

Proyecto covid-19

Objetivo del proyecto

- Aprender a consumir apis En react
- Aprender a mostrar mapas y gráficas
- Afianzar ciertos conceptos básicos en react

El proyecto finalizado debería verse de esta manera, por supuesto que puedes agregarle las funcionalidades que quieras

Bien antes de comenzar es importante establecer un orden de pasos para no volvernos locos

1. Entender cuál es el objetivo del sitio.
2. Crear un bosquejo mínimo para guiarnos
3. Crear el proyecto
4. Crear la estructura de nuestro proyecto
5. Una vez creada la estructura, empezar por cada parte
6. Subir el proyecto

Bien empecemos entonces por los pasos

Paso 1

El paso 1 es entender qué es lo que queremos lograr y como lo vamos a hacer posible.

Lo que queremos es un sitio web que nos brinde información en tiempo real de los casos de Covid-19, tanto a nivel mundial como a niveles nacionales. Voy a escribir en ítems cada uno de los puntos así queda más ordenando entonces lo que queremos es:

- Poder elegir países a través de un select que me permita ver información relacionada del covid
- Mostrar la cantidad de casos de covid que existen hasta el momento a nivel mundial y por país.
- Mostrar la cantidad de fallecidos que existen hasta el momento a nivel mundial y por país
- Mostrar la cantidad de recuperados que existe \ hasta el momento a nivel mundial y por país
- Mostrar la cantidad de casos fallecidos, recuperados e infectados por día a nivel mundial y por país
- Mostrar una tabla que muestre el total de casos por país y ordenarlos de mayor a menor para poder visualizar los países con más casos
- Mostrar un gráfico donde se pueda ver a través de una porción de tiempo los nuevos casos de covid
- Mostrar un mapa interactivo que nos muestre a través de círculos los casos de covid que existen, tanto los infectados como los recuperados y los fallecidos, dependiendo lo que quiera ver
- Que el sitio web se vea estético y responsivo
- Poder subir el proyecto a un host

¡Perfecto! Ya definimos nuestros objetivos

Paso 2:

Crear un bosquejo de nuestro proyecto. No necesitas ser un diseñador o tener grandes conocimientos. Necesitas visualizar cómo te gustaría que se vea tu página al final del proyecto y tratar de diagramarlo. Hay varias herramientas para realizar un maquetado. Por ejemplo adobe XD o figma pero en este caso vamos a usar algo muy básico que es <https://app.diagrams.net/> es una masa para diagramar base de datos o objetos sencillos.

Entonces repasamos

- Necesitamos un título
- Un Select para los países
- Tres box que muestren cantidad de casos, recuperados y fallecidos

- Una tabla que muestre los casos por país
- Una gráfica que muestre la cantidad de de casos en una línea de tiempo
- Un mapa interactivo que nos muestre los casos

Si juntamos todos esos puntos nos quedaría un boceto muy sencillo de esta manera:



Paso 3:

Bueno ahora empieza lo interesante. Vamos a crear el proyecto y en este caso vamos a usar el framework react. ¿Porque utilizamos react y no otra herramienta? Bueno si bien este es un sitio web sencillo, consume apis en tiempo real en varios de nuestros componentes. Si nos fijamos los datos van a cambiar constantemente en la tabla en los boxes y en la grafica ademas cuando ponemos elegimos un país debe cambiar toda su información, si lo hacemos con otros frameworks como puede ser laravel, este debería renderizar la página cada vez que hacemos una petición lo cual sería bastante tedioso. Entonces react nos permite visualizar los cambios que se realicen sin renderizar toda la página de nuevo, solo actualizando el componente que haya sido actualizado.

Otra ventaja es que existe una amplia comunidad que usa react lo que te va a facilitar a la hora de encontrar información y resolver inconvenientes que tengas. No quiero hacer muy largo esto así que vamos a empezar por crear el proyecto

Considero que si estás leyendo esto y pretendes hacer este proyecto tienes un conocimiento de las bases de JS. Si no lo tienes te aconsejo que lo estudies porque básicamente react al fin y al cabo es JS. Los conocimientos básicos que para entender lo que vamos a hacer es, comprender bien la asincronía de JS, como consume una api con fetch, que son las promesas y los callbacks, que es una función y conocimientos sobre algunas funciones específicas, entender los eventos y entender cómo funcionan los iteradores.

Si no te acuerdas de esas cosas puedes darle una repasada.

Vas a necesitar tener instalado node.js, puedes descargarlo en este link <https://nodejs.org/en/> lo instalar dando siguiente a todo.

También vas a necesitar un IDE, en este caso el que recomiendo es VScode pero puede ser cualquier otro

