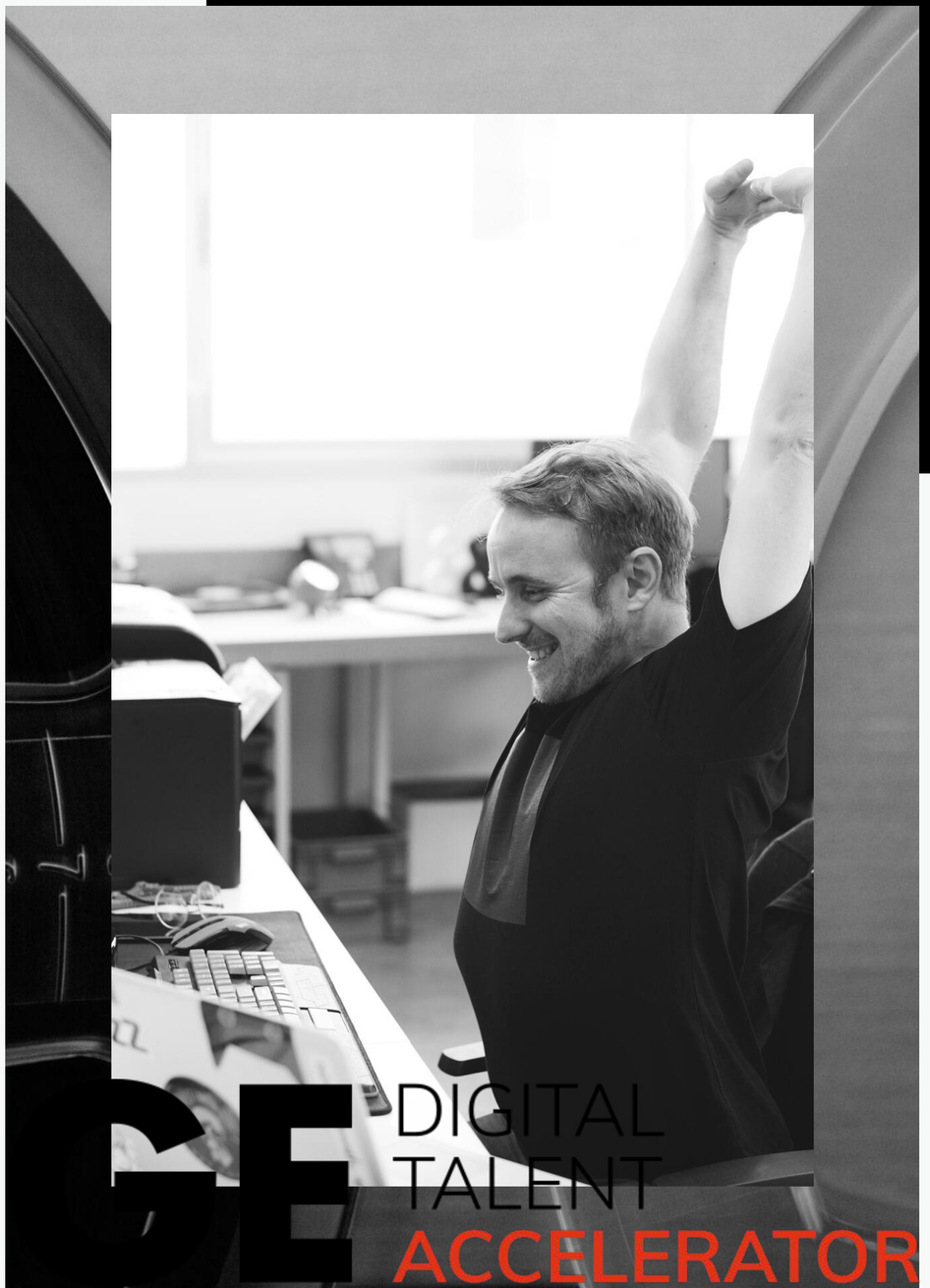
Styled Components



Índice

CSS-in-JS

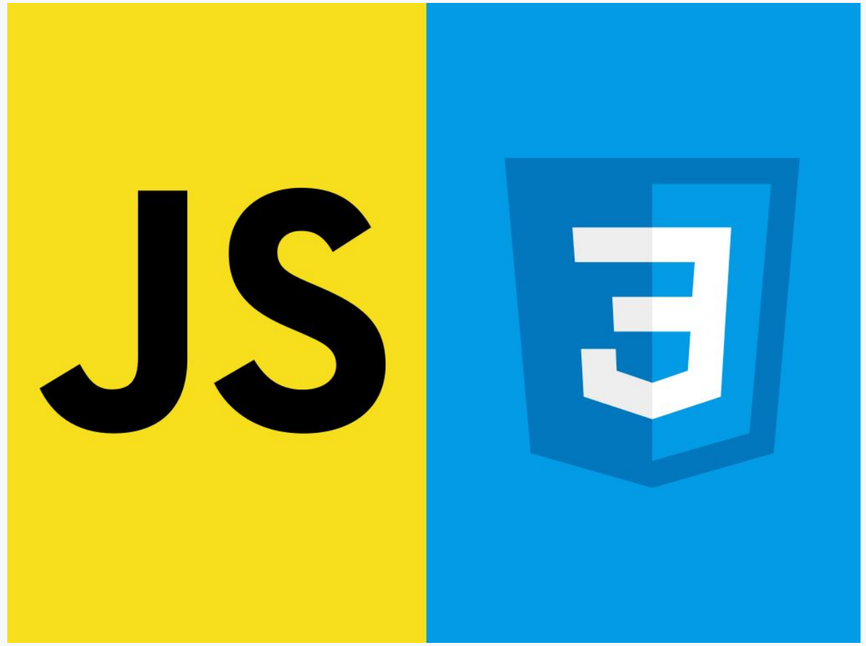
Styled Components



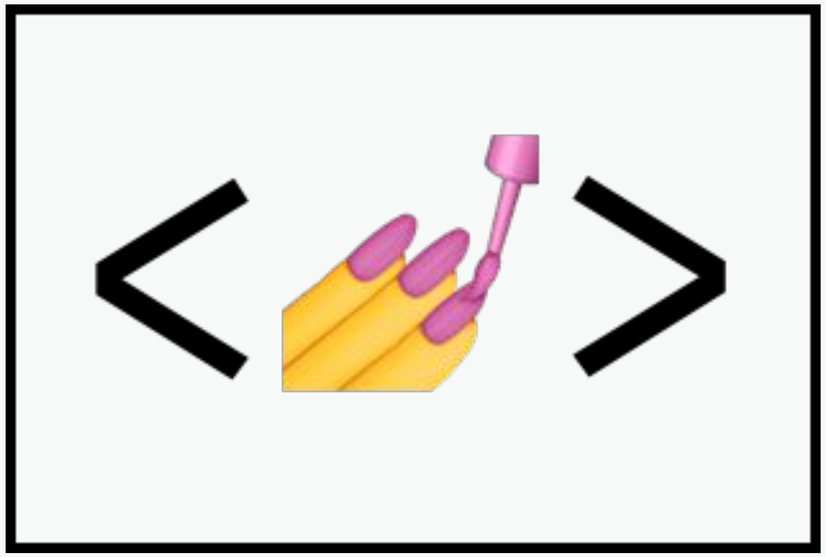
THE BRIDGE DIGITAL TALENT ACCELERATOR

¿Que es CSS-in-JS?

CSS-in-JS es una **abstracción** de CSS escrita en **JavaScript moderno**. Está basado en los CSS modules, una forma de escribir CSS que se enfoca en que **el alcance de los estilos funcione por componente** y no se filtren a ningún otro elemento de la página. Una de sus **ventajas** es que crea código más **semántico** y evita los **problemas de especificidad** que trae el CSS tradicional.
https://medium.com/@arantza.beitia/css-in-js-estilando-en-un-mundo-js-d19e88ae6c4e



Styled Components



Es una de las **librerías más populares** para **inyectar CSS en JavaScript**. **No es específica para React** ni para ningún otro framework. La puedes utilizar junto con cualquier **framework basado en componentes**.

Se tratan igual que **componentes**, por lo que podemos pasar **props**, poner **componentes dentro de otros componentes**, darle un **id**, ser **reutilizados**, etc.

https://styled-components.com/

Instalación

Styled Components

```
$ npm i -D styled-components@5.3.10
```

Importar

Importar styled components en **cada archivo** en el que lo queramos utilizar.

Recuerda **borrar los archivos de *.css** que el CLI de React genera para no tener ningún conflicto con nuestros estilos. Si queremos mantener algunos de estos estilos, podemos cambiarlos a Styled Components.

```
import styled from 'styled-components'

const Button = () => {
return <button>Botón Básico</button>
}

export default Button
```

Crear componente estilado

- **Crear un componente** con estilo basado en algún elemento html
- **Utilizar el componente** como si fuera cualquier componente de React

Notarás que **styled** utiliza **notación de punto** para obtener el elemento de HTML que queremos estilizar. En este caso, utilizamos **button**, pero igual podemos poner **div**, **h1**, o cualquier otro **elemento de HTML**.

```
import styled from 'styled-components'

const PrettyButton = styled.button`
   background-color: pink;
   border: 2px solid pink;
   border-radius: 5px;
   color: black;
   padding: 10px;
   box-shadow: 5px 5px 5px 0px lightgray;
`

const Button = () => {
   return
   <PrettyButton>Botón Básico</PrettyButton>
}

export default Button
```

Heredar Estilos de un Componente Existente

Tal vez necesitas **otro botón** idéntico al anterior, **pero** con un **color de fondo diferente**.

