IMPORTANTE

El servidor donde se aloja la API puede cambiar en un futuro. En ese caso, será notificado al grupo de WhatsApp, y además se verá reflejado en este documento.

En caso de que recibas errores por demasiadas peticiones (que normalmente se muestra como un error de CORS), hay otra API que funciona de la misma manera. Tener en cuenta lo siguiente:

- URL base: https://labor3-d60e.restdb.io/rest/ (es la que se usa como baseURL)
- X-APIKEY: 64a2e9bc86d8c525a3ed8f63

Es decir, que el cliente axios quedaría de la siguiente forma:

```
const apiClient = axios.create({
        baseURL: https://labor3-d60e.restdb.io/rest/',
        headers: {'x-apikey': '64a2e9bc86d8c525a3ed8f63'}
});
```

En caso que la anterior falle, está también disponible la siguiente:

- URL base: https://laboratorio3-5459.restdb.io/rest/ (es la que se usa como baseURL)
- X-APIKEY: 64a57c2b86d8c50fe6ed8fa5

Laboratorio de computación III - Trabajo final

En el siguiente documento se encuentra la consigna para el trabajo final de la materia Laboratorio de computación III del año 2023. Constará de un sistema para manejar tu cartera de criptomonedas, y realizar cálculos con la misma.

Es posible modificar cierto comportamiento de este sistema. Por ejemplo, en vez de crear dos pantallas (una para compra de criptomonedas y otra para venta), puede ser una sola en donde se tenga que seleccionar si es compra o venta con un select. Se da cierta libertad al alumno.

El trabajo es individual.

IMPORTANTE: Notificar al celular +549 3564 523642 o al email agustinruatta@gmail.com que el trabajo ha sido entregado. El mismo debe ser subido como mínimo 2 días anteriores a la mesa de examen en el cual será evaluado (dicho tiempo será usado para estudiar el sistema hecho).

Entrega

El código fuente deberá ser administrado usando Git, y subido a un servidor de Git como GitHub o Bitbucket. Se valorará las buenas prácticas como el uso de ramas dentro de Git.

No se permiten entregas directas al email. Siempre debe estar en un repositorio Git.

Introducción al dominio

Las criptomonedas son un tipo de monedas virtuales que se pueden intercambiar entre los usuarios. La más conocida es Bitcoin, aunque existen muchísimas más (en este link puedes ver una lista de las más conocidas).

Éstas tienen un valor, que puede ser estable (por ejemplo, la criptomoneda USDC siempre tiene un valor cercano a 1 dólar), o volátil (por ejemplo, el precio del bitcoin varía constantemente).

Asimismo existen los “exchange”, que son plataformas en donde uno puede comprar y vender estas criptomonedas, y que el precio de compra y venta varía entre las mismas (se puede establecer un paralelismo con el dólar, en donde los “exchange” serían los bancos, los cuales en cada uno tiene un precio diferente -algunos más caros y otros más baratos-). Ejemplos de exchanges de Argentina son SatoshiTango, ArgenBTC, etc.

Introducción técnica

Se brinda una API para guardar los datos introducidos en la aplicación, cuya URL es https://laboratorio3-f36a.restdb.io/rest/, y que será explicada en este documento. Esto ayuda a que si se recarga la página, no se pierdan los datos (ya que son almacenados en una Base de Datos).

Es muy importante que, en cada request a esta API, se le agregue una cabecera HTTP para poder tener acceso, ya que si no se hace, va a lanzar un error. La misma es “x-apikey” y su valor es “60eb09146661365596af552f”. Por ejemplo, para agregarla a Axios sería de la siguiente forma:

```
const apiClient = axios.create({
        baseURL: 'https://laboratorio3-f36a.restdb.io/rest',
        headers: {'x-apikey': '60eb09146661365596af552f'}
});
//Hacer las llamadas con Axios
```

Las requests para obtener el precio de las criptomonedas en los diferentes exchanges se harán a la API de la página https://criptoya.com/. Para ver la documentación de la misma ingresar a https://criptoya.com/api (importante hacerlo para entender muchas cosas). Con respecto al exchange a usar, puede ser cualquiera (por ejemplo, en este documento uso SatoshiTango, aunque hay muchos más). Asimismo se puede usar otra API para obtener el precio en ARS (Pesos Argentinos) de las criptomonedas.

IMPORTANTE: aunque la documentación de la API de criptoya menciona que se debe pasar el volumen, la misma siempre devuelve el valor de una unidad. Por ejemplo, si consultamos a

https://criptoya.com/api/argenbtc/btc/ars/0.5, el valor devuelto será el valor de 1 bitcoin (no de 0.5).

