


React - Parte 2

Ejercicio 3 -

Creamos un nuevo componente llamado `Persona.js` dicho componente contiene:

src/Componentes/Personas.js

 Copiar código


```
import React from "react";
// Usando props de forma muy básica y simple
function Persona(props){
  return(
    <div className="persona ver">
      <h2>Nombres : {props.nombre}</h2>
      <p>Edad : {props.edad}</p>
      <p>Correo : {props.correo}</p>
    </div>
  )
}

export default Persona;
```

NOTA

Recuerda que como lo hemos trabado anteriormente siempre llamamos el componente en el archivo principal `src/App.js`

src/App.js

 Copiar código

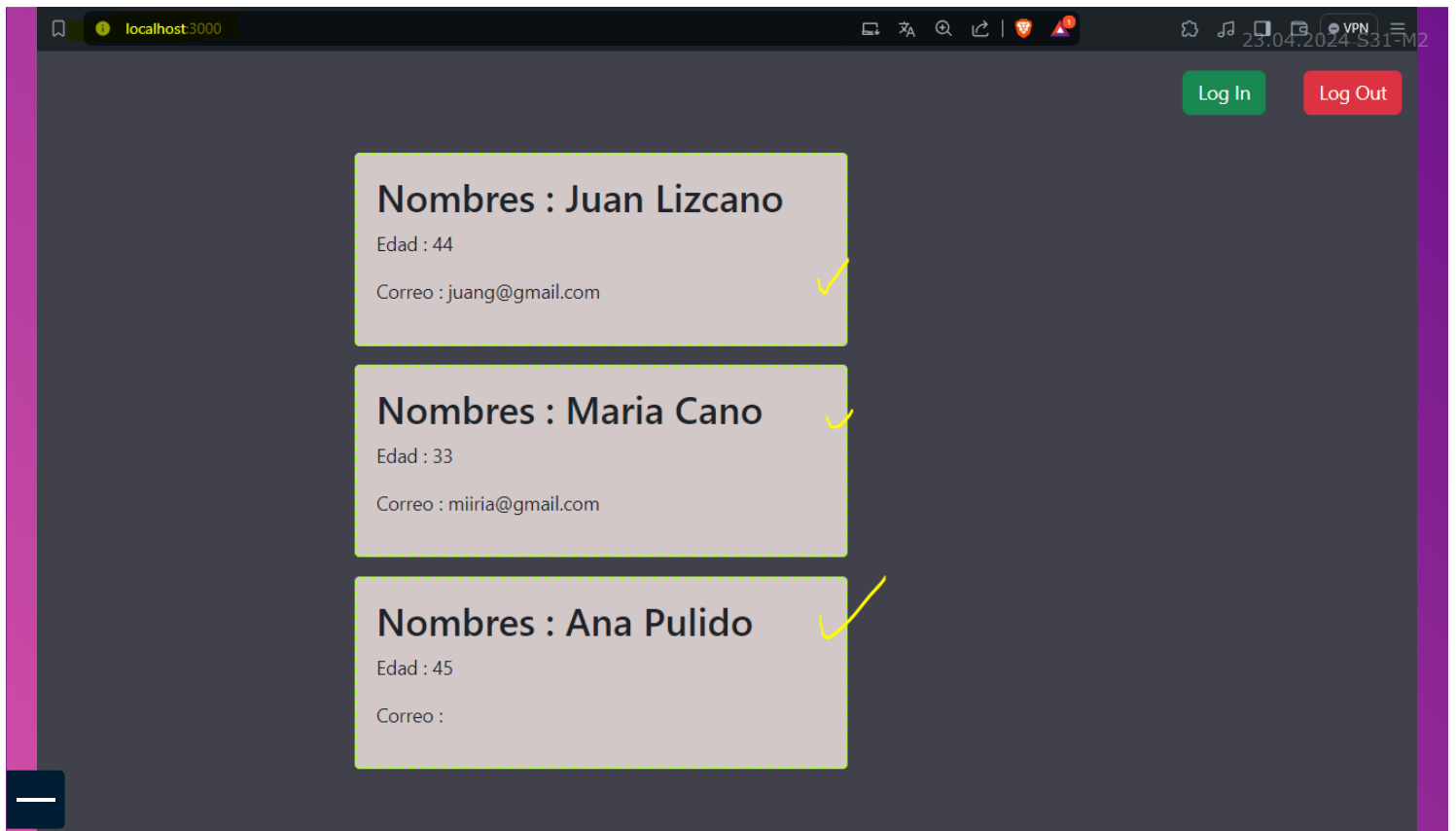
```
import './App.css';
import LoginButton from './Componentes/LoginButton';
import LogoutButton from './Componentes/LogoutButton';
import Profile from './Componentes/Profile';
import Persona from './Componentes/Persona';
```

```
function App() {  
  return (  
    <div className="App">  
      <nav className='nav-login'>  
        <LoginButton/>  
        <LogoutButton/>  
        <Profile />  
      </nav>  
  
      <Persona  
        nombre =" Juan Lizcano"  
        edad="44"  
        correo = "juang@gmail.com"  
      />  
      <Persona  
        nombre =" Maria Cano"  
        edad="33"  
        correo = "miiria@gmail.com"  
      />  
      <Persona  
        nombre =" Ana Pulido"  
        edad="45"  
        correo = ""  
      />  
    </div>  
  );  
}
```

```
export default App;
```

