

TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ Môn: Công Nghệ Thông Tin



BÀI TẬP LỚN

MÔN HỌC

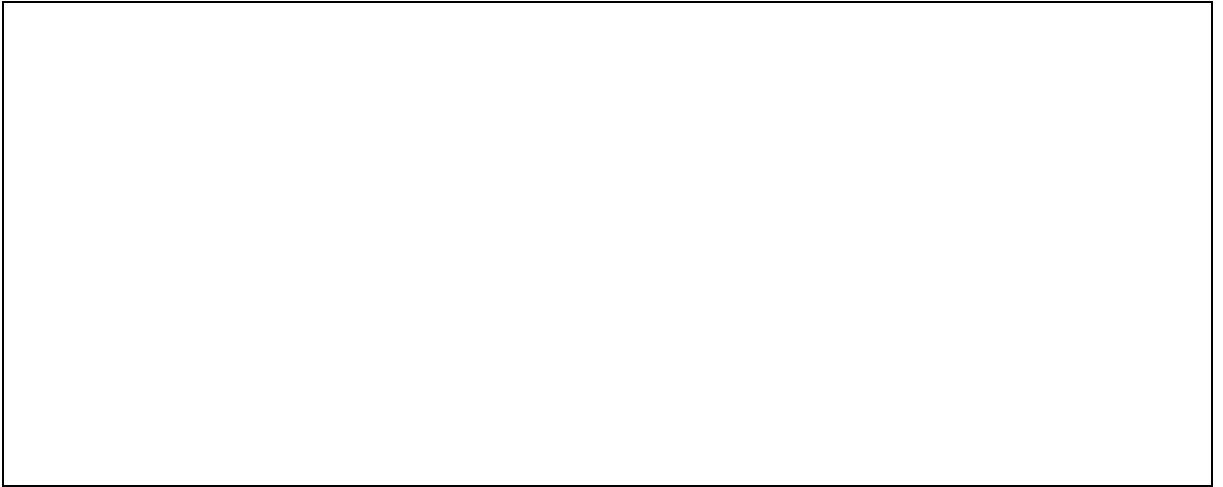
LẬP TRÌNH PYTHON

Sinh viên: Dương Văn Ninh

Lớp: 56KMT.01

Giáo viên hướng dẫn: Th.S Đỗ Duy Cốp

Thái Nguyên 2024



TRƯỜNG ĐH KỸ THUẬT CÔNG NGHỆ XÃ HỘI CHỦ NGHĨA VIỆT NAM

KHOA ĐIỆN TỬ

Độc lập – Tự do – Hạnh phúc

BÀI TẬP LỚN: LẬP TRÌNH PYTHON

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

Sinh viên: Dương Văn Ninh

Lớp: 56KMT

Nghành học: Kỹ thuật máy tính.

Ngày giao đề tài: 15/05/2024

Ngày hoàn thành: 26/05/2024

Nội dung bài làm: Đề tài: Theo dõi các tỉ giá tiền tệ.

1. Python và Fastapi.

- Sử dụng python tạo fastapi lấy dữ liệu tỉ giá tiền tệ từ API.
- API: <https://app.exchangerate-api.com/dashboard/confirmed>

2. Node-red:

- Xây dựng một chu trình trong Node-RED để tự động gọi API Python để lấy dữ liệu.

- Sau đó sử dụng các function xử lý dữ liệu và lưu dữ liệu vào MSSQL.

3. MSSQL.

- Table: lưu tỉ giá của tiền tệ.
- Tạo Sp insert lấy dữ liệu từ node-red
- Tạo Sp lấy dữ liệu và vẽ biểu đồ thể hiện tỉ giá trên web.
- Tạo Sp lấy dữ liệu và hiện thị lịch sử ra bảng trên web.

4. Web:

- Sử dụng HTML, CSS, JS để tạo giao diện web.
- Sử dụng ASP.NET để vào DB lấy dữ liệu rồi trả về web hiện thị lịch sử và vẽ biểu đồ.
- Có thêm phần chuyển đổi tiền tệ.