Con respecto a las criptomonedas, debido a que existen muchas, la aplicación debe manejar al menos 3 (por ejemplo, Bitcoin, USDC y Ethereum), aunque pueden ser más (lo cual es mejor). La elección de estas 3 está a cargo del estudiante.

Pantalla login

La primera pantalla debe ser la de login. En esta se debe ingresar un ID alfanumérico, y que será usado ya que, como las requests van a ser a una API que lo compartirán todos los alumnos, tenemos que distinguir de alguna forma quién está llamando (entonces los datos del Alumno A no se mezclan con lo del Alumno B).

Para guardar este dato, que va a ser tomado en un componente pero usado en casi todos los otros, recomiendo alguna de estas 4 estrategias:

- Usar Vuex: explicado en la última clase de la materia (ver documento de recursos). La documentación oficial la pueden encontrar acá (está en inglés), aunque hay muchos tutoriales en español
- Pinia: es el sucesor de Vuex, el cual se usa de una forma idéntica.
- Usar localstorage: una forma simple y sencilla de almacenar datos de forma local. Al igual que Vuex, existen varios tutoriales en español (por ejemplo éste, que aunque sea explicado dentro de la documentación de Vue 2, es válido para Vue 3).
- Pasarlo a los otros componentes a través de propiedades y emisión de eventos.

Pantalla dar de alta una nueva compra

Una pantalla debe permitir dar de alta una nueva compra de criptomonedas. Para ello se debe ingresar:

- La criptomoneda que se compró (recordar que deben ser al menos 3 las opciones).
- La cantidad que se compró. Tener en cuenta de que este es un número decimal, ya que se pueden comprar porciones de criptomonedas (por ejemplo, en el caso del Bitcoin, al ser tan caro, casi siempre es menor a 1, como “0.00070”).
- La fecha y hora en la que se compró.
- Cuánto se pagó. Esto incluye el número, ya que vamos a suponer que siempre manejamos pesos Argentinos.

Una vez ingresado, validar que los datos sean correctos (por ejemplo, que la cantidad de criptomonedas y dinero gastado sean mayor a 0). Además, realizar una request POST a la API https://laboratorio3-f36a.restdb.io/rest/transactions para poder guardarla en una Base de Datos (y que no se pierda si se recarga la página). Ejemplo del cuerpo:

```
{
   "user_id": "valor_introducido_login",
   "action": "purchase",
   "crypto_code": "usdc",
```

```
    "crypto_amount": "1.01",
    "money": "165.23",
    "datetime": "11-07-2021 17:50"
}
```

Como se puede ver:

- user_id: es el valor que se introdujo en la pantalla de login
- action: indica si es una compra o venta. En este caso que estamos comprando, será “purchase”. Cuando se da de alta una venta el valor es “sale”
- crypto_code: el código de la criptomoneda, en minúsculas. Por ejemplo, para el Bitcoin será “bitcoin”, para USDC “usdc”, etc. Lo importante es elegir un código único para cada criptomoneda y respetarlo.
- crypto_amount: la cantidad de criptomoneda que se compró.
- money: cuánto se pagó en Pesos Argentinos
- date: la fecha cuando se compró. Es importante que se use el formato “DD-MM-YYYY hh:ss”, y que sea una fecha anterior al momento que se realiza la consulta.

De esta forma, ya queda guardada en la base de datos para futuras consultas.

Pantalla dar de alta una nueva venta

Asimismo como hay una pantalla para indicar una compra, debe haber una para que el usuario indique que vendió una cantidad X de criptomonedas. Para eso, la pantalla debe permitir ingresar:

- La criptomoneda que se vendió.
- La cantidad que se vendió.
- La fecha y hora.
- Cuánto se cobró.

Una vez ingresado, validar que los datos sean correctos. Además, realizar una request POST a la API https://laboratorio3-f36a.restdb.io/rest/transactions para poder guardarla. Ejemplo del cuerpo:

```
{
    "user_id": "valor_introducido_login",
    "action": "sale",
    "crypto_code": "usdc",
    "crypto_amount": "1.01",
    "money": "170.98",
    "datetime": "19-07-2021 20:50"
}
```

Notar que es muy similar a la request anterior, excepto que “action” cambia.

Asimismo, si el estudiante quisiera, esta pantalla se podría unir con la anterior en una sola (y en esta indicar con un select si es una compra o una venta, total todo lo otro es igual).