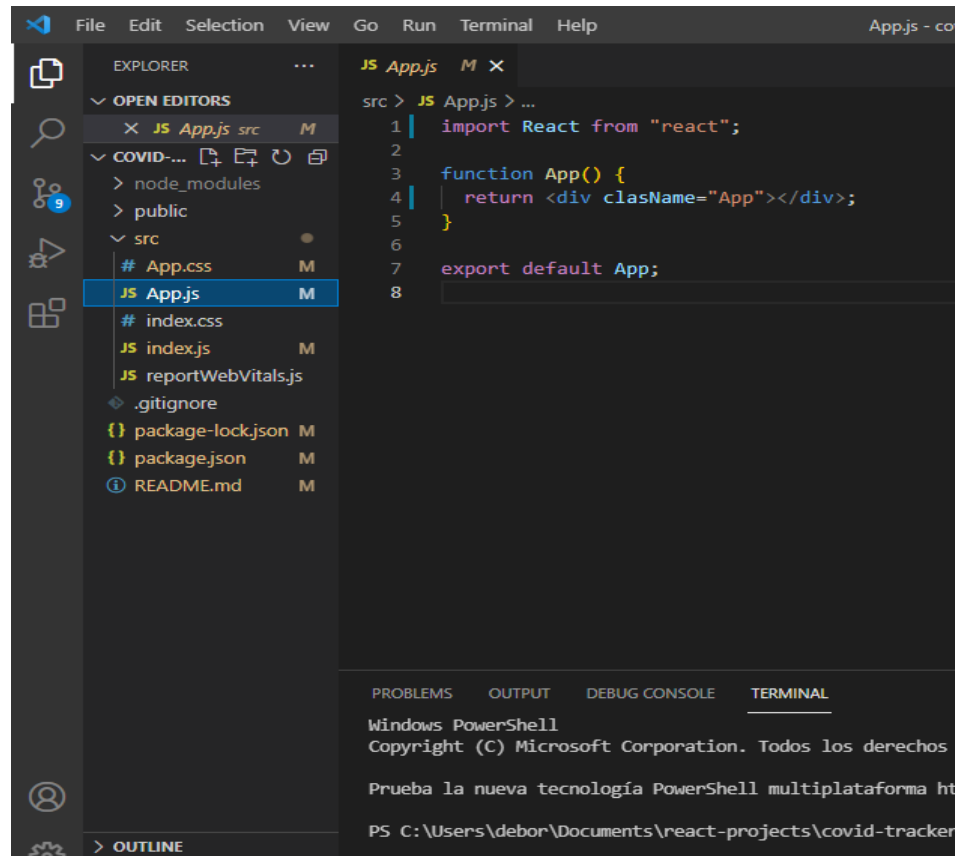
Bueno ahora si EMPECEMOS CON EL CÓDIGO DE UNA BUENA VEZ

- Para crear un nuevo proyecto abris la consola y te dirigis a la carpeta donde quieres que se guarde. Una vez ubicado en esa carpeta escribis lo siguiente:

```
npx create-react-app nombre-de-tu-app
```

Después si está todo bien puedes cd más el nombre de tu proyecto y listo ya estás adentro de tu proyecto

- Otra cosa que yo hago es limpiar el código porque sino se me arma un quilombo de archivos. Me queda masomenos algo así. Fijate cuando eliminas el logo o los css de eliminarlos del index en la parte de import sino te va a dar un error



- Otra cosa que vamos a usar es material UI. es un framework UI que nos permite generar interfaces más limpias y fáciles. No voy a entrar mucho en detalle, ya lo vas a ir comprobando a lo largo del proyecto.

Tenés que instalar las dependencias, entonces vas a la consola y poner lo siguiente:

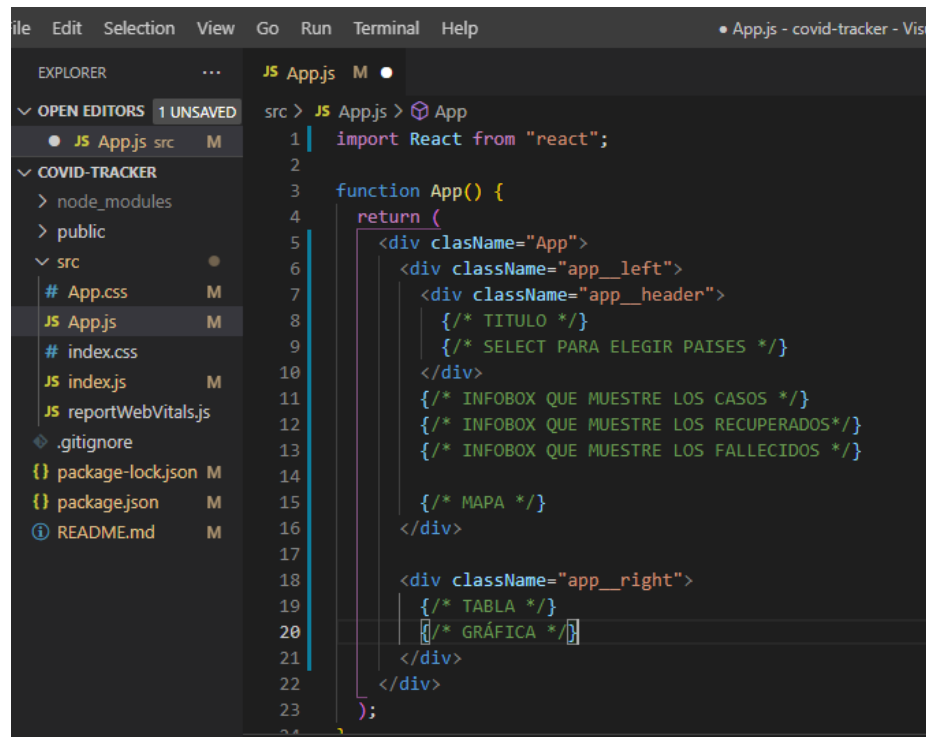
```
npm install @material-ui/core
```

Paso 4:

Bueno ya vamos 4 pasos y yo no escribí ni una sola línea de código pero bueno es mejor así sino después uno no entiende nada.

Ya entendimos lo que queremos hacer, ya hicimos un bello gráfico, ya instalamos las dependencias necesarias. Ahora vamos a organizar nuestro proyecto.

