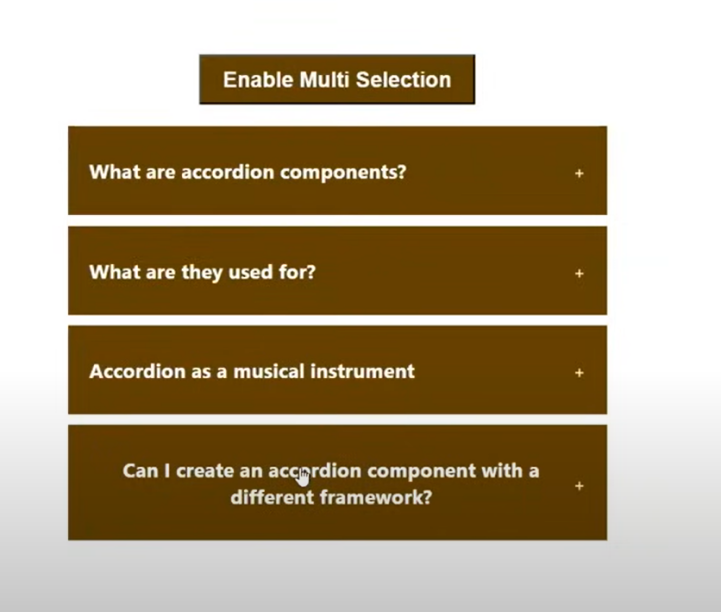
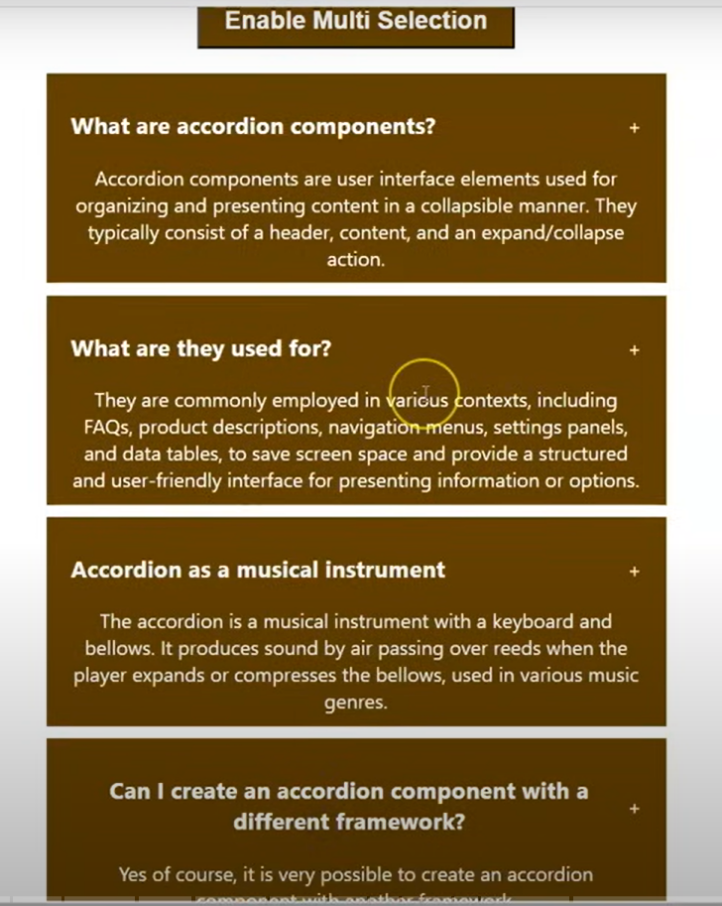
**REACT PROJECTS**

**Ejercicios en React para estudiar a fondo los conceptos**

**<https://www.youtube.com/watch?v=5ZdHfJVAY-s&t=10s>**

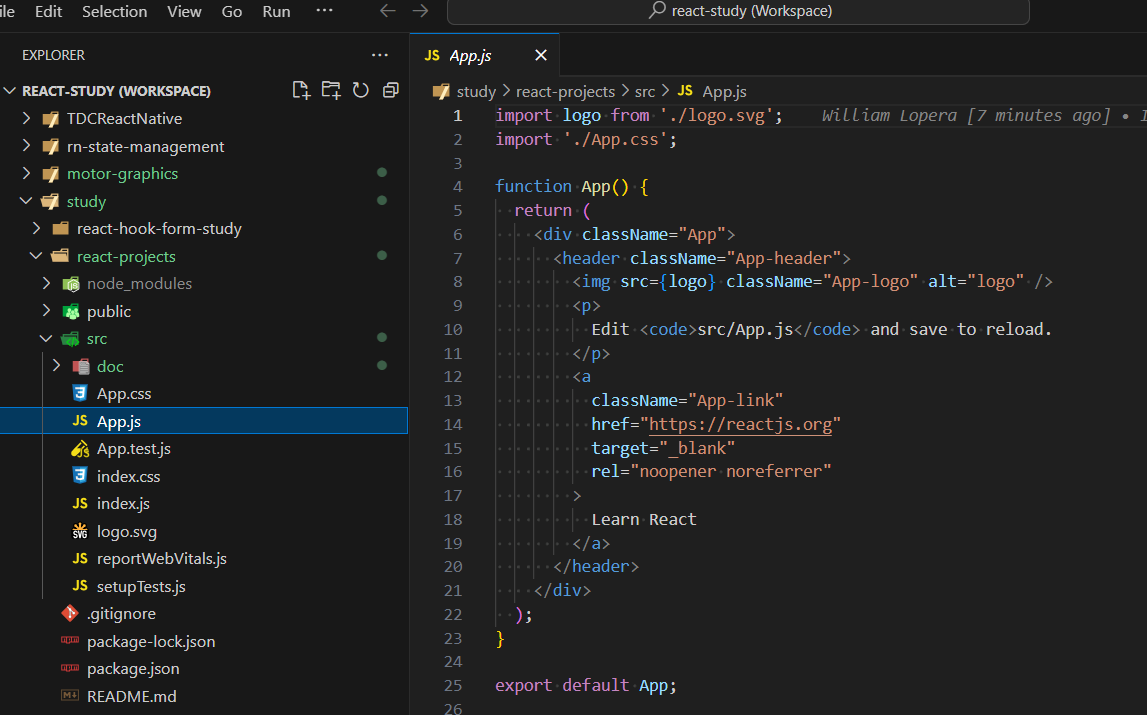
1. **Crear un acordeón de apertura simple o multi apertura a la vez**





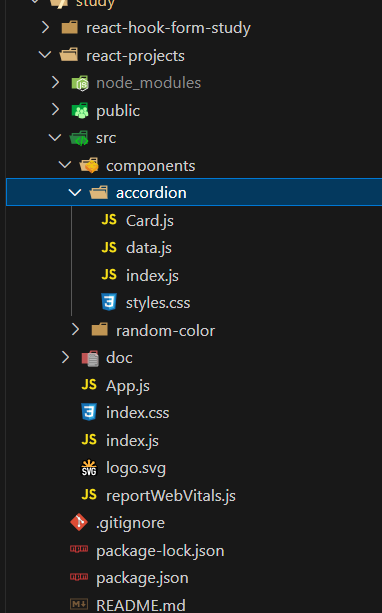
* Crear un App de react

\_ > npx create-react-app react-projects





* Código para crear Acordeón



* + Index.js

import React, { useState } from "react";

import { data } from "./data";

import { Card } from "./Card";

import "./styles.css";

export const Accordion = () => {

  const [listDisplay, setListDisplay] = useState(["1"]);

  const [enableMultiple, setEnabledMultiple] = useState(false);

  const records = data;

  const handleSetListDisplay = (id) => {

    if (enableMultiple) {

      let list = [...listDisplay];

      if (listDisplay.includes(id)) {

        list = list.filter((item) => item !== id);