Historial de movimientos

Necesitamos otra pantalla en donde se puedan mostrar todos los movimientos de criptomonedas (es decir, todas las compras y ventas que hicimos),y su respectiva información. Para eso, se debe hacer una request GET a https://laboratorio3-f36a.restdb.io/rest/transactions?q={"user_id": "XXXXXX"} (en donde se debe reemplazar “XXXXXX” por el ID ingresado en la pantalla de login), la cual devolverá una colección de todos los movimientos. Por ejemplo:


```
[
    {
        "_id": "60eb149ba4666761000216fc",
        "crypto_code": "usdc",
        "crypto_amount": "1.01",
        "money": "170.98",
        "user_id": "valor_introducido_login",
        "action": "sale",
        "datetime": "2021-11-07T20:50:00.000Z"
    },
    {
        "_id": "60eb148da4666761000216f9",
        "crypto_code": "usdc",
        "crypto_amount": "1.01",
        "money": "165.23",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-07T17:50:00.000Z"
    }
]
```


Cuando se muestran estos movimientos, además incluir un botón para poder llevar a la lectura, edición y borrado de las mismas.

Lectura, edición y borrado de las compras/ventas

Asimismo como permitimos dar de alta una compra o venta, también debemos permitir poder leerla, modificarla o borrarla.

Lo importante es que cuando ocurre alguna de estas acciones, sea llamada la API. Supongamos que una transacción tiene el ID “60eb148da4666761000216f9” (este ID se

obtiene de cuando hacemos una request a https://laboratorio3-f36a.restdb.io/rest/transactions, o como respuesta cuando creamos una):

- Para consultar la información de una transacción, se debería llamar con GET a https://laboratorio3-f36a.restdb.io/rest/transactions/{id} (por ejemplo, https://laboratorio3-f36a.restdb.io/rest/transactions/60eb148da4666761000216f9). Esto devuelve un cuerpo como el siguiente:


```
{
   "_id": "60eb148da4666761000216f9",
   "crypto_code": "usdc",
   "crypto_amount": "1.01",
   "money": "165.23",
   "user_id": "valor_introducido_login",
   "action": "purchase",
   "datetime": "2021-11-07T17:50:00.000Z"
}
```


- Para editar la información de una transacción, se debe llamar con PATCH a https://laboratorio3-f36a.restdb.io/rest/transactions/{id} y en el cuerpo un JSON con los nuevos valores. Si por ejemplo queremos actualizar el monto gastado a 170.55, entonces deberíamos llamar a https://laboratorio3-f36a.restdb.io/rest/transactions/60eb148da4666761000216f9 e incluir un cuerpo como el siguiente:


```
{
   "money": "170.55"
}
```


- Por último, para eliminar una transacción, usar DELETE a la url https://laboratorio3-f36a.restdb.io/rest/transactions/{id}. Siguiendo con el ejemplo, si llamamos a https://laboratorio3-f36a.restdb.io/rest/transactions/60eb148da4666761000216f9 con DELETE, se borra esa fila.

Pantalla análisis del estado actual

Necesitamos una vista en donde podamos ver el estado actual de nuestras finanzas. Vamos a hacerlo mostrando:

- Las criptomonedas que tenemos, su cantidad, y cuánto dinero representa.
- El monto total de dinero que tenemos.

¿Cómo lograrlo? Veamos un ejemplo.

Supongamos que hacemos una consulta a https://laboratorio3-f36a.restdb.io/rest/transactions?q={"user_id": "XXXXXX"} (recordar de reemplazar las XXXXXX), y nos devuelve el siguiente cuerpo:


```
[
```

```
    {
        "_id": "60eb148da4666761000216f9",
        "crypto_code": "usdc",
        "crypto_amount": "1.01",
        "money": "165.23",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-07T17:50:00.000Z"
    },
    {
        "_id": "60eb149ba4666761000216fc",
        "crypto_code": "usdc",
        "crypto_amount": "1.01",
        "money": "170.98",
        "user_id": "valor_introducido_login",
        "action": "sale",
        "datetime": "2021-11-07T20:50:00.000Z"
    },
    {
        "_id": "60eb148da4666761000216f7",
        "crypto_code": "bitcoin",
        "crypto_amount": "0.01",
        "money": "58447",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-11T17:50:00.000Z"
    },
    {
        "_id": "60eb148da4666761000216f5",
        "crypto_code": "bitcoin",
        "crypto_amount": "0.02",
        "money": "116894",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-12T17:50:00.000Z"
    }
]
```


Con eso podemos deducir que:

- No tenemos USDC, debido a que aunque compramos 1.01, luego vendimos la misma cantidad.
- Tenemos 0.03 Bitcoins (ya que sumamos la compra de 0.01 y la de 0.02).