NOTA

Anteriormente ya habíamos aplicado unos estilos en el archivo `src/App.css` es por eso que en pantalla veras todos los componentes en acción, incluyendo los botones de Login y Logout




React con Vite

En esta parte vamos a trabajar **React** con **Vite**. Primero deberás crear un nuevo proyecto con **Vite**. Luego sigue los pasos para instalar de forma correcta React. Elige tu directorio y en la terminal ejecuta:

Si deseas puedes apoyarte de la [documentación de vite](#)

Estructura de carpetas

 Copiar código

```
→ react_vite // New
```


react_vite

 Copiar código

```
npm create vite@latest
```

Una vez establecido tus preferencia en cuanto a lenguaje y estructura de trabajo se crearía un nuevo proyecto Vite, en este caso llamado por defecto **vite-project**


Estructura de carpetas

 Copiar código

```
→ react_vite
  → vite-project // New
```

Ahora situate en **vite-project** de forma manual o directamente en la terminal ejecutando:


Terminal ...react_vite

 Copiar código

```
cd vite-project
```

Y finalmente ejecuta:

Terminal ...react_vite/vite-project

 Copiar código

```
npm install
```

Inicializando Vite + React

Para inicializar tu proyecto ejecuta en la terminal el siguiente código:

`npm run dev`

😎 Perfecto ya puedes empezar a mover coitas y jugar un rato con React y Vite! 🤖

React - Ejercicio Cards - Parte 1

En esta parta realizaremos un ejercicio donde usaremos una serie `Cards` las cuales redireccionaran a su respectivo contenido en una ruta independiente. Pare ello los datos de cada card van a estar contenidos en un único `json` el cual recorreremos un cun `.map` en nuestro archivo principal. Debes tener en cuenta que vamos a crear distintos `css` para estilizar las card y demás partes de nuestro proyecto, igualmente se modificaran algunas lineas de el `css` existente tanto del `index.css` como del `App.css`

Para esta primera parte del ejercicio vamos a crear los siguientes directorios, archivos, y ademas se modificaran algunos existentes de la siguiente forma, todo esto contenido en el directorio

`src`


Estructura de carpetas

```
↓ src
  ↓ Componentes// New
    Card.css// New
    Card.jsx// New
  ↓ datos// New
    lenguajes.js// New
  ↓ vistas// New
    VistasLenguajes.css// New
    VistasLenguajes.jsx// New
  App.css// Modified
  App.jsx// Modified
  index.css// Modified
  main.jsx
```

Definimos nuestra Card

Es simple, el único componente como tal que usaremos de momento será `Card.jsx` y éste queda configurado de la siguiente manera:

src/Componentes/Card.jsx

 Copiar código

```
import "../Componentes/Card.css"


function Card (props){
  return(
    <div className="card">
      <h2>{props.titulo}</h2>
      <p>{props.descripcion}</p>
    </div>
  )
}

export default Card;
```

NOTA

Este código define un componente llamado "Card" en React. Toma dos propiedades, "titulo" y "descripcion", y las utiliza para renderizar un elemento div con una clase "card", que contiene un título y una descripción. Este componente puede ser reutilizado en cualquier parte de la aplicación para mostrar contenido estructurado de manera consistente.