      } else {

        list.push(id);

      }

      setListDisplay(list);

    } else {

      setListDisplay([id]);

    }

  };

  const handleSetEnabledMultiple = () => {

    if (enableMultiple) {

      setListDisplay(["1"]);

    }

    setEnabledMultiple(!enableMultiple);

  };

  return (

    <div className="container">

      <button onClick={handleSetEnabledMultiple} className="button">

        {enableMultiple ? "Selección Multiple" : "Selección Simple"}

      </button>

      {records &&

        records.length > 0 &&

        records.map((record) => (

          <Card

            key={record.id}

            id={record.id}

            question={record.question}

            answer={record.answer}

            showAnswer={listDisplay.includes(record.id)}

            onShowAnswer={handleSetListDisplay}

          />

        ))}

    </div>

  );

};

* card.js

import React from "react";

import "./styles.css";

export const Card = ({

  id,

  question,

  answer,

  showAnswer = false,

  onShowAnswer,

}) => {

  return (

    <div className="card-container">

      <div className={showAnswer ? "card-question-selected" : "card-question"}>

        <h5>{question}</h5>

        <div style={{ cursor: "pointer" }} onClick={() => onShowAnswer(id)}>

          {showAnswer ? "-" : "+"}

        </div>

      </div>

      {showAnswer && <div>{answer}</div>}

    </div>

  );

};

* data.js

export const data = [

  {

    id: "1",

    question: "¿Qué son los componentes del acordeón?",

    answer:

      "Los componentes de acordeón son elementos de la interfaz de usuario que se utilizan para organizar y presentar contenido de forma plegable. Por lo general, constan de un encabezado, contenido y una acción de expandir/contraer.",

  },

  {

    id: "2",

    question: "¿Para qué se utilizan?",

    answer:

      "Se emplean comúnmente en diversos contextos, incluidas preguntas frecuentes, descripciones de productos, menús de navegación, paneles de configuración y tablas de datos, para ahorrar espacio en la pantalla y proporcionar una interfaz estructurada y fácil de usar para presentar información u opciones.",

  },

  {

    id: "3",

    question: "El acordeón como instrumento musical",

    answer:

      "El acordeón es un instrumento musical con teclado y fuelle. Produce sonido mediante el paso del aire sobre las lengüetas cuando el intérprete expande o comprime el fuelle, utilizado en diversos géneros musicales.",

  },

  {

    id: "4",

    question: "¿Puedo crear un componente de acordeón con un marco diferente?",

    answer:

      "Sí, por supuesto, es muy posible crear un componente de acordeón con otro marco.",

  },

];

* + Styles.css

.container {

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

}

.button {

    margin: 10px;

}

.card-container {

    border: 1px solid lightgray;

    width: 30%;

}

.card-question {

    display: flex;

    flex-direction: row;

    justify-content: space-between;

}

.card-question-selected {

    display: flex;

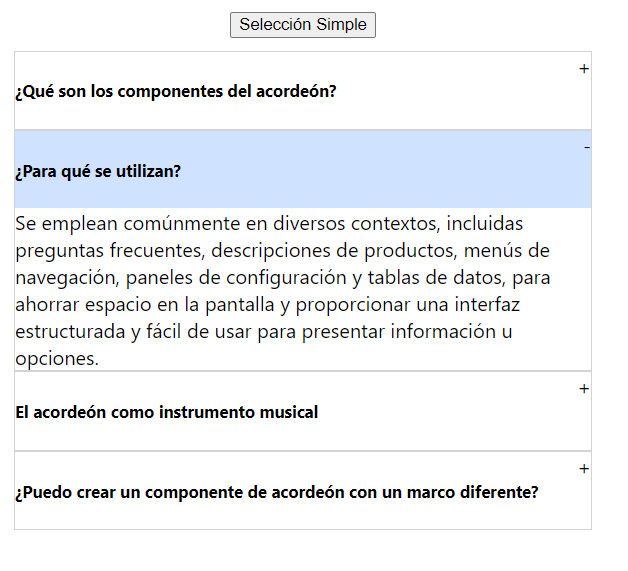
    flex-direction: row;

    background-color: #CFE2FF;

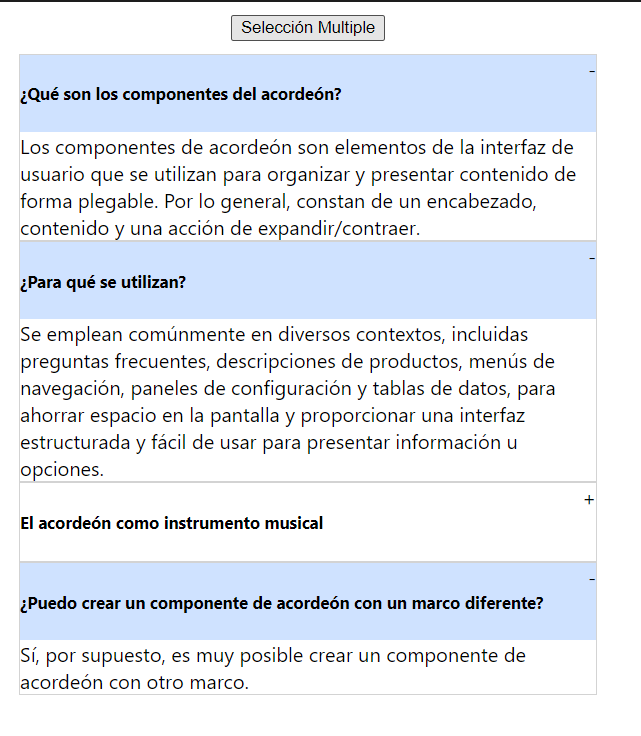
    justify-content: space-between;

}

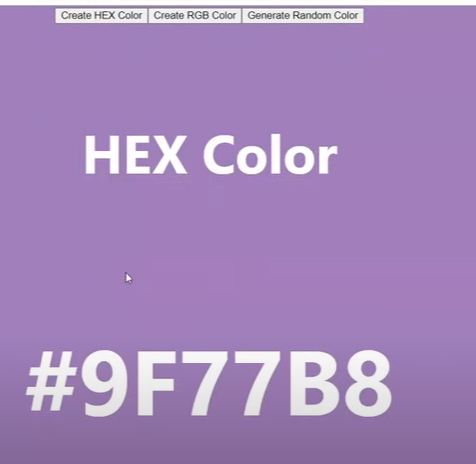
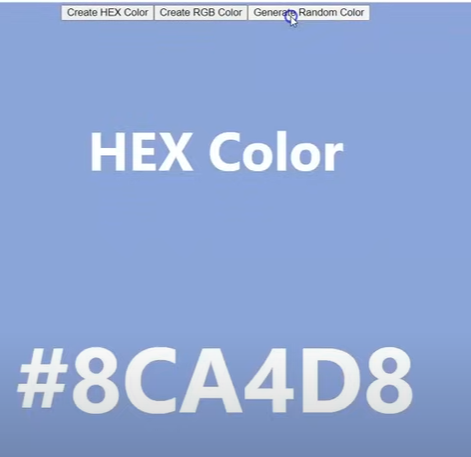
* Salida Selección Simple:

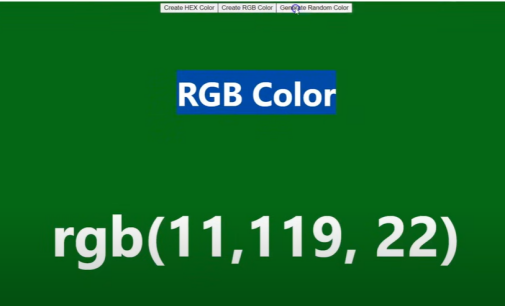
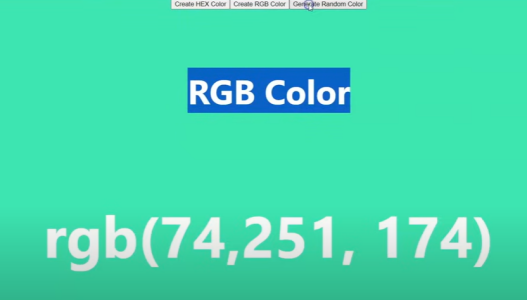