Tu primer impulso puede ser copiar el componente anterior y simplemente cambiarle el color.

Styled Components **nos provee una mejor manera** que no requiere **duplicar código**. Podemos utilizar el **constructor styled()**.

```
. . .
const InheritedButton = styled(PrettyButton)`
   background-color: lightblue;
   border: 2px solid lightblue;
`
const Button = () => {
   return (
      <>
         <InheritedButton>
            Boton modificado
         </InheritedButton>
      </>
   )
}
export default Button
```

Estilos condicionados a props

Una de las **funcionalidades más poderosas** de Styled Components es la habilidad de **estilizar componentes de acuerdo al estado**.

Para ello creamos un **estado nuevo** que controlará el color a rosa al hacer **click en el botón**.

```
. . .
import { useState } from 'react'

const BaseButton = () => {
const [isPink, setIsPink] = useState(false)

   return (
      <>
         <StyledButton
            isPink={isPink}
            onClick={() => setIsPink(!isPink)}>
            cambio de color!!
         </StyledButton>
      </>
   )
}
export default BaseButton
```

Estilos condicionados a props

Obtén la prop en el estilo que quieras cambiar de acuerdo al estado

```
const StyledButton = styled.button`
   background-color: ${(props) => (props.isPink ? '#fa25ef' : '#3c7f8b')};
   border: 2px solid ${(props) => (props.isPink ? '#fa25ef' : '#3c7f8b')};
   border-radius: 5px;
   color: ${(props) => (props.isPink ? '#000' : '#fff')};
   padding: 10px;

   :hover {
      box-shadow: 5px 5px 5px 0px lightgray;
   }
`
```

Estilos globales

Los **styled components** que hemos visto hasta el momento están **aislados de otros componentes**. Las clases de CSS generadas son **solo locales**. ¿Qué pasa si queremos inyectar estilos que apliquen a **toda nuestra aplicación**? Algunos ejemplos de esto pueden ser la **fuente**, **colores**, etc..

Styled Components provee una función para esto. En este caso tenemos que importar **createGlobalStyle**.

Estilos globales

1. Creamos un archivo de JavaScript en **/src/assets/styles** para los estilos globales.

```
import { createGlobalStyle } from 'styled-components'

const GlobalStyles = createGlobalStyle`
   body {
      background-color: whitesmoke;
      color: #000000;
      font-family: Arial, sans-serif;
      margin: 0;
   }
`

export default GlobalStyles
`
```

Estilos globales

2. Añadimos el componente como **primer componente en App.jsx**, en el root del árbol de React.

```
. . .
import GlobalStyles from './assets/styles/GlobalStyles'

function App() {
    return (
        <>
            <BrowserRouter>
                <GlobalStyles />
                <Header />
                <Routes>
                    <Route path="/" element={<Home />} />
                    <Route path="/user" element={<User />} />
                </Routes>
            </BrowserRouter>
        </>
    )
}

export default App
```

Temas

Podemos utilizar **temas** para cambiar los estilos de la aplicación. Es muy útil para crear una **versión dark mode y light mode**, por ejemplo.

Creamos un archivo independiente para el tema en **/src/assets/styles** llamado **Theme.jsx** y pasamos **dos objetos** con atributos **idénticos** pero con valores **diferentes**, según el tema dark o light.

```
export const lightTheme = {
  text: '#0a155a',
  bodyBackground: '#7D80AA',
}

export const darkTheme = {
  text: '#ffffff',
  bodyBackground: '#1C1938',
}
```

Temas

Ahora aplicamos a los estilos globales los valores según el tema que aplique. Para ello desestructuramos theme que pasaremos como props.

```
import { createGlobalStyle } from 'styled-components'

const GlobalStyles = createGlobalStyle`
body {
   background-color: ${({ theme }) => theme.bodyBackground};
   color: ${({ theme }) => theme.text};
   }
`

export default GlobalStyles
```

Temas

Style Components nos provee de un componente **ThemeProvider**. Lo importamos en App.jsx junto a **Theme.jsx**, desestructurando los dos objetos lightTheme y darkTheme.

Englobamos **ThemeProvider** en el root de React con el componente del tema.

Será una **prop** desde la que podemos acceder desde cada propiedad de los styled components.

```jsx
import { ThemeProvider } from 'styled-components'
import { lightTheme, darkTheme } from './assets/styles/Theme'
import GlobalStyles from './Global.styles'

function App() {
    return (
        <>
            <BrowserRouter>
                <ThemeProvider theme={darkTheme}>
                <GlobalStyles />
                    <GlobalProvider>
                        <Header />
                        <Routes>
                            <Route path="/" element={<Home />} />
                            <Route path="/user" element={<User />} />
                        </Routes>
                    </GlobalProvider>
                </ThemeProvider>
            </BrowserRouter>
        </>
    )
}

export default App
```