src/Componentes/Card.css


 Copiar código

```
.card{
  border: dashed 2px greenyellow;
  background-color: #d6c8c8;
  padding: 1em;
  border-radius: 4px;
  width: 30%;
  color: #242424;
}
```

Configurando nuestro Json

Es momento de definir que datos son los que van a contener nuestras **Cards**, estos estarán dentro de nuestro archivo `src/datos/lenguajes.js`, quedando configurado de la siguiente forma:

`src/datos/lenguajes.js`

 Copiar código

```
const lenguajes = [
  {
    ntitulo: "React",
    ndescripcion: "React es una biblioteca de JavaScript",
    imagen: "https://es.wikipedia.org/wiki/Archivo:React.svg",
  },
  {
    ntitulo: "SQL",
    ndescripcion: "Es un lenguaje de base de datos",
    imagen: "https://es.wikipedia.org/wiki/Archivo:Sql_database_shortcut_icon.png",
  },
  {
    ntitulo: "PHP",
    ndescripcion: "Lenguaje de programación para construir backend",
    imagen: "https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/PHP-logo.svg/200px-PHP-log",
  },
  {
    ntitulo: "C#",
    ndescripcion: "lenguaje de programación para trabajar objetos",
    imagen: "https://upload.wikimedia.org/wikipedia/commons/thumb/d/d2/C_Sharp_Logo_2023.svg/320px-",
  },
  {
    ntitulo: "Angular",
    ndescripcion: "Es un framework para frontend",
    imagen: "https://upload.wikimedia.org/wikipedia/commons/thumb/c/cf/Angular_full_color_logo.sv",
  },
  {
    ntitulo: "JAVA",
    ndescripcion: "Lenguaje de programación para hacer apps web y escritorio",
    imagen: "https://upload.wikimedia.org/wikipedia/commons/thumb/4/47/Java_Black_icon.svg/320px-",
  }
]

export default lenguajes;
```

Implementando con App.jsx

Es momento de importar nuestra `Card`, recorrer nuestros datos contenidos en `lenguajes.js` y confirmar que nuestras Cards están siendo visibles y todo funciona de momento, correctamente.

src/App.jsx

Copiar código

```
import './App.css'
import Card from './Componentes/Card'
// Importamos nuestro Json con nuestros datos
import lenguajes from './datos/lenguajes'

function App() {

  const ListasLenguajes = lenguajes.map((l, index) =>{
    return <Card key = {index} titulo = {l.ntitulo} descripcion = {l.ndescripcion}/>
  })

  return (
    <div className="app">
      <h1>React - Ejercicio Cards</h1>
      <div className="container">
        {ListasLenguajes}
      </div>
    </div>
  )
}

export default App
```

NOTA

Este código representa una aplicación de React que muestra tarjetas (cards) con información sobre diferentes lenguajes de programación. Utiliza un archivo CSS para estilos y un componente llamado "Card" para renderizar cada tarjeta. Importa un archivo JSON llamado "lenguajes" que contiene datos sobre los lenguajes de programación. Luego, utiliza el método "map" para recorrer este archivo JSON y crear una lista de tarjetas, pasando los datos relevantes de cada lenguaje como props al componente "Card". Finalmente, la aplicación renderiza el título principal y la lista de tarjetas dentro de un contenedor en el DOM.

src/App.css


Copiar código

```
/* Estilos principales para App.jsx */
.app h1{
  text-align: center;
}
```



```
.container{
  display: flex;
  justify-content: center;
  gap: 1em;
  flex-wrap: wrap;
}
```