* Salida Selección Multiple:



1. **Crear un App con tres botones para mostrar colores aleatoriamente en RGB o Hexadecimal**

* Código para crear colores
* index.js

import React, { useState } from "react";

import "./styles.css";

export const RandomColor = () => {

  const [color, setColor] = useState("#FF0000");

  const [typeColor, setTypeColor] = useState("HEX");

  const [typeButton, setTypeButton] = useState("HEX");

  const getRandom15 = () => {

    return Math.floor(Math.random() \* 16)

      .toString(16)

      .toUpperCase();

  };

  const getRandom255 = () => {

    return Math.floor(Math.random() \* 256);

  };

  const getColorHEX = () => {

    let hex = "#";

    for (let index = 0; index < 6; index++) {

      hex += getRandom15();

    }

    return hex;

  };

  const getColorRGB = () => {

    return `rgb(${getRandom255()},${getRandom255()},${getRandom255()})`;

  };

  const handleColorHex = () => {

    setColor(getColorHEX());

    setTypeColor("HEX");

    setTypeButton("HEX");

  };

  const handleColorRGB = () => {

    setColor(getColorRGB());

    setTypeColor("RGB");

    setTypeButton("RGB");

  };

  const getColorRandom = () => {

    setTypeButton("RAN");

    if (typeColor === "RGB") {

      setColor(getColorRGB());

    } else {

      setColor(getColorHEX());

    }

  };

  const divColor = {

    backgroundColor: color,

  };

  return (

    <div className="color-container" style={divColor}>

      <div className="color-button-container">

        <button

          onClick={handleColorHex}

          style={

            typeButton === "HEX"

              ? { backgroundColor: "#6d99f1" }

              : { backgroundColor: "" }

          }

        >

          Color HEX

        </button>

        <button

          onClick={handleColorRGB}

          style={

            typeButton === "RGB"

              ? { backgroundColor: "#6d99f1" }

              : { backgroundColor: "" }

          }

        >

          Color RGB

        </button>

        <button

          onClick={getColorRandom}

          style={

            typeButton === "RAN"

              ? { backgroundColor: "#6d99f1" }

              : { backgroundColor: "" }

          }

        >

          Generar Color

        </button>

      </div>

      <div className="color-txt">{`Color ${typeColor}`}</div>

      <div className="color-txt-color">{color}</div>

    </div>

  );

};

* styles.css

.color-container {

    display: flex;

    flex-direction: column;

    justify-content:flex-start;

    align-items: center;

    width: 100vw;

    height: 100vh;

}

.color-button-container{

    margin: 20px;

    display: flex;

    gap: 15px;

}

.color-txt{

    font-size: 60px;

    font-weight: bold;

    color: white;

    margin: 60px;

}

.color-txt-color {

    font-size: 80px;

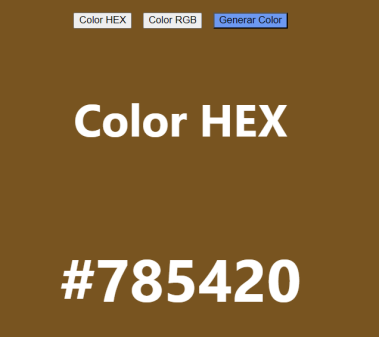
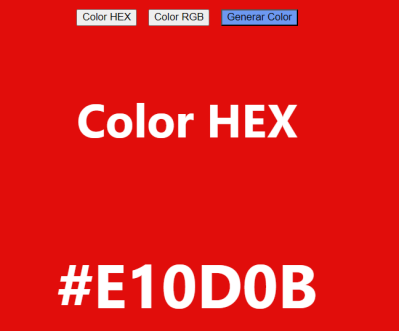
    font-weight: bold;

    color: white;

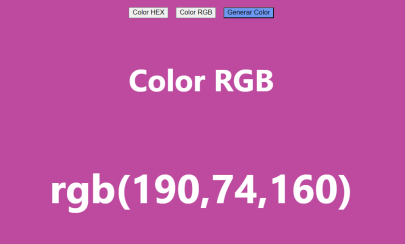
    margin: 60px;

}

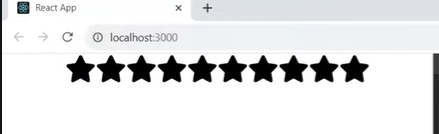
* Salida HEX:

* Salida RGB:

1. **Seleccionar Raiting - Calificación de estrellas**





* Instalación de libraría para imágenes (icon start)

npm install react-icons --save

* Código
* index.js

import React, { useState } from "react";

import { Star } from "./star";

import "./styles.css";

const arr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

export const StarRating = () => {

  const [currentSeleted, setCurrentSeleted] = useState([]);

  const [hoverSeleted, setHoverSeleted] = useState([]);

  const getArrSeleted = (id) => {

    const arr = [];

    for (let index = 0; index < id + 1; index++) {

      arr.push(index);

    }

    return arr;

  };

  const handleClick = (id) => {

    setCurrentSeleted(getArrSeleted(id));

  };

  const handleMouseEnter = (id) => {

    setHoverSeleted(getArrSeleted(id));

  };

  const handleMouseLeave = (id) => {

    setHoverSeleted(currentSeleted);

  };

  const selected = hoverSeleted.length > 0 ? hoverSeleted : currentSeleted;

  return (

    <div className="container">

      {arr &&

        arr.map((item) => (

          <Star

            key={item}

            id={item}

            isSeleted={selected.includes(item)}

            onPress={handleClick}

            mouseEnter={handleMouseEnter}

            mouseLeave={handleMouseLeave}

            size={40}          />

        ))}

      <button

        className="button"

        onClick={() => {

          setHoverSeleted([]);

          setCurrentSeleted([]);

        }}

      >

        Limpiar

      </button>

    </div>

  );

};

* star.js

import React from "react";

import { FaStar } from "react-icons/fa";

import "./styles.css";

export const Star = ({

  id,

  isSeleted,

  onPress,

  mouseEnter,

  mouseLeave,

  size,

}) => {

  return (

    <FaStar

      className={isSeleted ? "star-selected" : "star"}

      size={size}

      onClick={() => onPress(id)}

      onMouseEnter={() => mouseEnter(id)}

      onMouseLeave={() => mouseLeave(id)}

    />

  );

};

* styles.css

.container {

    display: flex;

    justify-content: center;

    align-items: center;

    margin: 20px;

}

.star {

    cursor: pointer;

}

.star-selected {

    color: yellow;

}

.star:hover {

    color: yellow;

}

.button{

    border: none;

    background-color: transparent;

    cursor: pointer;

    text-decoration: underline;

}

.button:hover {

    color: blue;

}

* Salida:











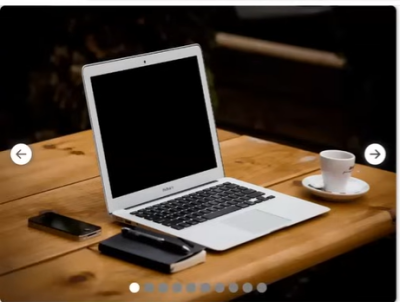
* + Luego de click en limpiar



1. **Crear un carrusel de imágenes**

* Crear un carrusel que muestre varias imágenes
* Utilizar esta API de imágenes

<https://picsum.photos/v2/list?page=2&limit=100>

* Utilizar la libraría para imágenes (icon start)

npm install react-icons --save

import { BsArrowLeftCircleFill, BsArrowRightCircleFill } from "react-icons/bs";