En el paso dos mencionamos todo lo que necesitábamos y lo pusimos en nuestro boceto. Ahora voy a hacer lo mismo pero en Visual Studio



```
1 | import React from "react";
2 |
3 | function App() {
4 |   return (
5 |     <div className="App">
6 |       <div className="app_left">
7 |         <div className="app_header">
8 |           {/* TITULO */}
9 |           {/* SELECT PARA ELEGIR PAISES */}
10 |        </div>
11 |        {/* INFOBOX QUE MUESTRE LOS CASOS */}
12 |        {/* INFOBOX QUE MUESTRE LOS RECUPERADOS*/}
13 |        {/* INFOBOX QUE MUESTRE LOS FALLECIDOS */}
14 |
15 |        {/* MAPA */}
16 |      </div>
17 |
18 |      <div className="app_right">
19 |        {/* TABLA */}
20 |        {/* GRÁFICA */}
21 |      </div>
22 |    </div>
23 |  );
24 | }
```

Paso 5

Ya tenemos todo ahora si a empezar a escribir código

Header:

- React siempre tiene que tener un componente padre que incluya a todos los componentes y a todos los divs, es decir no podemos por ejemplo tener dos divs separados que no están dentro de un div padre, por lo que nuestro div padre en este caso lo vamos a llamar

con el className=app y dentro vamos a poner dos divs, uno llamado app_left y otro app_right

- Dentro del app_left vamos a poner nuestro titulo

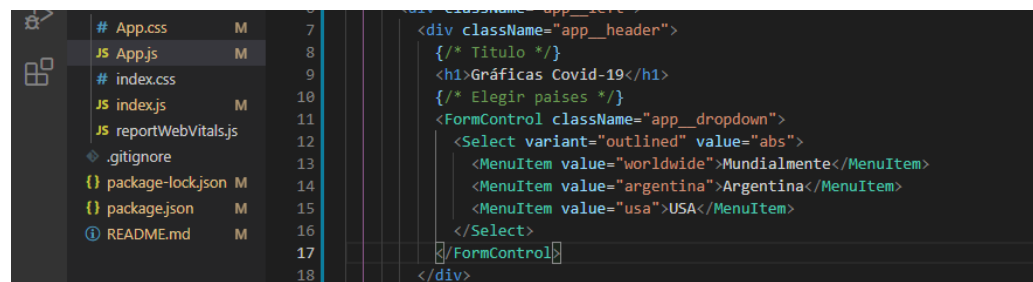
```
<div className="app__header">
  { /* Titulo */ }
  <h1>Gráficas Covid-19</h1>
  { /* Select para países */ }
</div>
```

- Ahora empecemos con lo interesante, el select para mostrar los países.

Primero voy a importar form Control y Selector de material ui de la siguiente manera

```
import { MenuItem, FormControl, Select } from
"@material-ui/core";
```

- Voy a crear un form control y dentro voy a poner un select y dentro del select voy a poner los items. Y nos quedaria algo asi

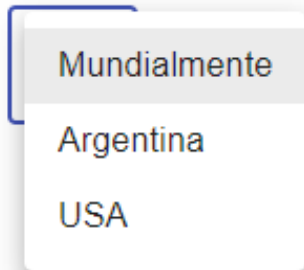


Si nosotros hacemos esto lo que vamos a ver es lo siguiente



Y si hacemos click vamos a ver nuestros items

Gráficas Covid-19



El tema es que es muy poco práctico llenar estos datos manualmente porque son como 180 países, ¡una locura! Entonces aquí viene la verdad de la milanesa y si entiendes esto literal entiendes todo porque después es hacer exactamente lo mismo en todos los pasos.

¡Vamos a consumir una API! Qué significa esto? Bueno Las APIs permiten que las aplicaciones se comuniquen y puedan aprovechar desarrollos ya contruidos en lugar de tener que crearlos desde cero. La arquitectura más usada es **REST** y el formato de envío de datos más usado es **JSON**. es decir nosotros vamos a tomar datos de otra aplicación para nuestra app.

Los datos lo vamos a sacar el siguiente sitio <https://disease.sh/v3/covid-19/countries>

Si entras a ese link te van a aparecer todos los países y cada país va a tener sus datos para consumir. Por ejemplo el primero es afganistán


```

▼ 0:
  updated: 1628791925225
  country: "Afghanistan"
  ▶ countryInfo: {}
  cases: 151770
  todayCases: 207
  deaths: 7000
  todayDeaths: 12
  recovered: 105490
  todayRecovered: 462
  active: 39280
  critical: 1124
  casesPerOneMillion: 3804
  deathsPerOneMillion: 175
  tests: 738599
  testsPerOneMillion: 18512
  population: 39897899
  continent: "Asia"
  oneCasePerPeople: 263
  oneDeathPerPeople: 5700
  oneTestPerPeople: 54
  activePerOneMillion: 984.51
  recoveredPerOneMillion: 2644
  criticalPerOneMillion: 28.17
  ▶ 1: {}
  ▶ 2: {}
  ▶ 3: {}

```

Y esto es un objeto con un montón de datos que podemos usar tranquilamente para nuestra página. Bien ahora ¿cómo traemos esta info y la usamos en nuestra web? Bueno con la gran ayuda de useState y useEffect que nos proporciona react. useState es una propiedad de react hook que nos permite que cada vez que guardemos algo dentro de él cambie, se renderice de nuevo el componente.

Voy a poner un ejemplo para que se entienda que es lo que hace useState en este caso.

Primero importo useState para poder usarlo

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

Luego voy a generar una variable, Los states al fin y al cabo son una manera de introducir las variables en react.

```
const [countries, setCountries] = useState(["USA", "INDIA", "UK"]);
```

Después voy a ir a mi select y voy a abrir unas llaves donde voy a recorrer los valores de mi array principal countries con la función array map y voy a retornar item menu que van a tener como valores lo que tiene cada una de las posiciones del array

```
<FormControl className="app__dropdown">
  <Select variant="outlined" value="abs">
    {countries.map((country) => (
      <MenuItem value={country}>{country}</MenuItem>
    ))}
  </Select>
</FormControl>
```

Si escribimos todo eso deberíamos tener esto en la pantalla



Ahora entendemos como funciona useState pero si pudieramos acceder a esa api que contenia los paises seria un golazo! Bueno para eso usamos useEffect en este caso lo que va incorporar la información de los objetos que vamos a utilizar.

