Assignment

This assignment will test your backend development skills, including API integration, data processing, and building robust endpoints to serve up-to-date currency exchange data.

Note: Please, take the time to read all the documents at once before starting, and remember: It's so much better DONE than PERFECT, so focus primarily on must-have features and leave nice-to-have's to the end!

Business Logic

For regional purposes, candidates from Brazil will need a USD to BRL currency converter, while candidates from Argentina will need a USD to ARS currency converter.

Tech stack

- NodeJS
- Any SQL-based database of your choice

1. Backend

Create an HTTP server (using `Node.js`) that exposes three different endpoints:

1.a Quotes

GET `/quotes`: It returns an array of objects with USD quotes (or "dollar citations") retrieved from 3 different sources:

ARS:

- https://www.ambito.com/contenidos/dolar.html
- https://www.dolarhoy.com
- https://www.cronista.com/MercadosOnline/moneda.html?id=ARSB

BRL:

- https://wise.com/es/currency-converter/brl-to-usd-rate
- https://nubank.com.br/taxas-conversao/
- https://www.nomadglobal.com

The objects must have the following minimum structure/attributes (you can add new useful attributes and/or insights)

```
{
  "buy_price": 140.3,
  "sell_price": 144,
  "source": "https://www.ambito.com/contenidos/dolar.html"
}
```

1.b Average

GET `/average`: It returns an object with the average positions of all the quotes.

The objects must have the following minimum structure/attributes (you can add new useful attributes and/or insights)

```
{
  "average_buy_price": 142.3,
  "average_sell_price": 147.4
}
```

1.c Slippage

GET `/slippage`: It returns an array objects with the information of how much slippage percentage there is between each source and the average.

The objects must have the following minimum structure/attributes (you can add new useful attributes and/or insights)

```
{
  "buy_price_slippage": 0.04,
  "sell_price_slippage": -0.06,
  "source": "https://www.ambito.com/contenidos/dolar.html"
}
```

There is no requirement of *how* to collect the information (you can either use web scrapping techniques, reverse engineering, just HTTP requests, etc.)

Must-have requirements:

- Always retrieve fresh information (max time between last update is 60s)
- Deploy the project and make it available in some public URL
- Deploy the project on GitHub