* Código
* index.js

import React, { useEffect, useState } from "react";

import { BsArrowLeftCircleFill, BsArrowRightCircleFill } from "react-icons/bs";

import "./styles.css";

export const ImageSlider = () => {

  const [images, setImages] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(false);

  useEffect(() => {

    const getListData = async () => {

      try {

        const pageRd = Math.floor(Math.random() \* 50) + 3;

        const limitRd = Math.floor(Math.random() \* 10) + 5;

        const response = await fetch(

          "https://picsum.photos/v2/list?page=" + pageRd + "&limit=" + limitRd

        );

        const resJson = await response.json();

        const list = resJson.map((item, index) => ({

          key: item.id,

          id: index,

          url: item.download\_url,

          selected: false,

        }));

        list[0].selected = true;

        setImages(list);

        setLoading(false);

      } catch (err) {

        console.log("[ERROR]: ", err);

        setError(true);

      }

    };

    if (loading) {

      setError(false);

      getListData();

    }

  }, [loading]);

  const handlePrevious = () => {

    const image = images.find((item) => item.selected);

    if (image.id === 0) {

      return;

    }

    updateImage(image.id - 1);

  };

  const handleNext = () => {

    const image = images.find((item) => item.selected);

    if (image.id === images.length - 1) {

      return;

    }

    updateImage(image.id + 1);

  };

  const updateImage = (id) => {

    let currentData = [...images];

    currentData = currentData.map((item) => {

      if (item.id === id) {

        item.selected = true;

      } else {

        item.selected = false;

      }

      return item;

    });

    setImages(currentData);

  };

  if (error) {

    return <div>Error Cargando datos. Intente nuevamente...!</div>;

  }

  if (loading) {

    return <div>Cargando datos. Favor espere...!</div>;

  }

  return (

    <div className="container">

      <div className="area">

        <BsArrowLeftCircleFill

          className="arrow arrow-left"

          onClick={handlePrevious}

        />

        {images &&

          images.map((image) => (

            <img

              key={image.id}

              src={image.url}

              alt={image.url}

              width="100%"

              height="100%"

              className={image.selected ? "image" : "hideImage"}

            />

          ))}

        <BsArrowRightCircleFill

          className="arrow arrow-right"

          style={{ right: "10px" }}

          onClick={handleNext}

        />

        <div className="circle-container">

          {images &&

            images.map((item) => (

              <div

                key={item.id}

                className={item.selected ? "cicle-selected" : "cicle"}

              ></div>

            ))}

        </div>

        <div className="button-container">

          <button onClick={() => setLoading(true)} className="button">

            Consultar

          </button>

        </div>

      </div>

    </div>

  );

};

* styles.css

.container {

    width: 100vw;

    height: 100vh;

}

.area {

    position: relative;

    width: 45vw;

    height: 75vh;

    margin: 5px;

}

.image {

    color: white;

    cursor: pointer;

    border-radius: 0.5rem;

    box-shadow: 0px 0px 7px #666;

  }

  .hideImage {

    display: none;

  }

  .arrow {

    position: absolute;

    top: 50%;

    transform: translateY(-50%);

    cursor: pointer;

    color: #fff;

    filter: drop-shadow(0px 0px 5px #555);

  }

  .arrow-left {

    left: 1rem;

  }

  .arrow-right {

    right: 1rem;

  }

  .circle-container {

    display: flex;

    bottom: 1rem;

    flex-direction: row;

    position: absolute;

    justify-content: center;

    align-items: center;

    width: 100%;

  }

  .cicle {

    width: 10px;

    height: 10px;

    border-radius: 50px;

    background-color: grey;

    margin: 2px;

  }

  .cicle-selected {

    width: 10px;

    height: 10px;

    border-radius: 50px;

    background-color: white;

    margin: 2px;

  }

  .button-container {

    display: flex;

    justify-content: center;

    align-content: center;

    width: 100%;

    margin-top: 0.5rem;

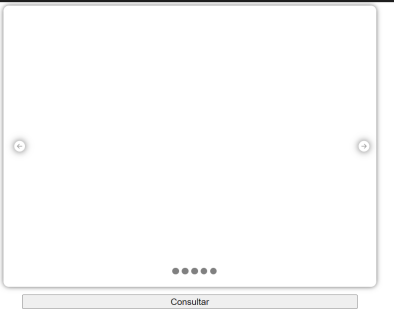
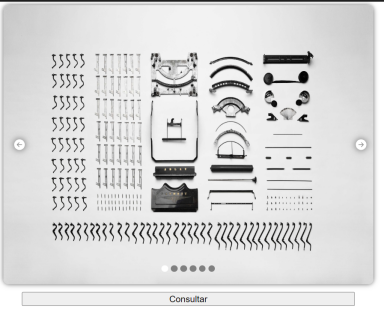
  }

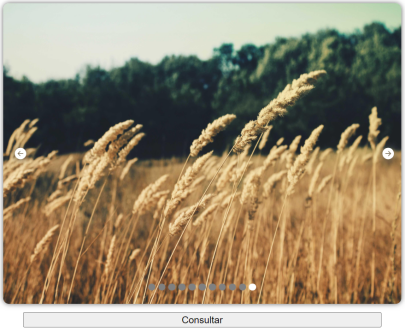
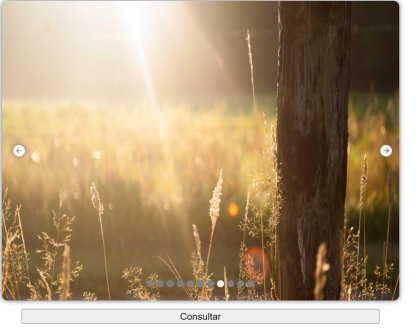
  .button {

    width: 90%;

  }

* Salida

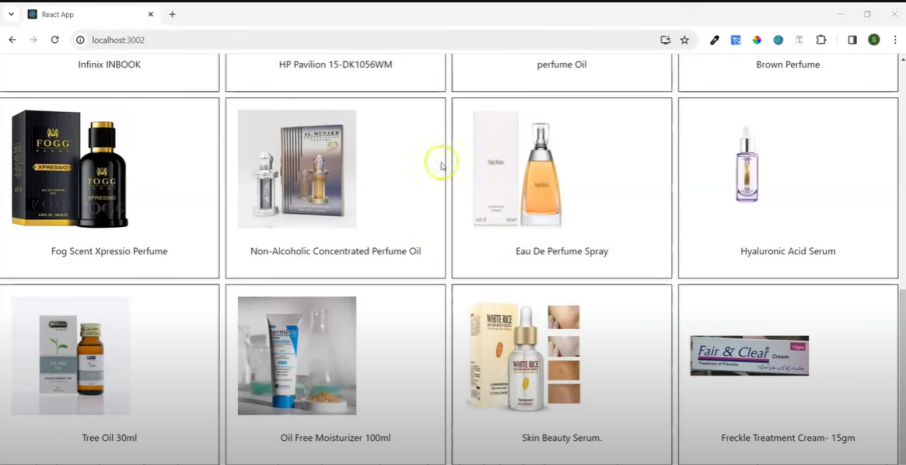
  



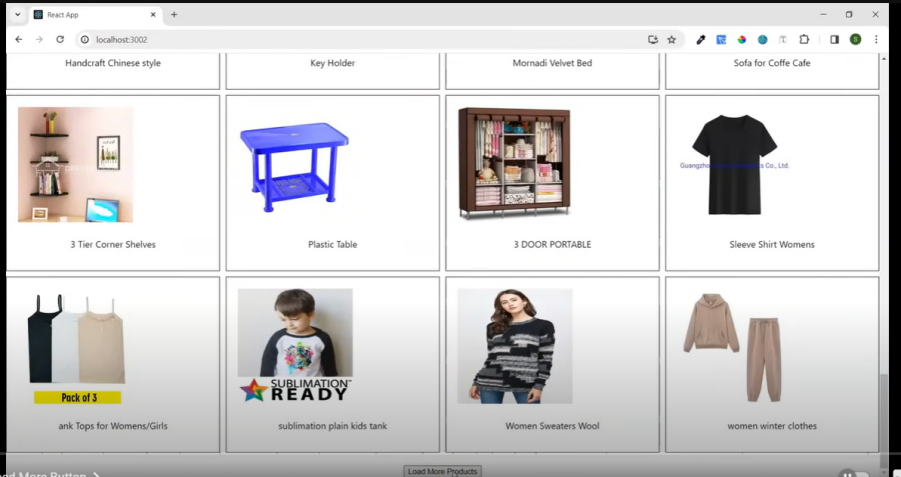
1. **Crear una App que permita cargar imágenes “infinitamente”**