Lo que vamos a hacer es que dentro de useEffect va a estar la pieza de código que basada en una condicion dada. Para esto vamos a utilizar async y await. No voy a explicar mucho esto pero en pocas palabras

significa que async va a enviar una petición, va a esperar y va a hacer algo con esa info. Bueno como se escribe todo esto?

Primero importamos useEffect

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

Después escribimos esto

```
useEffect(() => {  
  const getCountriesData = async () => {  
    await fetch("https://disease.sh/v3/covid-19/countries")  
      .then((response) => response.json())  
      .then((data) => {  
        const countries = data.map((country) => ({  
          name: country.country,  
          value: country.countryInfo.iso2,  
        }));  
        setCountries(countries);  
      });  
  };  
  getCountriesData();  
}, []);
```

Parece un bardo pero si logras entender esto, el resto es exactamente lo mismo

Voy a crear una función llamada `getCountriesData` que va a ser asíncrona. Por eso uso el `async`, uso el `await fetch` que nos sirve para hacer el llamado a la url y eso nos va a devolver un json y esto lo guardo en mi constante `data`. Entonces ahora recorro esta `data` y cada objeto del mismo lo llamo `country`. Luego pongo lo que voy a consumir, en este caso `name` va a ser igual al valor dentro de mi objeto con la clave `country` y `value` vamos a tener que entrar al objeto `country` que dentro tiene un objeto llamado `countryInfo` y el mismo tiene una clave llamada `iso2` con un valor ahora lo vamos a renderizar con `setCountries` poniendo la información que obtuvimos de la api y luego retornamos la función que creamos.

A lo último del `useEffect` tenemos que poner `[]` para no generar un loop infinito.

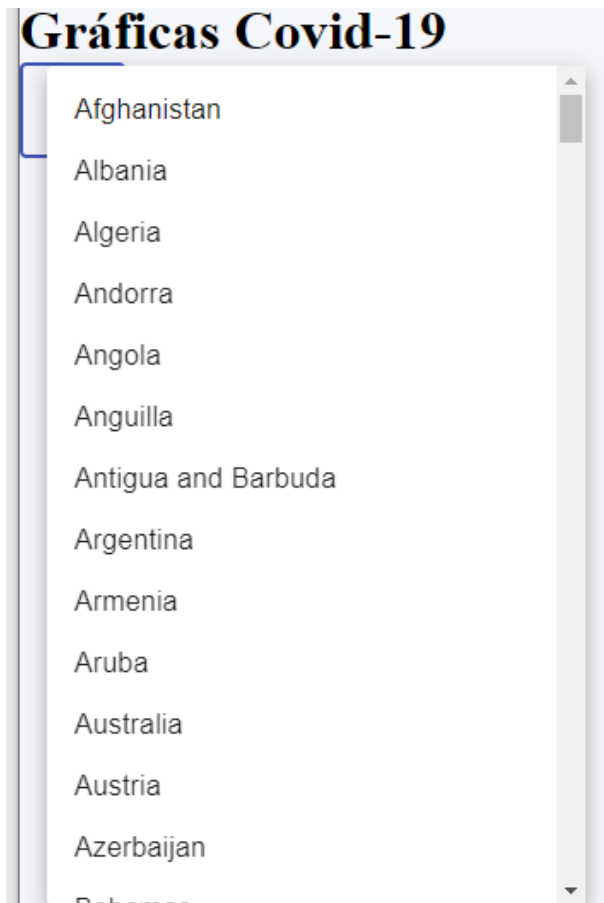
Ahora solo nos queda pasar lo que obtuvimos en nuestro `ItemMenu`

```

<FormControl className="app__dropdown">
  <Select variant="outlined" value="abs">
    {countries.map((country) => (
      <MenuItem value={country.value}>{country.name}</MenuItem>
    ))}
  </Select>
</FormControl>

```

Si todo sale bien debemos obtener lo siguiente



Wiiiiiiiiiiiiiiiiii funciono!! Ahora le voy a dar un par de estilos para que quede más lindo.

En la carpeta App.css hago lo siguiente

```

.app__header {
  display: flex;
  justify-content: space-between;

```

```
align-items: center;
margin-bottom: 20px;
}
```

No voy a escribir mucho sobre esto pero lo que digo es que el contenedor padre que tiene dentro a nuestro select y a nuestro título va a ser flexible y le digo que me justifique el contenido con space-between le digo que tenga una alineación centrada y un margen de 20px.

Si quieres saber más sobre flexbox puedes entrar aca <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Ahora se ve así:

Gráficas Covid-19

- Afghanistan
- Albania
- Algeria
- Andorra

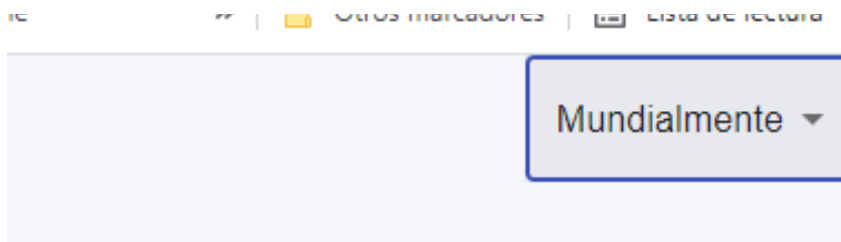
El primero de los valores no va a entrar en el loop. Va a tener un valor fijo entonces creamos un estado con ese valor

```
const [country, setCountry] = useState("worldwide");
```

Luego pasamos ese valor al select

```
<Select variant="outlined" value={country}>
  <MenuItem value="worldwide">Mundialmente</MenuItem>
```

Y obtenemos esto



Si te encontrás en este punto vas a notar que si quieres cambiar el valor de mundialmente por un país que elijas, no va a pasar nada por lo que tenemos que agregar una funcionalidad.

