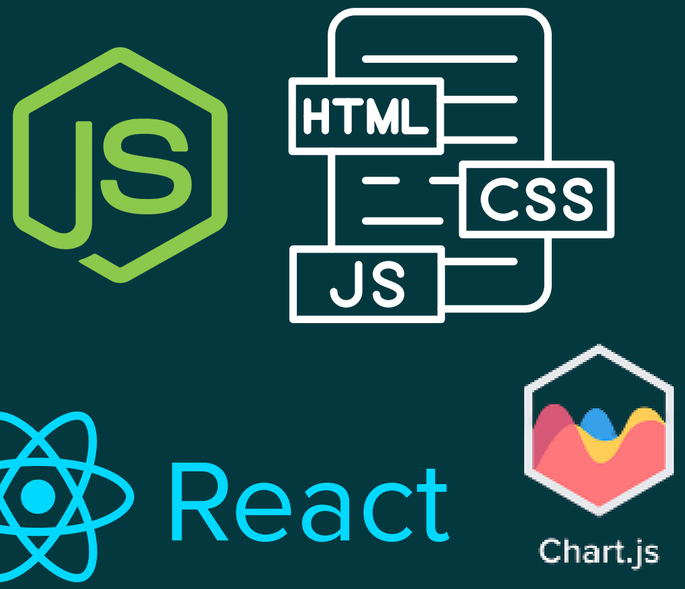
Crypto Stadistics

Angela Diaz
Andres Gonzalez
Alejandro Sotelo
Brayan Duarte
Camila Forero
Dylan Ramirez
Jesus Ocampo
Nicolas Naranjo

RECURSOS

Node.js : **npx create-react-app**
Jest : **npm install --save-dev jest**
Chart.js : **npm install chart.js**
react chart.js : **npm add react-chartjs-2 chart.js**
React icons : **npm i react-icons**
Moment : **npm install moment**



Componentes

- App
- Header
- Card Principal
- Card
- Convert
- InputConvert
- TableCoins
- CoinRow
- Graph
- Footer

<App/>

Este componente es el encargado de renderizar toda la aplicación.
Aquí se hace la importación de los componentes.

Se crearán los siguientes estados:

```
const [coins, setCoins] = useState()
const [currency, setCurrency] = useState()
const [selCur, setSelCur] = useState("usd")
```

Se crearán 2 hooks, uno que se ejecuta una vez se renderiza el componente y que consuma el api, y otro que consuma el api al cambiar el estado selCur

CONSUMO DE LA API

Vamos a consumir la API de Coingecko para traer la información de las criptomonedas con mayor capitalización del mercado.

En este componente se consumen los siguientes links:

Link de información principal de la app:

https://api.coingecko.com/api/v3/coins/markets?vs_currency=usd&order=market_cap_desc&per_page=4&page=1&sparkline=false&price_change_percentage=1h%2C24h%2C7d%2C30d%2C90d%2C1y

Link de la API que cambiará la información dependiendo de la moneda que elijamos:

https://api.coingecko.com/api/v3/simple/supported_vs_currencies

Ejemplo consumo api:

```
const getData = async () =>{
const response = await fetch(`https://unUrl)
const json = await response.json()
setCoins(json)
```

<CardPrincipal/>

Las props que utilizaremos para nuestro componente <CardPrincipal />

```
{ json: { id,
    symbol,
    current_price,
    image,
    price_change_percentage_1h_in_currency,
    price_change_percentage_24h_in_currency,
    price_change_percentage_7d_in_currency,
    price_change_percentage_30d_in_currency,
    price_change_percentage_1y_in_currency}, cur = "usd" }

className={colorDec(price_change_percentage_1h_in_currency)}
deleteDec(price_change_percentage_7d_in_currency, 2)
```

Resultado previsto:

<Card/>

Resultado previsto:

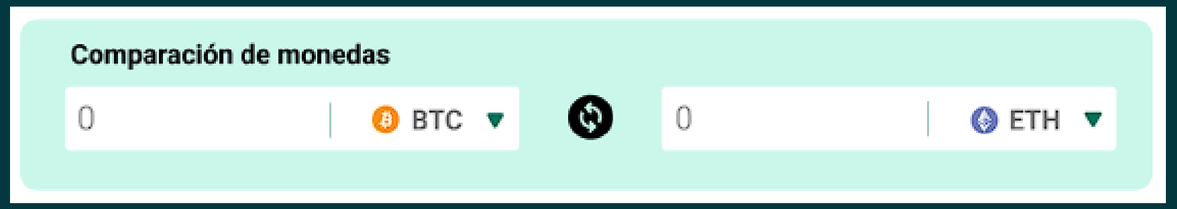
Los props utilizados para este componente son:

{coinld, cur, porcentaje, price, img}

<Graph coin={coinld} currency={cur}
color={colorDec(porcentaje)}/>

<InputConvert/>

Resultado previsto:



Los estados utilizados para este componente son:

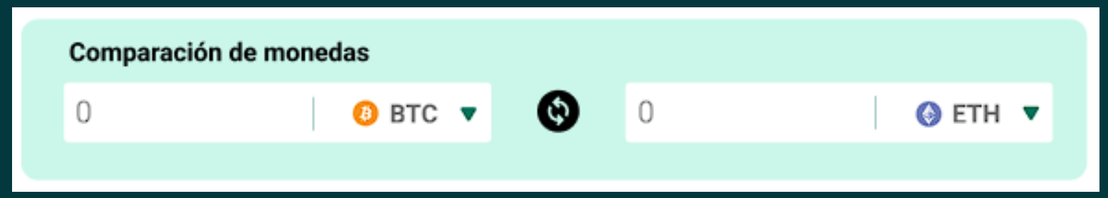
```
const [coin, setCoin] = useState([])
const [selCoin1, setSelCoin1] = useState("btc")
const [selCoin2, setSelCoin2] = useState("eth")
const [mainTxt, setMainTxt] = useState(0)
const [res, setRes] = useState(0)
```

Utilizaremos el siguiente Hook al cambiar los estados [mainTxt,selCoin1,selCoin2])

```
let a,b
  coin.forEach(({symbol, current_price}) =>{
   if(symbol == selCoin1){
    a = (mainTxt * current_price) / 1
   }else if(symbol == selCoin2){
    b = current_price
   }
  })
   a ? setRes(a / b) : setRes(0)
 }
```

<Convert/>

Resultado previsto:



Los hooks utilizados para este componente son:

```
const selRef = useRef(null)
const [selVal, setSelVal] = useState(sel)
```

Las props utilizados para este componente son:

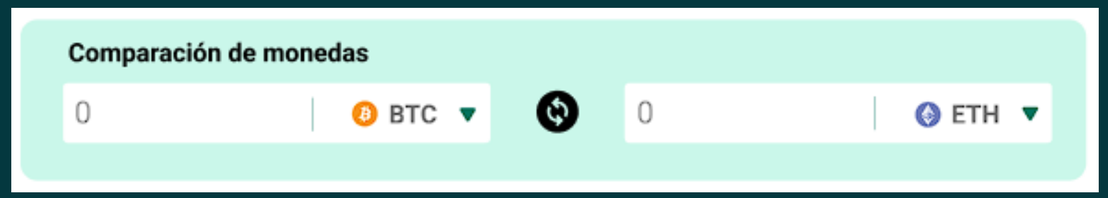
```
{ coin,  sel = "btc", fun, other,text, type = 1, result = 0}
```

Se hace uso de ternario

```
(type == 0) ? <input type="number" placeholder="0" onChange={e => {text(parseInt(e.target.value))}}/>
            : <input type="number" placeholder="0" value= {deleteDec(result, 4)} readOnly={true}/>
```

<Convert/>

Resultado previsto:



Llenado del select

```
<select value={selVal} ref={selRef} onChange={() => {
        setSelVal(selRef.current.value)
        fun(selRef.current.value)
      }}>
```