* Utilizar el siguiente link en pasos de 20 en 20

> <https://dummyjson.com/products?limit=20&skip=>



* Cuenta con un botón que debe agregar 20 imágenes adicionales en cada caso hasta llegar a 100



* + Luego de 100 imagenes el boton debe desactivarse.
* Código
* index.js

import React, { useEffect, useState } from "react";

import "./styles.css";

export const LoadMoreData = () => {

  const [images, setImages] = useState([]);

  const [count, setCount] = useState(0);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    const getImages = async () => {

      try {

        const response = await fetch(

          "https://dummyjson.com/products?limit=20&skip=" + count

        );

        const resJson = await response.json();

        const data = resJson.products.map((item) => ({

          key: item.id,

          id: item.id,

          url: item.images[0],

        }));

        setImages((prevImages) => {

          // El uso de React.StrictMode me esta duplicando la data

          if (prevImages.length !== 0 && prevImages.length !== count) {

            return prevImages;

          }

          return [...prevImages, ...data];

        });

        setLoading(false);

      } catch (error) {

        console.log("[ERROR]: ", error);

      }

    };

    getImages();

  }, [count]);

  const handleGetMoreImages = () => {

    setCount((prevCount) => prevCount + 20);

  };

  if (loading) {

    return <div className="loading">Cargando imágenes. Por favor espere.</div>;

  }

  return (

    <div className="container">

      <div className="grid-container">

        {images.map((image) => (

          <img

            key={image.id}

            src={image.url}

            alt={`Diapositiva ${image.id}`}

            className="image"

          />

        ))}

      </div>

      <div className="button-area">

        <button onClick={handleGetMoreImages} disabled={count >= 100}>

          {count >= 100

            ? "Has llegaqdo al tope de productos"

            : "Cargar Productos"}

        </button>

      </div>

    </div>

  );

};

* styles.css

.container {

    width: 100vw;

    height: 100vh;

}

.loading {

    background-color: rgb(157, 157, 251);

    color: blue;

    text-align: center;

    width: 99%;

}

.grid-container {

    display: flex;

    justify-content: space-around;

    flex-wrap: wrap;

    gap: 0.5rem;

    width: 99%;

}

.image {

    width: 18rem;

    height: 15rem;

    border: 1px solid grey;

}

.button-area {

    display: flex;

    width: 100%;

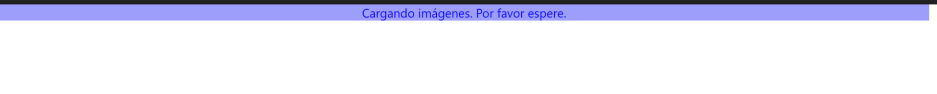
    justify-content: center;

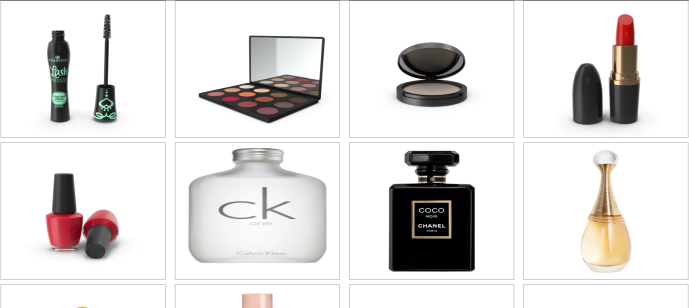
    align-items: center;

    margin-top: 1rem;