Vamos a usar el evento onChange que va a estar pendiente de los cambios de nuestro input y le pasamos la función asíncrona llamada onCountryChange que recupera la entrada del valor que se solicitó. Entonces por ejemplo hago click en argentina entonces se activa esta función y recupera el valor que en este caso viene de country.country que en este caso es "argentina" ahora si tenemos esta función y esta constante que nos recupera este valor lo podemos pasar en setCountry para que este se encarga de renderizar cada vez que uno cambie el valor. La función quedaría así

```
const onCountryChange = async (event) => {
  const countryCode = event.target.value;
  setCountry(countryCode);
};
```

Y en select

```
<Select
  variant="outlined"
  value={country}
  onChange={onCountryChange}
/>
```

InfoBox:

- Bueno quiero un infobox que me traiga todos los datos de los fallecidos, recuperados y infectados tanto a nivel mundial como a nivel país, por día y el total hasta el día de hoy
 - Para tener todo el código más ordenado voy a crear el componente llamado infobox.js y le voy a pasar tres parámetros, el título, la cantidad de casos y el total
- Solo por una cuestión estética voy a importar de material ui etiquetas como card, card content y typography. También voy a crear un archivo llamado InfoBox.css donde voy a copiar todos los estilos.
- Debería quedar algo así

```
import React from "react"
import { Card, CardContent, Typography } from "@material-ui/core";
import "./InfoBox.css";

function InfoBox({ title, cases, total }) {
```

```

return (
  <Card>
    <CardContent>
      <Typography> {title}</Typography>
      <Typography className="infoBox__title">{total}</Typography>
      <Typography className="infoBox__title">{cases}</Typography>
    </CardContent>
  </Card>
);
}

export default InfoBox;

```

- Ahora en apps voy a escribir dos use Effect, uno para cuando el usuario elige la opción mundialmente y otra para cuando elige un país
- Entonces voy con la receta de nuevo:

- Crear un useState);

```
const [countryInfo, setCountryInfo] = useState({});
```

- Crear el useEffect dentro de onCountryChange:

```

const onCountryChange = async (event) => {
  const countryCode = event.target.value;
  setCountry(countryCode);

  const url =
    countryCode === "worldwide"
      ? "https://disease.sh/v3/covid-19/all"
      :
`https://disease.sh/v3/covid-19/countries/${countryCode}`;

  await fetch(url)
    .then((response) => response.json())
    .then((data) => {
      setCountry(countryCode);
      setCountryInfo(data);
    });
};

```

Entonces sanalicemos el código de arriba, nosotros habíamos creado una función que recuperaba el valor elegido en el select y renderizaba nuestro box de select, bien ahora nosotros podemos darle un uso interesante a ese valor almacenado en countryCode. Lo que podemos hacer es decirle es “si el usuario

elige 'mundialmente', mostrarte los datos mundiales, pero si elige un país, mostrarle los datos de ese país. para eso usamos dos apis. Una que nos trae los datos de todo y otra que nos trae los datos de cada país.

Creamos un componente llamado url y ponemos una condición. Fíjate que en el segundo valor de la condición usamos comillas invertidas `` esto lo usamos para concatenar strings de manera más optimizada.

Una vez que el componente ya tiene la dirección de su api con su país correspondiente o si es a nivel mundial hacemos lo mismo que habíamos hecho antes. Traemos la info, la convertimos y devolvemos la data.

para que todo funcione de manera correcta debemos hacer un useEffect que traiga los datos a nivel mundial

```
useEffect(() => {
  fetch("https://disease.sh/v3/covid-19/all")
    .then((response) => response.json())
    .then((data) => {
      setCountryInfo(data);
    });
}, []);
```

```
<InfoBox
  className="infoBox__cases"
  title="Casos diarios De Covid"
  total={countryInfo.cases}
  cases={countryInfo.todayCases}
/>
<InfoBox
  className="infoBox__recovered"
  title="Recuperados"
  total={countryInfo.recovered}
  cases={countryInfo.todayRecovered}
/>
<InfoBox
  className="infoBox__deaths"
  title="Fallecimientos"
  total={countryInfo.deaths}
  cases={countryInfo.todayDeaths}
/>
```


Si todo sale bien deberíamos ver esto



Table:

- Ya tenemos el header y el infobox medio terminados, vamos con la parte derecha de nuestra app, que era una tabla y las gráficas.
- Primero voy a crear un nuevo archivo llamado Table.js y uno llamado Table.css
- En este archivo voy a tratar de mostrar los países y los respectivos casos, entonces voy a necesitar dos parámetros , country y cases. Por lo que voy a recorrer un array llamado countries y con la función map voy a devolver los datos que necesito .
- Voy a poner el código y lo voy a ir explicando

```
import React from "react";
import "../Table.css";
import numeral from "numeral";

function Table({ countries }) {
  return (
    <div className="table">
      {countries.map(({ country, cases }) => (
        <tr>
          <td>{country}</td>
          <td>
            <strong>{numeral(cases).format("000,000")}</strong>
          </td>
        </tr>
      ))}
    </div>
  );
}

export default Table;
```

En App creo el useState

```
const [tableData, setTableData] = useState([]);
```

Y en el card de right :

```
<Table countries={tableData} />
```

Bien le estamos diciendo a la tabla que countries se le asigna los valores de tableData , este table data viene del useState que en un principio tiene como valor un array vacío y se renderiza con setTableData, y este setTableData donde va? Bueno si nosotros analizamos de dónde queremos que venga la info , esta viene de getCountriesData que es la que tiene almacenada toda la información de los países y lo traemos y los transformamos en data para usar, entonces debajo en ese useEffect del cual sacamos los datos, ponemos nuestro setTable data

```
useEffect(() => {
  const getCountriesData = async () => {
    await fetch("https://disease.sh/v3/covid-19/countries")
      .then((response) => response.json())
      .then((data) => {
        const countries = data.map((country) => ({
          name: country.country,
          value: country.countryInfo.iso2,
        }));
        setCountries(countries);
        setTableData(data);
      });
  };
  getCountriesData();
}, []);
```



Ahora y si yo quiero ordenar y que me muestre primero los países con más casos? Una manera que podríamos usar es crear un archivo que se encargue de estas cuestiones secundarias. Entonces creamos un archivo util.js y ponemos lo siguiente

```
import React from "react";

export const sortData = (data) => {
  const sortedData = [...data];

  sortedData.sort((a, b) => b.cases - a.cases);

  return sortedData;
};
```

Lo que hicimos fue crear un componente llamado SortData con la propiedad data. Dentro de este componente vamos a traer los datos del objeto obtenido en data y a través del método sort, vamos a generar una matriz que ordene de manera descendente el orden de los casos.