Llenado de las opciones del select

```
coin.map((co) => {
        if(co.symbol === selVal){
            selRef.current.previousSibling.src
= co.image
            return <option value={co.symbol}
key={co.id}>{co.symbol}</option>
        }else if(co.symbol != other){
            return <option value={co.symbol}
key={co.id}>{co.name}</option>
        }
      })
```

<TableCoins/>

Resultado previsto:

#	Moneda	Precio	24h	Vol. en 24h	Cap. mercado	Últimos 7 días
1		16.490,89 US$	1.8%	26.339.191.989 US$	316.490.116.965 US$	
2		1172,65 US$	3.7%	9.028.236.107 US$	140.560.206.984 US$	
3		12,75 US$	4.3%	191.846.806 US$	3.821.754.983 US$	

Se reciben las siguientes props

{ coins }

Y se llena cada fila con el componente Coinrow

```
coins.map((coin, index) => (
        <CoinRow coin={coin} key={index} index={index + 1} />
    ))
```

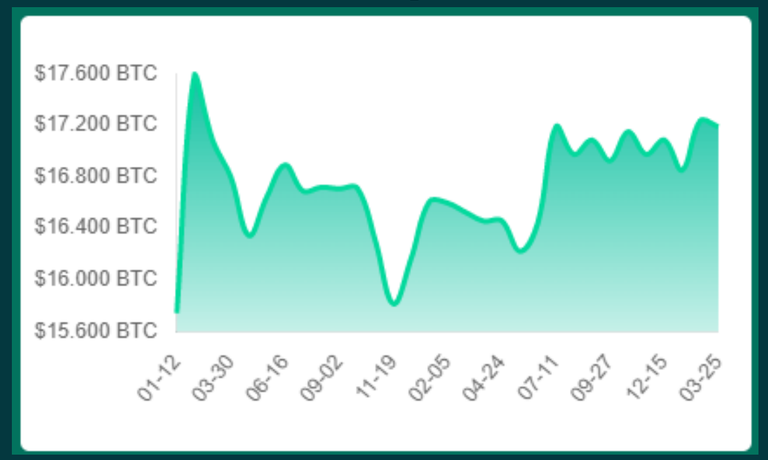
<CoinRow/>

```
<tr>
    <td>{index}</td>
    <td>
      <div className="coin_image_container">
          <img src={coin.image} title={coin.name} alt={coin.name} />
      </div>
    </td>
    <td>{numberF.format(coin.current_price)}US$</td>
    <td className={colorDec(coin.market_cap_change_percentage_24h)}>
{deleteDec(coin.market_cap_change_percentage_24h, 2)}%</td>
    <td>{numberF.format(coin.total_volume)}US$</td>
    <td>{numberF.format(coin.market_cap)}US$</td>
    <td><Graph coin={coin.id} days={7} color=
{colorDec(coin.market_cap_change_percentage_24h)}/></td>
  </tr>
```

Se reciben las props
{ coin, index }
y se crea la tabla

<Graph/>

Resultado previsto:



Se hace la importación de librerias y componentes

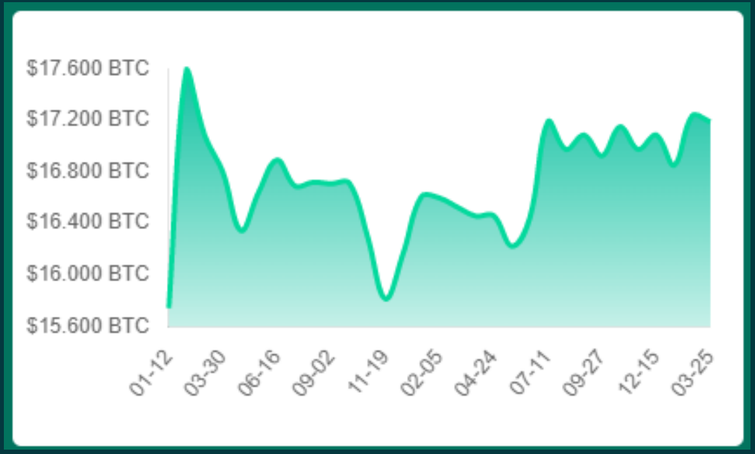
```
import { Line } from "react-chartjs-2";
import moment from "moment/moment";
import {
    Chart as ChartJS, CategoryScale,LinearScale,PointElement,
    LineElement, Title,Tooltip,Filler, Legend,
  } from 'chart.js';
```