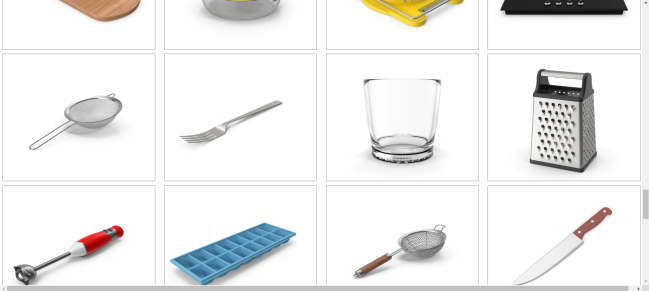
}

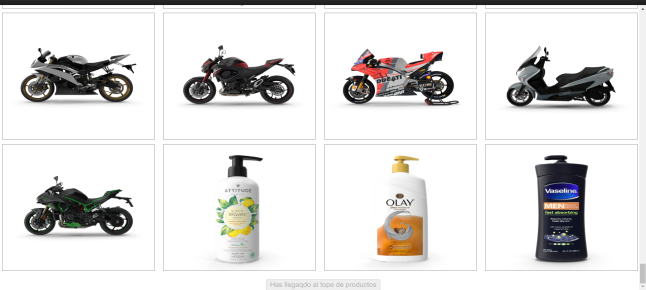
* Salida



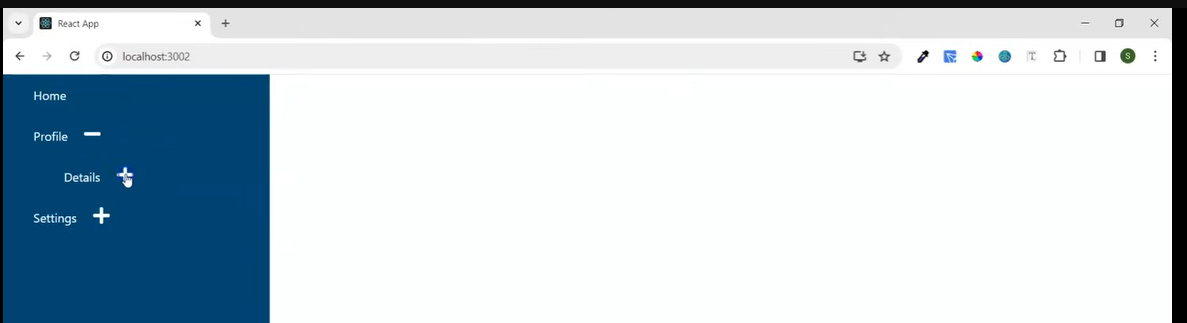


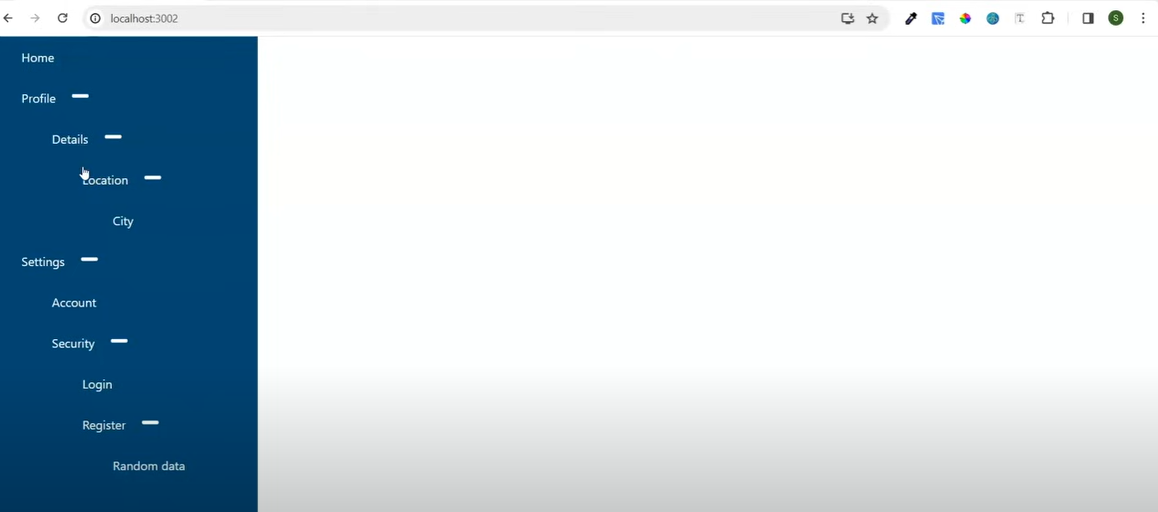






1. **Menú de opciones**





* Data Json de entrada

menus = [

{

label: "Home",

to: "/",

},

{

label: "Profile",

to: "/profile",

children: [

{

label: "Details",

to: "details",

children: [

{

label: "Location",

to: "location",

children: [

{

label: "City",

to: "city",

},

],

},

],

},

],

},

{

label: "Settings",

to: "/settings",

children: [

{

label: "Account",

to: "account",

},

{

label: "Security",

to: "security",

children: [

{

label: "Login",

to: "login",

},

{

label: "Register",

to: "register",

children : [

{

label : 'Random data',

to : ''

}

]

},

],

},

],

},

];

* Uso de imágenes

import {FaMinus, FaPlus} from 'react-icons/fa'

* Código
  + index.js

import React from "react";

import { menus } from "./data";

import "./styles.css";

import { MenuList } from "./MenuList";

export const TreeView = () => {

  return (

    <div className="container">

      <div className="menu">

        <MenuList list={menus} />

      </div>

      <div className="body">Body</div>

    </div>

  );

};

* MenuList.js

import React from "react";

import { MenuItem } from "./MenuItem";

export const MenuList = ({ list } = []) => {

  return (

    <ul className="list-ul">

      {list && list.length > 0

        ? list.map((item, index) => <MenuItem key={index} item={item} />)

        : null}

    </ul>

  );

};

* MenuItem.js

import React, { useState } from "react";

import { MenuList } from "./MenuList";

import { FaMinus, FaPlus } from "react-icons/fa";

export const MenuItem = ({ item }) => {

  const [show, setShow] = useState(false);

  const handleSetShow = () => {

    setShow((prevShow) => !prevShow);

  };

  const controlChildren = () => {

    if (item.children) {

      if (item.children.length && show) {

        return (

          <FaMinus onClick={handleSetShow} className="item-icon" size={15} />

        );

      }

      return <FaPlus onClick={handleSetShow} className="item-icon" size={15} />;

    }

    return "";

  };

  return (

    <li>

      <div className="item-container">

        <p>{item.label}</p>

        {controlChildren()}

      </div>

      {item.children && item.children.length > 0 && show ? (

        <MenuList list={item.children} />

      ) : null}

    </li>

  );

};

* styles.css

.container {

    display: flex;

    width: 100vw;

    height: 100vh;

}

.menu {

    background-color: #0047AB;

    color: white;

    width: 25%;

    overflow-y: auto;

}

.body {

    width: 75%;

}

.list-ul {

    list-style-type: none;

}

.item-container {

   display: flex;

}

.item-icon {

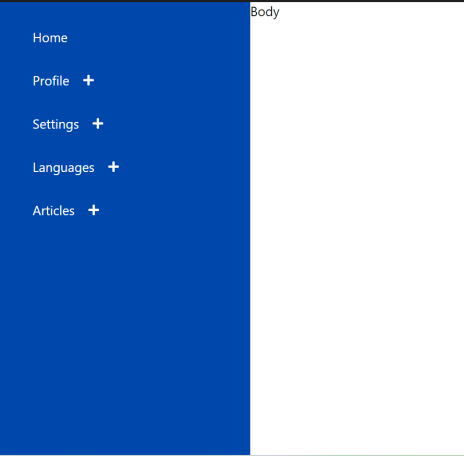
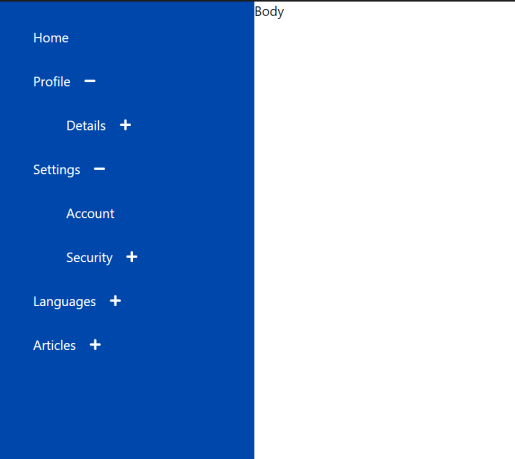
    padding-left: 1rem;

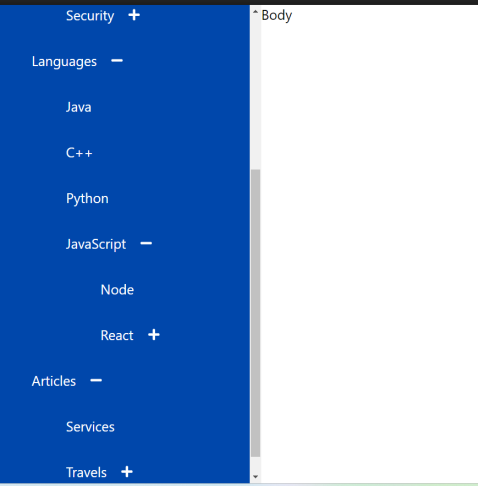
    align-self: center;

    cursor: pointer;

}

* Salida:



1. **Crear un código QR dando un texto**



Generar un código QR dado el texto de entrada. Debe utilizar la librería

> npm install react-qr-code

* Código
  + index.js

import React, { useState } from "react";

import QRCode from "react-qr-code";

import "./styles.css";

export const QRCodeGenerator = () => {

  const [name, setName] = useState("");

  const [qr, setQr] = useState(null);

  const generarQR = () => {

    if (name.length > 0) {

      setQr(<QRCode value={name} />);

    }

  };

  return (

    <div className="container">

      <h2>QR Code Generator</h2>

      <div>

        <input

          name="name"

          value={name}

          onChange={(e) => setName(e.target.value)}

        />

        <button onClick={generarQR}>Generar</button>

      </div>

      <div className="qr-area">{qr}</div>

    </div>

  );

};

* + styles.css

.container {

    display: flex;

    width: 100vw;

    height: 100vh;

    flex-direction: column;

    align-items: center;