TRƯỜNG BỘ MÔN

GIÁO VIÊN HƯỚNG DẪN

Ths. Đỗ Duy Cốp

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

MỤC LỤC

MỤC LỤC	5
LỜI NÓI ĐẦU	6
PHẦN I: PYTHON + FASTAPI.....	7
Phần II: NODE-RED + SQL.....	11
PHẦN 3: C# + WEB	17

LỜI NÓI ĐẦU

Xin chào và chào mừng đến với dự án theo dõi tỉ giá ngoại tệ của chúng tôi. Trong thời đại toàn cầu hóa kinh tế ngày nay, việc theo dõi và hiểu biết về tỉ giá ngoại tệ đóng vai trò quan trọng trong quyết định kinh doanh, đầu tư và quản lý tài chính. Điều này là do tỉ giá ngoại tệ không chỉ ảnh hưởng đến giá trị của hàng hóa và dịch vụ mà còn có thể tác động lớn đến lợi nhuận và rủi ro của các tổ chức và cá nhân. Dự án của chúng tôi được tạo ra với mục đích cung cấp một giải pháp tự động và hiệu quả để theo dõi và phân tích tỉ giá ngoại tệ. Thông qua việc kết hợp các công nghệ như Python, FastAPI, Node-RED, MSSQL và ASP.NET, chúng tôi xây dựng một hệ thống toàn diện cho việc thu thập, xử lý và hiển thị dữ liệu tỉ giá ngoại tệ. Trong phần tiếp theo của dự án, chúng tôi sẽ trình bày chi tiết về cách hệ thống của chúng tôi hoạt động và những lợi ích mà nó mang lại. Chúng tôi hy vọng rằng dự án này sẽ mang lại giá trị cho các nhà kinh doanh, nhà đầu tư và bất kỳ ai quan tâm đến thị trường tài chính toàn cầu. Cảm ơn bạn đã tham gia vào dự án của chúng tôi và chúng tôi mong muốn nhận được sự phản hồi tích cực từ phía các bạn để cải thiện và phát triển dự án ngày càng tốt hơn.

Vì thời gian làm đề tài ngắn và là lần đầu thực nghiệm nên em không tránh khỏi những thiếu sót, mong thầy cô sẽ có những ý kiến để em hoàn thiện hơn về phần mềm này.

PHẦN I: PYTHON + FASTAPI

B1: Lấy dữ liệu từ trang: ExchangeRate-API

The screenshot shows the ExchangeRate-API dashboard. On the left sidebar, there are links for Dashboard, API Keys, Request Usage, Account Details, Manage Plan, Billing, Invoices, Documentation, Docs Overview, Supported Currencies, Standard API Requests, Product Homepage, Email Support, and Terms & Conditions. The main content area is titled "API Access" and displays the API Key: f00110473747395c7d05d4dc. It also shows an example request URL: https://v6.exchangerate-api.com/v6/f00110473747395c7d05d4dc/latest/USD. A blue banner at the bottom of this section says "Get Pro! • More Accurate Hourly Updates • 30k Reqs p/m • Only \$10/m or \$100/y 2 Months Free!". Below this is the "API Request Quota Usage" section, which shows the API request quota available (1481), API requests today (0), API requests this month (19), API requests last month (0), and the monthly quota reset date (15th). The "Account & Plan Summary" section shows the current plan as "Free" and payment status as "No Payment Required".

Link của trang: <https://app.exchangerate-api.com/dashboard>

Trang sẽ trả về dữ dưới dạng json như thế này:

The screenshot shows a browser window displaying the JSON response from the API. The URL in the address bar is https://v6.exchangerate-api.com/v6/f00110473747395c7d05d4dc/latest/USD. The page content is a large JSON object with the following structure:

```
{
  "result": "success",
  "documentation": "https://www.exchangerate-api.com/docs",
  "terms_of_use": "https://www.exchangerate-api.com/terms",
  "time_last_update_unix": 1716422401,
  "time_last_update_utc": "Thu, 23 May 2024 00:00:01 +0000",
  "time_next_update_unix": 1716508801,
  "time_next_update_utc": "Fri, 24 May 2024 00:00:01 +0000",
  "base_code": "USD",
  "conversion_rates": {
    "USD": 1,
    "AED": 3.6725,
    "AFN": 71.9089,
    "ALL": 92.4853,
    "AMD": 388.1414,
    "ANG": 1.7900,
    "AOA": 860.4597,
    "ARS": 864.7500,
    "AUD": 1.5068,
    "AWG": 1.7900,
    "AZN": 1.7004,
    "BAM": 1.8065,
    "BBD": 2.0000,
    "BDT": 117.1760,
    "BGN": 1.8064,
    "BHD": 0.3760,
    "BIF": 2867.9870,
    "BMD": 1.0000,
    ...
  }
}
```

B2: Tạo 1 file Python sử dụng Fastapi để lấy dữ liệu đầy lên localhost.

