

Desafío 6 - Conversor de monedas

En este desafío validaremos nuestros conocimientos del método `fetch`.

Lee todo el documento antes de comenzar el desarrollo **individual**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

Aplicando los conceptos y herramientas aprendidas hasta ahora, en este desafío deberás programar un conversor de monedas a partir de un monto en pesos chilenos. Para esto será necesario consultar la api mindicador.cl usando el método `fetch`.



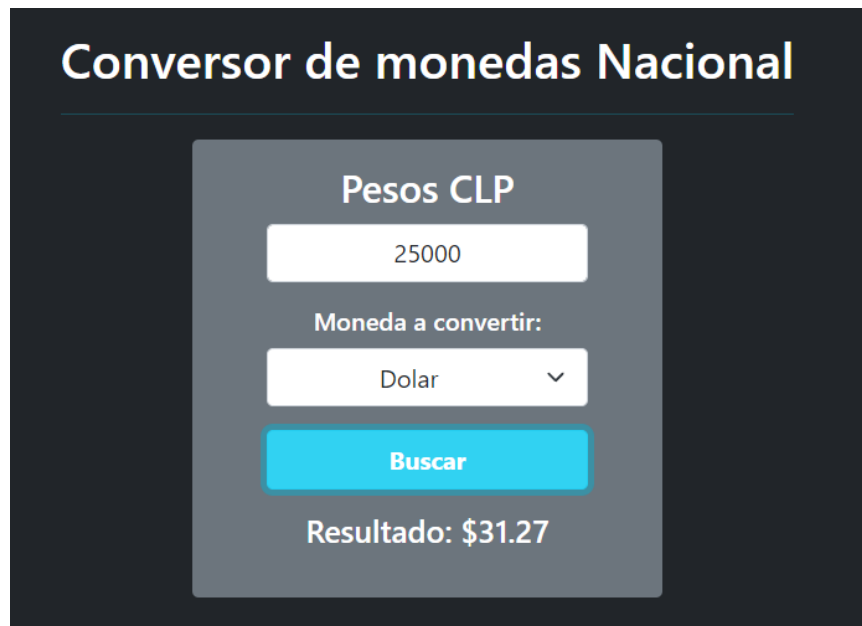
En caso que la api no se encuentre disponible, puedes utilizar la versión offline que se encuentra en la plataforma, en el archivo con nombre "Apoyo desafío - Conversor de monedas".

A continuación te mostramos lo que debes maquetar:

Imagen 1. Maqueta general del desafío
Fuente: Desafío Latam

En donde tenemos:

- Un *input* para tipear la cantidad de pesos chilenos a convertir
- Un *select* para elegir la moneda a convertir
- Un botón para iniciar el proceso de consulta y renderización de datos



The image shows a web application interface for a currency converter. The title is "Conversor de monedas Nacional". Below the title, there is a section titled "Pesos CLP". Inside this section, there is a text input field containing the number "25000". Below the input field, there is a label "Moneda a convertir:" followed by a dropdown menu showing "Dolar" with a downward arrow. Below the dropdown, there is a blue button labeled "Buscar". At the bottom of the section, it displays "Resultado: \$31.27".

Imagen 2. Búsqueda realizada con éxito
Fuente: Desafío Latam

Luego ocupa una librería de JavaScript de gráficas para mostrar un historial del valor de la moneda a convertir de los últimos 10 días.



Imagen 3. Conversión con historial
Fuente: Desafío Latam

Requerimientos

1. Se obtienen los tipos de cambio desde mindicador.cl. **(2 puntos)**
2. Se calcula correctamente el cambio y se muestra en el DOM. **(2 puntos)**
3. El select implementa más de un tipo de moneda (con 2 es suficiente), todos los cambios funcionan correctamente. **(2 puntos)**
4. Se usa *try catch* para ejecutar el método fetch y capturar los posibles errores mostrando el error en el DOM en caso de que haya problemas. **(2 puntos)**
5. Se Implementa el gráfico pedido **(2 puntos)**



¡Mucho éxito!