scr/index.css

 Copiar código

```
/* Estilos adicionales scr/index.css */
:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;

  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: #242424;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

a {
  font-weight: 500;
  color: #646cff;
  text-decoration: inherit;
}
a:hover {
  color: #535bf2;
}

body {
  margin: 0;
  min-width: 320px;
  min-height: 100vh;
}

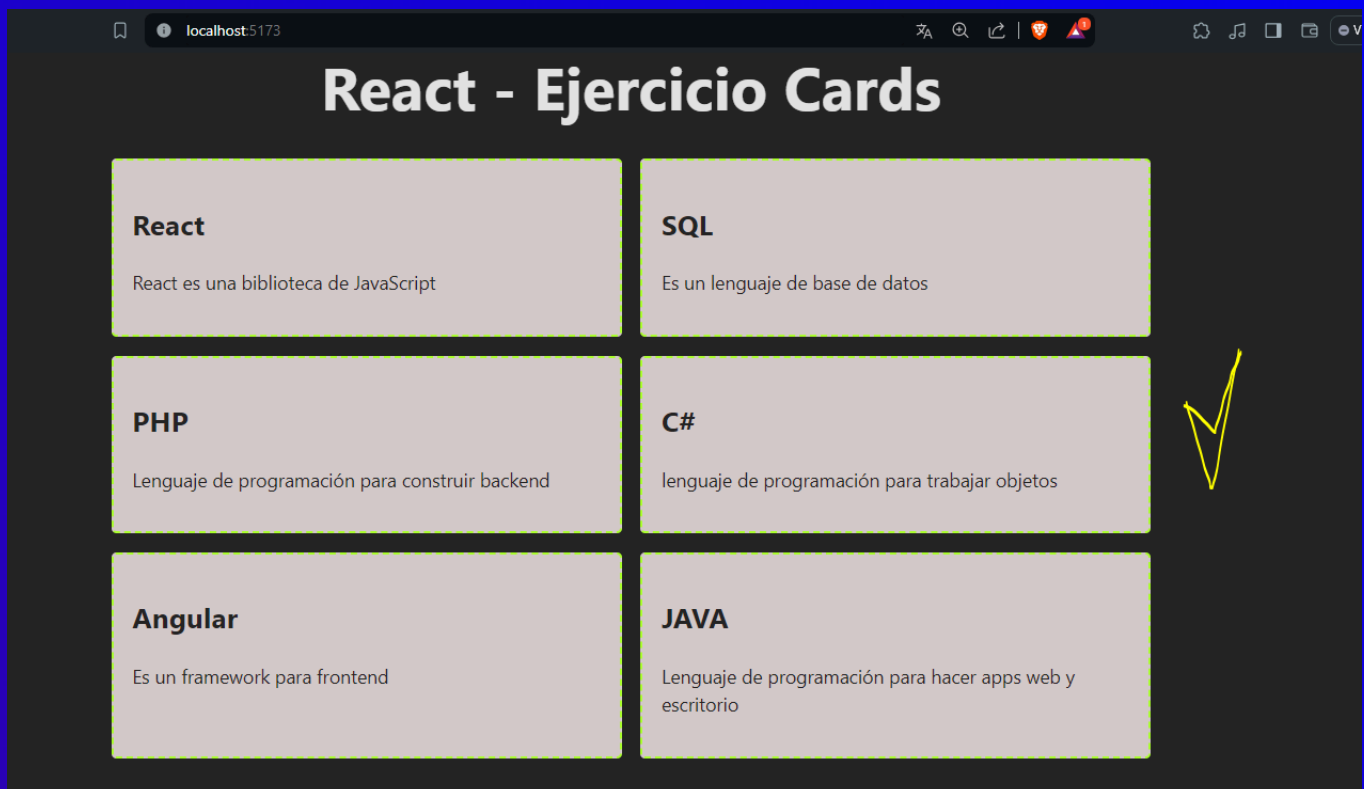
h1 {
  font-size: 3.2em;
  line-height: 1.1;
}

button {
  border-radius: 8px;
  border: 1px solid transparent;
  padding: 0.6em 1.2em;
  font-size: 1em;
  font-weight: 500;
  font-family: inherit;
  background-color: #1a1a1a;
  cursor: pointer;
  transition: border-color 0.25s;
}
button:hover {
```

```
border-color: #646cff;
}
button:focus,
button:focus-visible {
  outline: 4px auto -webkit-focus-ring-color;
}

@media (prefers-color-scheme: light) {
  :root {
    color: #213547;
    background-color: #ffffff;
  }
  a:hover {
    color: #747bff;
  }
  button {
    background-color: #f9f9f9;
  }
}
```

Comprobación de la vista del DOM desde el navegador



De esta manera damos por finalizada esta sesión, en la siguiente trabajaremos con la parte de mostrar las imágenes del `Json`, así como también dar la funcionalidad a las `cards` para que

redireccionen a otras páginas o directorios.

23.04.2024 S31-M2



Mira el repositorio en Github

Bryan Hernández | Telento Tech DWFSV2-42 | 2024

<https://4shc.online/tech/>

—