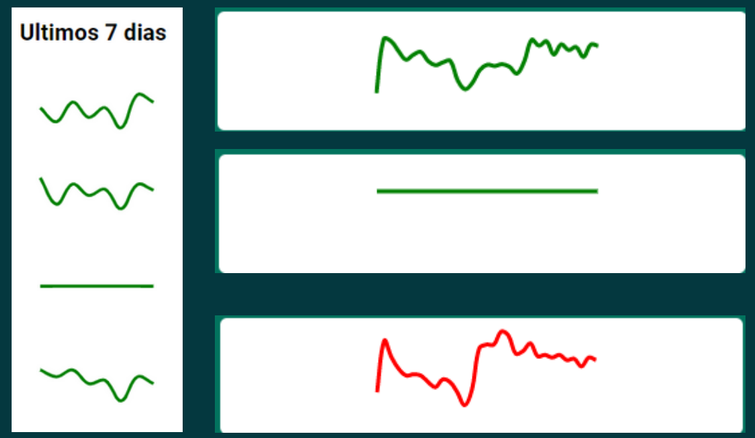
Se registra el grafico y los plugins a usar

```
ChartJS.register(
    CategoryScale, LinearScale, PointElement, LineElement,
    Title, Tooltip, Filler, Legend
)
```

<Graph/>

Resultado previsto:





Se usaron las siguientes props

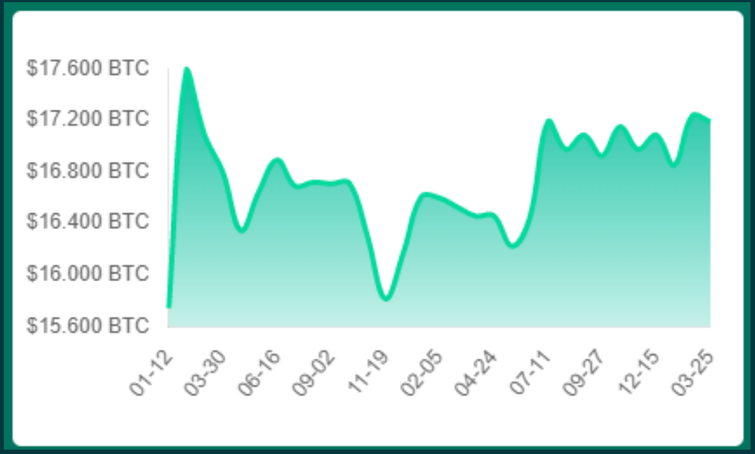
{type = 1, coin = "bitcoin", currency = "usd", days = 30,color = "#04D99D"}

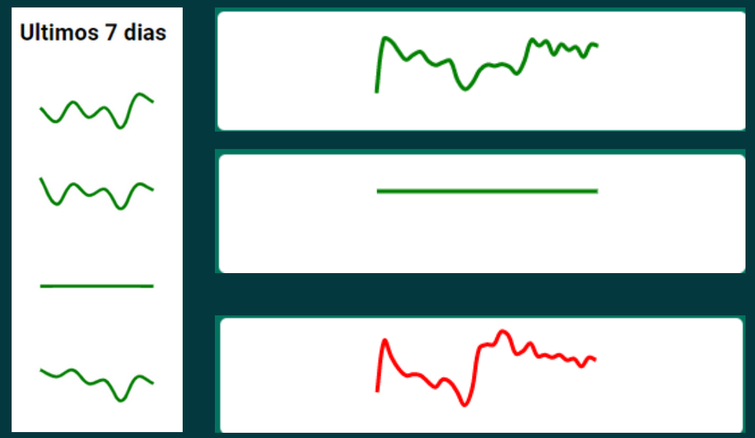
Se crea una constante con un json que utilizaremos más adelante

```
const chartStyle = {
        border: {
            display: false
        },
        grid:{
            display: false,
        },
        ticks: {
            display: false
        }
    }
```