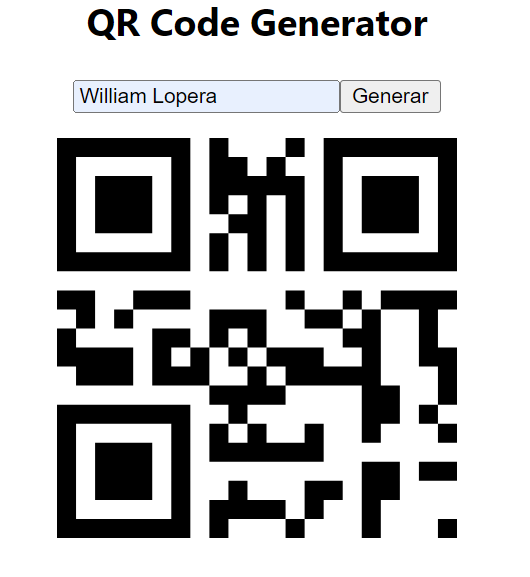
}

.qr-area {

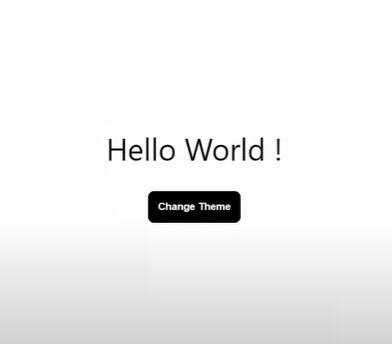
    margin-top: 1rem;

}

* Salida

1. **Crear *useState* para cambiar fondo de oscuro (dark) <--> claro (light)**

* Crear un *useState* para cambiar la pantalla de claro a oscuro o viceversa. Almacenar el valor en el *LocalStorage*.
* Código
  + index.js

import React from "react";

import { useChangeTheme } from "./useChangeTheme";

import "./styles.css";

export const LightDarkMode = () => {

  const [theme, setTheme] = useChangeTheme("theme", "light");

  const handleTheme = () => {

    setTheme((prevTheme) => (prevTheme === "light" ? "dark" : "light"));

  };

  return (

    <div className="container" data-theme={theme}>

      <p className="text">Hola Mundo!</p>

      <div className="button-container">

        <button className="button" onClick={handleTheme}>

          Cambiar a {theme === "dark" ? "claro" : "oscuro"}

        </button>

      </div>

    </div>

  );

};

* + useChangeTheme.js

import { useEffect, useState } from "react";

const init = (key, initValue) => {

  try {

    const valueStorage = localStorage.getItem(key);

    return valueStorage ? JSON.parse(valueStorage) : initValue;

  } catch (err) {

    console.log("[Error]: Problema al consultar el localStorage");

  }

};

export const useChangeTheme = (key, initValue) => {

  // Inicializa valor

  const [value, setValue] = useState(init(key, initValue));

  // Actualiza el valor al cambiar el tema

  useEffect(() => {

    localStorage.setItem(key, JSON.stringify(value));

  }, [key, value]);

  return [value, setValue];

};

* + styles.css

:root{

    --background: white;

    --text-primary: black;

    --background-button: black;

    --color-button: white

}

[data-theme="dark"] {

    --background: black;

    --text-primary: white;

    --background-button: white;

    --color-button: black

}

.container {

    display: flex;

    align-content: center;

    justify-content: center;

    flex-direction: column;

    align-items: center;

    width: 100vw;

    height: 100vh;

    background-color: var(--background);

}

.text {

    height: 1.5rem;

    font-size: 2rem;

    color:var(--text-primary)

}

.button-container{

    padding-top: 0.5rem;

    border: none;

}

.button {

    width: 10vw;

    height: 10vh;

    border-radius: 5px;

    border: none;

    background-color: var(--background-button) ;

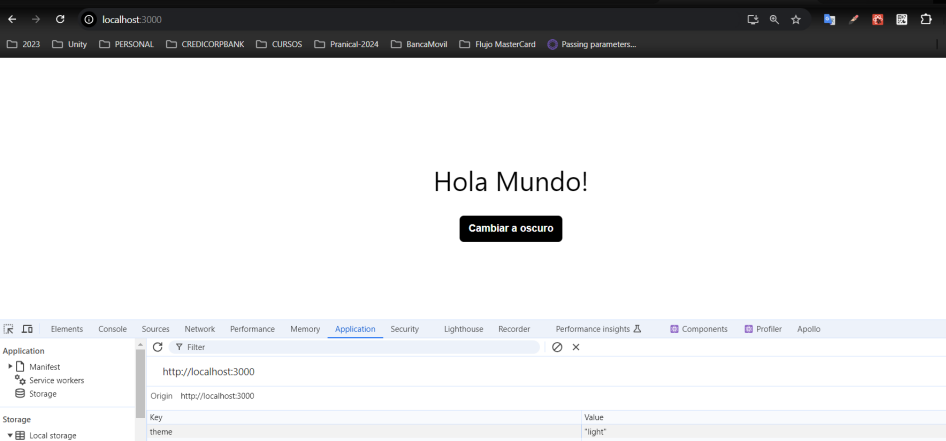
    color: var(--color-button);

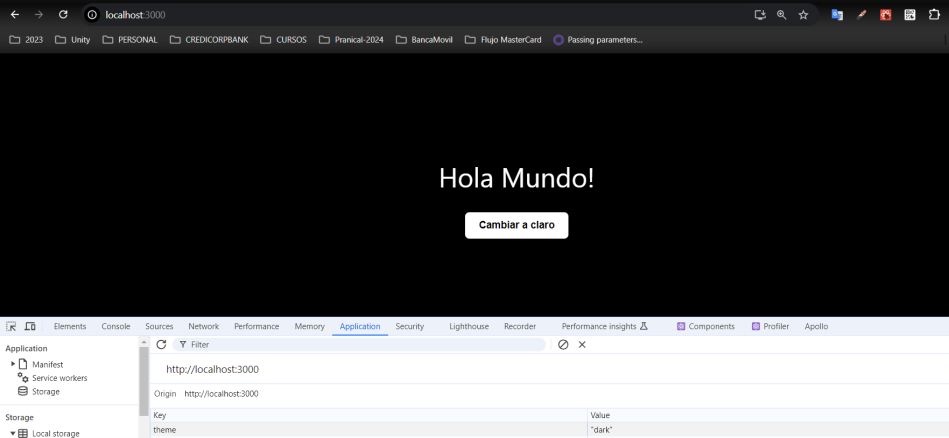
    font-weight: bold;

    font-size: 12px;