Ahora hay que hacer que este componente se ponga en acción. Para eso vamos a app.js

Importamos el componente

```
import { sortData } from "../util";
```

Y luego lo utilizamos en nuestro componente de tabla de la siguiente manera cambiamos

```
setTableData(data);
```

Por

```
const sortedData = sortData(data);
setTableData(sortedData);
```

Si copian todos los estilos de Table.css que se encuentra en este repositorio les quedaría algo de este estilo

- liamente ▾
- ecimientos
- 8665
- i3

Casos por País

USA	37,906,794
India	32,285,101
Brazil	20,417,204
Russia	6,663,473
France	6,504,978
UK	6,322,241
Turkey	6,118,508
Argentina	5,096,443
Colombia	4,874,169
Spain	4,733,602
Iran	4,556,417

LineGraph:

- Bueno en LineGraph la idea es usar las bondades de Chart.js, no voy a especificar cómo funciona chart.js, pero si no conoces lo que tenes que saber es que nos sirve para realizar gráficos.
- Vamos a instalar la dependencia
npm reacciono-chartjs-2 chart.js
- Luego vamos a crear una carpeta llamada LineGraph.js

```
import React, { useState, useEffect } from "react";
import { Line } from "react-chartjs-2";
import numeral from "numeral";

const options = {
  legend: {
    display: false,
  },
  elements: {
    point: {
      radius: 0,
    },
  },
  maintainAspectRatio: false,
  tooltips: {
    mode: "index",
    intersect: false,
```

```

callbacks: {
  label: function (tooltipItem, data) {
    return numeral(tooltipItem.value).format("+0,0");
  },
},
},
scales: {
  xAxes: [
    {
      type: "time",
      time: {
        format: "MM/DD/YY",
        tooltipFormat: "ll",
      },
    },
  ],
  yAxes: [
    {
      gridLines: {
        display: false,
      },
      ticks: {
        callback: function (value, index, values) {
          return numeral(value).format("0a");
        },
      },
    },
  ],
},
};

```

```

const buildChartData = (data, casesType = "cases") => {
  let chartData = [];
  let lastDataPoint;
  for (let date in data.cases) {
    if (lastDataPoint) {
      let newDataPoint = {
        x: date,
        y: data[casesType][date] - lastDataPoint,
      };
    }
  }
};

```

```

    };
    chartData.push(newDataPoint);
  }
  lastDataPoint = data[casesType][date];
}
return chartData;
};

function LineGraph() {
  const [data, setData] = useState({});

  useEffect(() => {
    const fetchData = async () => {
      await
fetch("https://disease.sh/v3/covid-19/historical/all?lastdays=120")
      .then((response) => {
        return response.json();
      })
      .then((data) => {
        let chartData = buildChartData(data, "cases");
        setData(chartData);
        console.log(chartData);
        // buildChart(chartData);
      });
    };

    fetchData();
  }, []);

  return (
    <div>
      {data?.length > 0 && (
        <Line
          data={{
            datasets: [
              {
                backgroundColor: "rgba(204, 16, 52, 0.5)",
                borderColor: "#CC1034",
                data: data,
              },
            ],
          }}
        />
      )}
    </div>
  );
}

```

```

    ],
  })
  options={options}
/>
  )}
</div>
);
}

export default LineGraph;

```

- En LineGraph estamos usando react-chartjs-2 para crear el gráfico. Necesitamos pasar algún valor en Line que se importa de 'react-chartjs-2',
- Así que estamos definiendo opciones, en realidad define los detalles que se necesitan para crear el gráfico de líneas, como mostrar información sobre herramientas, detalles de escalas para los ejes xey, etc. Así que en la opción estamos configurando la opción de visualización, datos de información sobre herramientas, datos de escala. El eje X tendrá información de fecha y el eje Y tendrá el número de casos.
- En la función BuildChartData estamos enviando datos al eje x y, BuildChartData tiene dos parámetros data y casesType. Entonces, estamos creando los datos del gráfico para LineGraph. Entonces, lo que estamos haciendo es iterar a través de los casos de datos, estamos creando una matriz de objetos que tienen el nombre del eje y los datos.
- Luego creamos un objeto newDataPoint enviando la fecha en x, estamos restando los datos de la fecha anterior con los datos de la fecha actual y enviándolos al eje y, para que nuestro eje y pueda reflejar la diferencia entre el total de casos fecha por fecha. Luego estamos pusheando nuestro objeto newDataPoint a la matriz vacía chartData, configuramos el punto de datos como lastDataPoint para la siguiente iteración.
- Finalmente devolvemos ese chartData. En LineGraph queremos que los datos de nuestro gráfico cambien cada vez que cambie el tipo de caso, para eso estamos usando ganchos useEffect, dentro de eso estamos obteniendo datos de la api disease.sh para obtener los datos de los últimos 120 días, cuando obtenemos los datos que estamos convertirlo a json y pasarlo a nuestra función BuildChartData, para que obtengamos un formato adecuado de datos que se pueda usar para crear la representación gráfica. Ahora veamos la sección de renderizado, allí estamos verificando si data.length > 0, significa tenemos datos para representar en Graph, estamos renderizando el componente Line. Estamos pasando pocos parámetros en Line como datos, los datos son un objeto que tiene dos conjuntos de datos de parámetros y opciones. En el conjunto de datos, estamos enviando una matriz que tiene un objeto con backgroundColor, borderColor y datos que preparamos en la función BuildChartData. En

opciones estamos pasando la variable de opciones que definimos en la parte superior, así que ahora nuestra aplicación está lista. Luego pasamos el componente en app.js