Asimismo, usando la API de https://criptoya.com/, podemos obtener el precio del bitcoin al momento actual (recordar que para entender cómo funciona esta API, se debe acceder https://criptoya.com/api). Si hacemos una consulta a https://criptoya.com/api/satoshitango/btc/ars/, y estos nos devuelve el siguiente cuerpo:


```
{
   "ask": 5912442.48,
   "totalAsk": 5971566.9,
   "bid": 5797256.86,
   "totalBid": 5739284.29,
   "time": 1626027655
}
```


Con todos estos datos (que tenemos 0.03 Bitcoins después de calcularlo, y que 1 Bitcoin valen $5.739.284,29), vamos a mostrar la pantalla con la siguiente información:

Criptomoneda	Cantidad	Dinero
Bitcoin	0.03	$ 172.178,53
Total		$ 172.178,53

USDC no se muestra debido a que no tenemos más (compramos pero vendimos después).

Pantalla análisis de inversiones

Por último, necesitamos una vista en donde podamos ver el resultado de nuestras inversiones (si ganamos o perdimos). Veámoslo con el siguiente ejemplo.

Supongamos que hacemos una consulta a https://laboratorio3-f36a.restdb.io/rest/transactions?q={"user_id": "XXXXXX"}, y nos devuelve el siguiente cuerpo:


```
[
   {
       "_id": "60eb148da4666761000216f9",
       "crypto_code": "usdc",
       "crypto_amount": "1.01",
       "money": "165.23",
       "user_id": "valor_introducido_login",
```

```
        "action": "purchase",
        "datetime": "2021-11-07T17:50:00.000Z"
    },
    {
        "_id": "60eb149ba4666761000216fc",
        "crypto_code": "usdc",
        "crypto_amount": "1.01",
        "money": "170.98",
        "user_id": "valor_introducido_login",
        "action": "sale",
        "datetime": "2021-11-07T20:50:00.000Z"
    },
    {
        "_id": "60eb148da4666761000216f7",
        "crypto_code": "bitcoin",
        "crypto_amount": "0.01",
        "money": "58447",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-11T17:50:00.000Z"
    },
    {
        "_id": "60eb148da4666761000216f5",
        "crypto_code": "bitcoin",
        "crypto_amount": "0.02",
        "money": "116894",
        "user_id": "valor_introducido_login",
        "action": "purchase",
        "datetime": "2021-11-12T17:50:00.000Z"
    }
]
```


Además, si hacemos una consulta a https://criptoya.com/api/satoshitango/btc/ars, nos devuelve el siguiente cuerpo:


```
{
    "ask": 5912442.48,
    "totalAsk": 5971566.9,
    "bid": 5797256.86,
    "totalBid": 5739284.29,
```

```
    "time": 1626027655
}
```


Con toda esa información ya podemos ver que:

1. Ganamos $5,75 comprando y vendiendo USDC, ya que lo calculamos como 170,98−165,23 (uno es el precio de la venta, y el otro de la compra).
2. Perdimos $3162,47 con la compra de Bitcoin. Esto se debe a que realice el cálculo “Precio actual - precio pagado“ = “(5739284,29 * 0.03) - (58447 + 116894)”, en dónde “(5739284,29 * 0.03) = 172178,53” es el precio actual de 0.03 bitcoins, $58.447 es lo que gaste comprando los 0.01 Bitcoins, y $116.894 es lo que gaste comprando los otros 0.02 Bitcoins.

Por ende, esta pantalla debería mostrar la siguiente información

Criptomoneda	**Resultado**
USDC	+ $5,75
Bitcoin	- $3162,47

Tener en cuenta de que obviamente puede haber tanto pérdidas como ganancias.

Mejoras a la aplicación

Aunque compleja, esta aplicación está lejos de poder usarse en un ambiente profesional. Por ende, se le pueden hacer mejoras que **no son obligatorias**, pero sumarán puntaje extra a la nota final (es decir, no son necesarias hacerlas sino opcionales, pero si sentís que querés aprender más, ¡es ideal que se hagan!).

Dificultad Media: Uso de tests

Una aplicación sin testing unitarios y de integración no puede considerarse robusta. Incluso se pueden usar varios enfoques como TDD.

Investigar cómo se introducen tests en Vue.js y en Javascript, y realizarlo.

Dificultad Media: Agregar gráficos

Usando alguna librería Javascript, agregar gráficos para, por ejemplo, mostrar a través de un gráfico circular la composición de mi cartera de inversión.

Dificultad Media: ¿dónde comprar o vender?

Como se puede ver en la página de https://criptoya.com, cada exchange tiene un precio de compra y venta diferente. Es por eso que se le podría agregar a la aplicación una pantalla en donde se selecciona la criptomoneda, si se desea comprar o vender, y busque entre todos los

exchange que lo trabajan para ver en cuál de todos conviene realizar la operación (ya que te pagan más en caso de vender, o te sale más barato en caso de comprar).