}

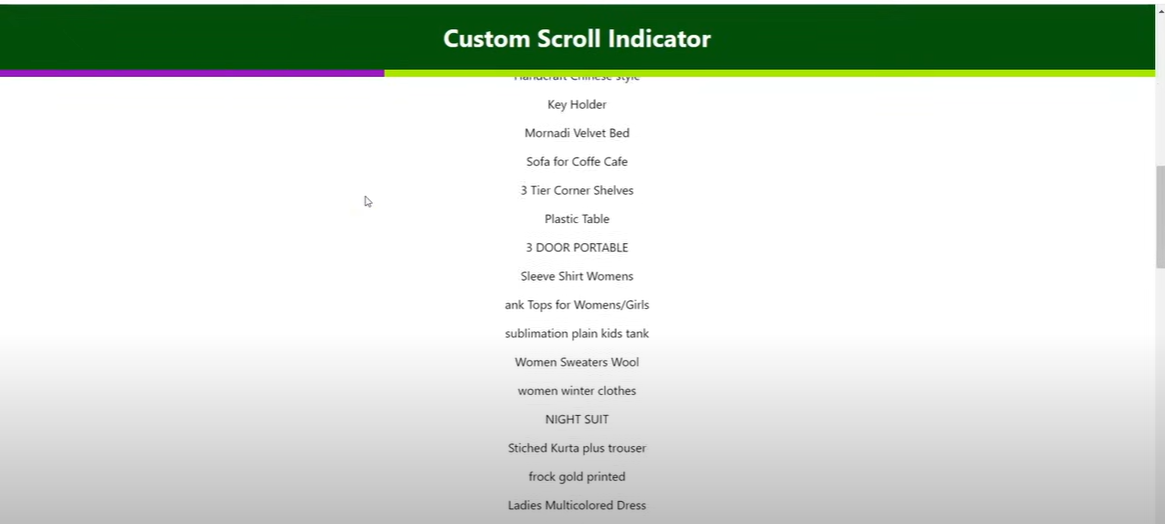
* Salida

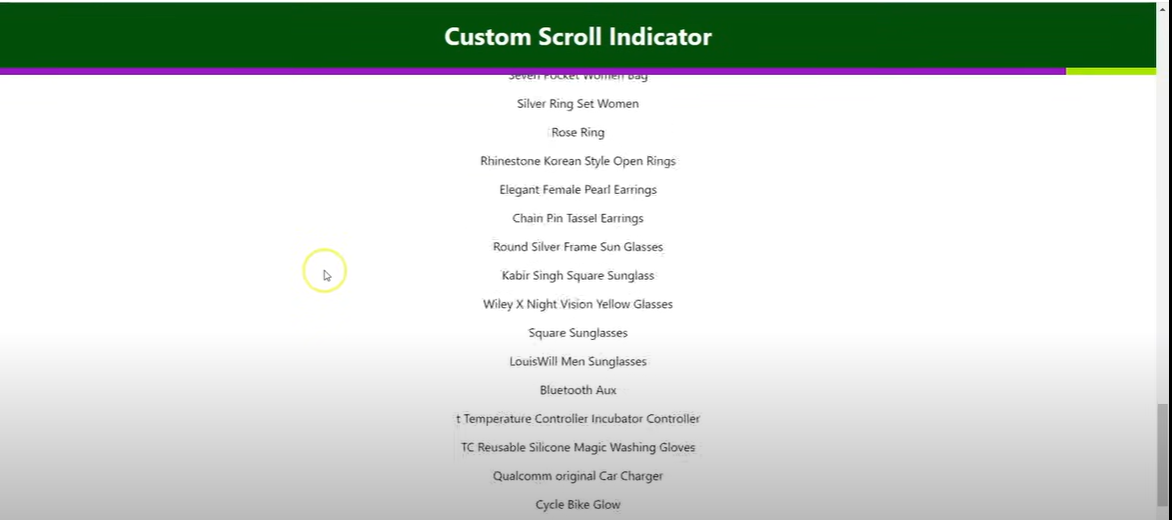




1. **Aplicación para avanzar (scroll) de una vista e indicar su avance a medida que se avance en en scroll.**









* Datos:

> https://dummyjson.com/products?limit=100

* Código
  + index.js

import React, { useEffect, useState } from "react";

import "./styles.css";

export const ScrollIndicator = () => {

  const [data, setData] = useState([]);

  const [percentage, setPercentage] = useState(0);

  useEffect(() => {

    const getData = async () => {

      const response = await fetch("https://dummyjson.com/products?limit=100");

      const resJson = await response.json();

      setData(

        resJson.products.map((item) => ({

          key: item.id,

          title: item.title,

        }))

      );

    };

    getData();

  }, []);

  useEffect(() => {

    window.addEventListener("scroll", handleScrollPercentage);

    return () => window.removeEventListener("scroll", () => {});

  }, []);

  const handleScrollPercentage = () => {

    const scrollTop = document.documentElement.scrollTop; // Cantidad de píxeles desplazados

    const scrollHeight = document.documentElement.scrollHeight; // Altura total del documento

    const clientHeight = document.documentElement.clientHeight; // Altura del viewport (pantalla)

    // Calcular el porcentaje de scroll

    const scrollPercent = (scrollTop / (scrollHeight - clientHeight)) \* 100;

    console.log("scrollPercent:", parseInt(Math.round(scrollPercent)));

    setPercentage(parseInt(Math.round(scrollPercent)));

  };

  return (

    <div className="container">

      <div className="title">{`Indicador de avance: ${percentage}%`}</div>

      <div className="advance">

        <div

          style={{

            width: `${percentage}%`,

            backgroundColor: "#8b02b5",

          }}

        ></div>

      </div>

      <div className="data">

        {data.map((item) => (

          <p key={item.key}>{item.title}</p>

        ))}

      </div>

    </div>

  );

};

* + styles.css

.container {

    display: flex;

    width: 100vw;

    height: 100vh;

    flex-direction: column;

}

.title {

    display:flex;

    justify-content: center;

    align-items: center;

    background-color: #075b0a;

    font-size: 25px;

    font-weight: bold;

    color: white;

    width: 100%;

    height: 3rem;

    position: fixed;

}

.advance {

    display:flex;

    background-color: #aaf900;

    width: 100%;

    height: 0.5rem;

    margin-top: 3rem;

    position: fixed;

}

.data {

    align-self: center;

}

.percentage {

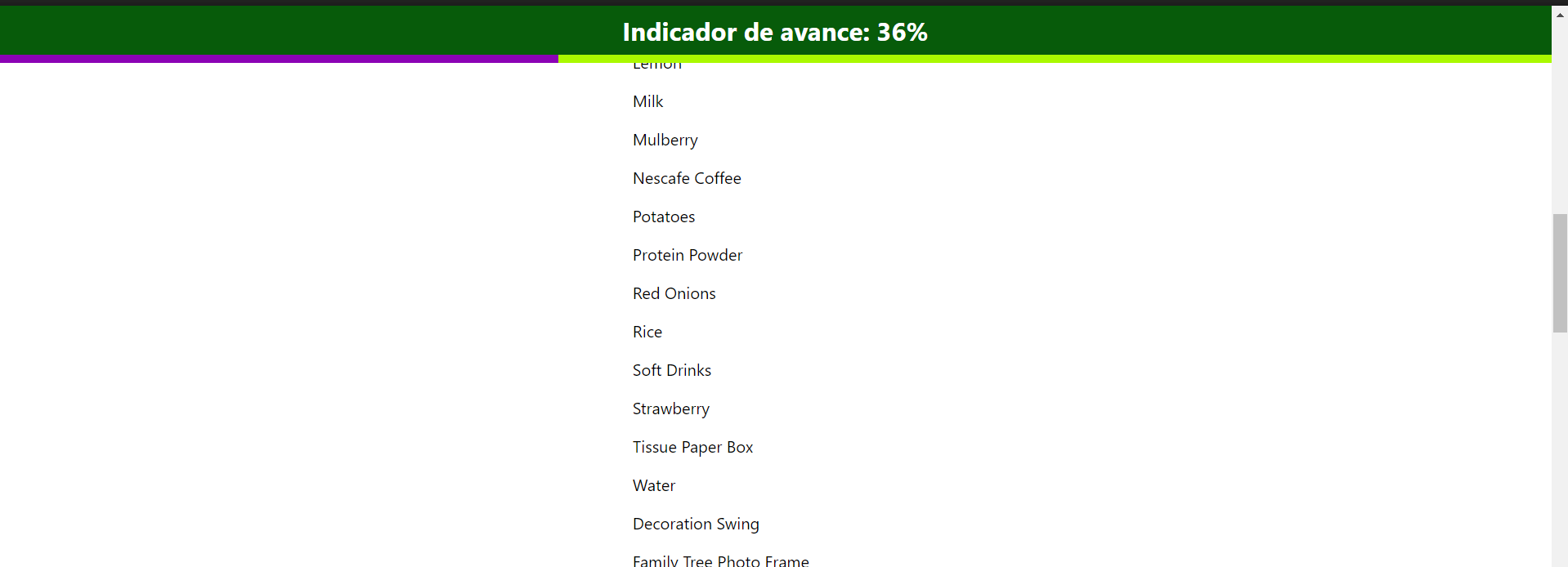
    font-size: 22px;

    font-weight: bold;

}

* Salida







* Regresando el scroll