Si todo sale bien deberíamos ver así:



Map.js

- Este sería el último de los componentes de esta app.
- Vamos a usar la dependencia
npm install react-leaflet
Estamos usando leaflet para implementar el mapa en nuestra aplicación.
- Vamos a crear un componente llamado map.js y un archivo llamado Map.css

```
import React from "react";
import { MapContainer, TileLayer, useMap } from "react-leaflet";
import "./Map.css";
import { showDataOnMap } from "../util";

function Map({ countries, casesType, center, zoom }) {
  function ChangeView({ center, zoom }) {
    const map = useMap();
    map.setView(center, zoom);
  }
}
```



```

    return null;
  }

  return (
    <MapContainer
      casesType={casesType}
      className="map"
      center={center}
      zoom={zoom}
      scrollWheelZoom={false}
    >
      <ChangeView center={center} zoom={zoom} />
      <TileLayer
        attribution='&copy; <a
href="http://osm.org/copyright">OpenStreetMap</a> contributors'
        url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
      />
      {showDataOnMap(countries, casesType)}
    </MapContainer>
  );
}

export default Map;

```

- Importamos LeafletMap, TileLayer de 'react-leaflet'
- En el componente Map estamos pasando props como casesType para obtener la lista de países que se mostrarán en el mapa, zoom y center.
- ShowDataOnMap es una función que se llama en la función Map, tiene dos parámetros, data y casesType. Cuando hacemos clic en el país en particular en el mapa, obtenemos una ventana emergente con la bandera del país, el nombre del país y el recuento total de casos, recuperaciones y muertes. Estamos resaltando el país con un círculo para eso enviamos el valor de latitud y longitud del país seleccionado, fillOpacity a 0.4, color según el color hexadecimal de casesType y radio según el valor de casesType y país. Esto lo importamos de util.js

```

export const showDataOnMap = (data, casesType) => {
  data.map((country) => {
    <Circle
      center={[country.countryInfo.lat, country.countryInfo.long]}
      fillOpacity={0.4}
      pathOptions={{

```

```

        color: casesTypeColors[casesType].hex,
        fillColor: casesTypeColors[casesType].hex,
    }}
    radius={
        Math.sqrt(country[casesType] / 10) *
        casesTypeColors[casesType].multiplier
    }
    >
    <Popup>
      <div className="info-container">
        <div
          className="info-flag"
          style={{ backgroundImage:
            `url(${country.countryInfo.flag})` }}
        />
        <div className="info-name">{country.country}</div>
        <div className="info-confirmed">
          Casos: {numeral(country.cases).format("0,0")}
        </div>
        <div className="info-recovered">
          Recuperados: {numeral(country.recovered).format("0,0")}
        </div>
        <div className="info-deaths">
          Fallecidos: {numeral(country.deaths).format("0,0")}
        </div>
      </div>
    </Popup>
  </Circle>
  ));

```

En app.js pasamos los useStates. MapCenter será el objeto que tenga el valor de latitud y longitud del país seleccionado. mapZoom se utiliza para acercar y alejar el mapa, mapCountries es una matriz que tiene la lista de países que estarán presentes en nuestro mapa

```

const [mapCenter, setMapCenter] = useState([34.80746, -40.4796]);
const [zoom, setZoom] = useState(3);
const [mapCountries, setMapCountries] = useState([]);

```

En const url tambien ponemos un if para que sepa que mostrar en el caso de elegir la opcion mundial o por pais.

```
countryCode === "worldwide"
  ? setMapCenter([34.80746, -40.4796])
  : setMapCenter([data.countryInfo.lat,
data.countryInfo.long]);
setZoom(4);
```

Lo que hace que tod este enganchado es casesTypes. Ahora voy a poner como queda el App.js con todo lo que fuimos agregando y un par de cosas más para mejorar la estetica

```
import React, { useEffect, useState } from "react";
import "./App.css";
import {
  MenuItem,
  FormControl,
  Select,
  CardContent,
  Card,
} from "@material-ui/core";
import InfoBox from "./InfoBox";
import Map from "./Map";
import Table from "./Table";
import { sortData, prettyPrintStat } from "./util";
import LineGraph from "./LineGraph";
import "leaflet/dist/leaflet.css";
import "./InfoBox.css";

function App() {
  const [countries, setCountries] = useState([]);
  const [country, setCountry] = useState("worldwide");
  const [countryInfo, setCountryInfo] = useState({});
  const [tableData, setTableData] = useState([]);
  const [mapCenter, setMapCenter] = useState([34.80746, -40.4796]);
  const [zoom, setZoom] = useState(3);
  const [mapCountries, setMapCountries] = useState([]);
  const [casesType, setCasesType] = useState("cases");
  const [isLoading, setLoading] = useState(false);

  useEffect(() => {
```

```

    fetch("https://disease.sh/v3/covid-19/all")
      .then((response) => response.json())
      .then((data) => {
        setCountryInfo(data);
      });
  }, []);

useEffect(() => {
  const getCountriesData = async () => {
    await fetch("https://disease.sh/v3/covid-19/countries")
      .then((response) => response.json())
      .then((data) => {
        const countries = data.map((country) => ({
          name: country.country,
          value: country.countryInfo.iso2,
        }));
        const sortedData = sortData(data);
        setTableData(sortedData);
        setMapCountries(data);
        setCountries(countries);
      });
  };

  getCountriesData();
}, []);

const onCountryChange = async (event) => {
  setLoading(true);
  const countryCode = event.target.value;

  setCountry(countryCode);

  const url =
    countryCode === "worldwide"
      ? "https://disease.sh/v3/covid-19/all"
      : `https://disease.sh/v3/covid-19/countries/${countryCode}`;

  await fetch(url)
    .then((response) => response.json())
    .then((data) => {