<Graph/>

Resultado previsto:





Se consumira esta api:

https://api.coingecko.com/api/v3/coins/${coin}/market_chart?vs_currency=${currency}&days=${days}&interval=daily

Crearemos los siguientes estados y variables:

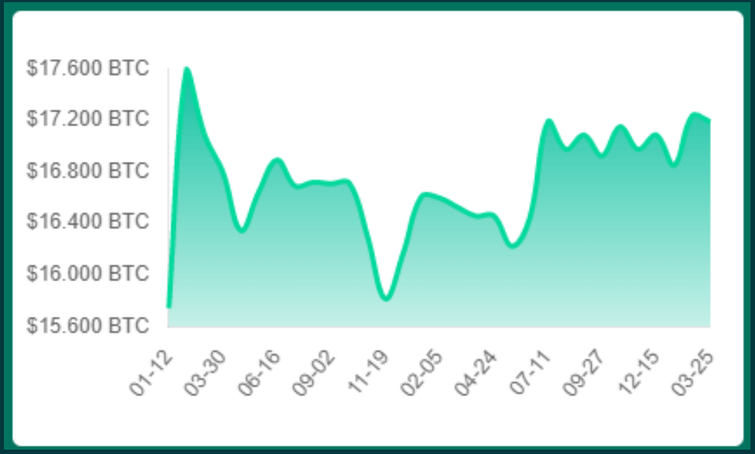
```
let data , options
const [prices, setPrices] = useState()
const [dates, setDates] = useState()
const [gradient, setGradient] = useState()
```

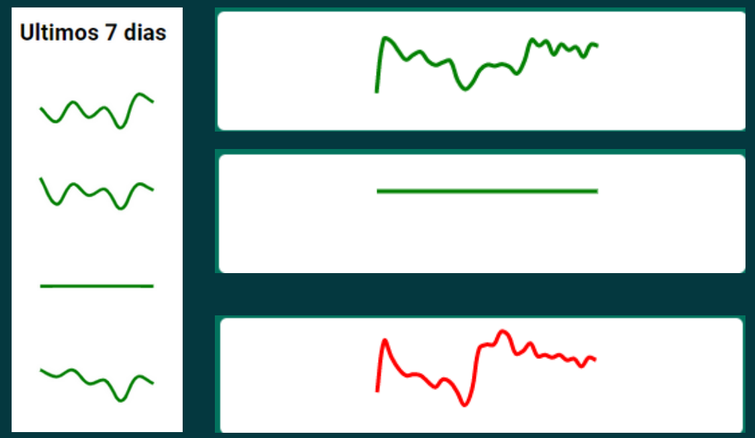
Se establecen los estados

```
setPrices(json.prices.map(item => Math.round(item[1])))
setDates(json.prices.map(item => moment.unix(item[0]).format("MM-DD")))
```

<Graph/>

Resultado previsto:





Usamos el hook ref

```
const chartRef = useRef(null);
```

Al cargar el componente definimos los gradientes

```
const canvas = chartRef.current.firstChild
let BGgradient = canvas.getContext("2d").createLinearGradient(0, 0, 0,
canvas.height);
        BGgradient.addColorStop(0, 'rgba(4, 191, 157, 1)');
        BGgradient.addColorStop(1, 'rgba(4, 191, 157, 0)')
        setGradient(BGgradient)
```

Renderizamos el componente

```
<div ref={chartRef} className="graph">
        <Line data={data} options={options}/>
  </div>
```