```

```

        setCountry(countryCode);
        setCountryInfo(data);
        setLoading(false);
        countryCode === "worldwide"
            ? setMapCenter([34.80746, -40.4796])
            : setMapCenter([data.countryInfo.lat,
data.countryInfo.long]);
        setZoom(4);
    });

    console.log(countryInfo);
};

return (
    <div className="app">
        <div className="app__left">
            <div className="app__header">
                <h1>Gráficas Covid-19</h1>
                <FormControl className="app__dropdown">
                    <Select
                        variant="outlined"
                        onChange={onCountryChange}
                        value={country}
                    >
                        <MenuItem value="worldwide">Mundialmente</MenuItem>
                        {countries.map((country) => (
                            <MenuItem
value={country.value}>{country.name}</MenuItem>
                        ))}
                    </Select>
                </FormControl>
            </div>

            <div className="app__stats">
                <InfoBox
                    isRed
                    active={casesType === "cases"}
                    className="infoBox__cases"
                    onClick={(e) => setCasesType("cases")}
                    title="Casos diarios De Covid"
                >

```

```

        total={prettyPrintStat(countryInfo.cases)}
        cases={prettyPrintStat(countryInfo.todayCases)}
        isloading={isLoading}
      />
    <InfoBox
      active={casesType === "recovered"}
      className="infoBox__recovered"
      onClick={ (e) => setCasesType("recovered")}
      title="Recuperados"
      total={prettyPrintStat(countryInfo.recovered)}
      cases={prettyPrintStat(countryInfo.todayRecovered)}
      isloading={isLoading}
    />
    <InfoBox
      isGrey
      active={casesType === "deaths"}
      className="infoBox__deaths"
      onClick={ (e) => setCasesType("deaths")}
      title="Fallecimientos"
      total={prettyPrintStat(countryInfo.deaths)}
      cases={prettyPrintStat(countryInfo.todayDeaths)}
      isloading={isLoading}
    />
  </div>

  <Map
    countries={mapCountries}
    center={mapCenter}
    zoom={zoom}
    casesType={casesType}
  />
</div>
<Card className="app__right">
  <CardContent>
    <h3>Casos por País</h3>
    <Table countries={tableData} />
    <h3 className="app__graphTitle">Gráfica de Casos</h3>
    <LineGraph className="app__graph" casesType={casesType} />
  </CardContent>
</Card>

```

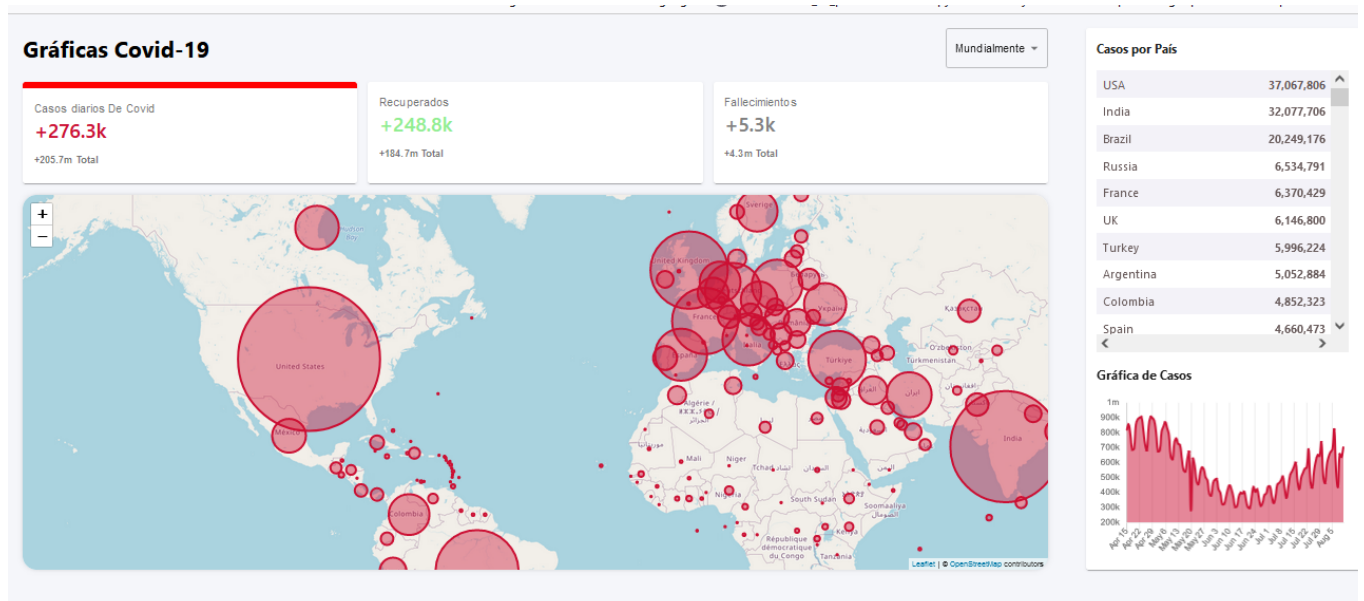
```

</div>

);
}

export default App;

```



Por último usamos Firebase para subir el proyecto. Tienes que tener una cuenta de gmail.

Te dejo un video que vi en youtube que es bastante explicativo

https://www.youtube.com/watch?v=QNMVURtx86c&t=250s&ab_channel=ProCodeTv

Saludos!