- Cài đặt FastAPI

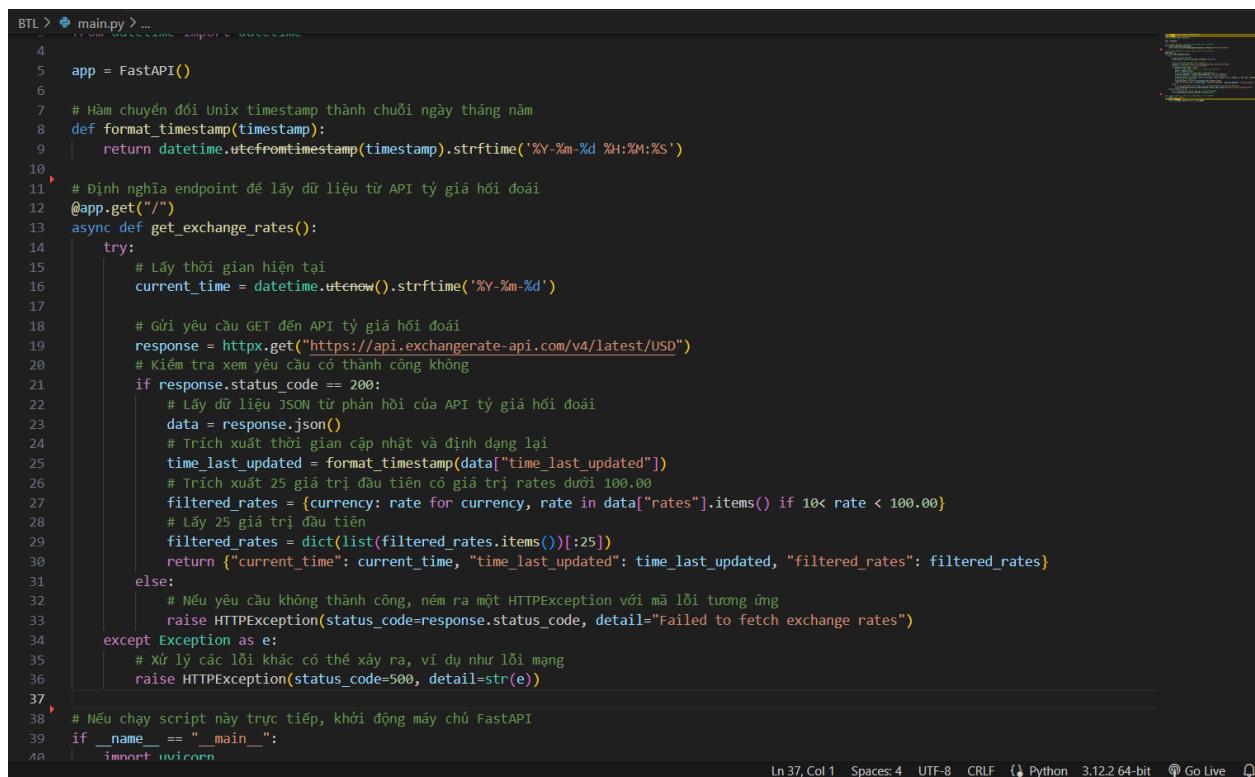
Để cài đặt framework này, bạn cần phiên bản python ≥ 3.6 .

Lệnh cài: `pip install fastapi`

Cần ASGI server khi deploy sản phẩm như Uvicorn.

Lệnh cài: `pip install uvicorn`

- Tạo file python để sử dụng FastAPI



```

1  #!/usr/bin/env python3
2  # coding: utf-8
3
4  app = FastAPI()
5
6
7  # Hàm chuyển đổi Unix timestamp thành chuỗi ngày tháng năm
8  def format_timestamp(timestamp):
9      return datetime.datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')
10
11 # Định nghĩa endpoint để lấy dữ liệu từ API tỷ giá hối đoái
12 @app.get("/")
13 async def get_exchange_rates():
14     try:
15         # Lấy thời gian hiện tại
16         current_time = datetime.datetime.utcnow().strftime("%Y-%m-%d")
17
18         # Gửi yêu cầu GET đến API tỷ giá hối đoái
19         response = httpx.get("https://api.exchangerate-api.com/v4/latest/USD")
20         # Kiểm tra xem yêu cầu có thành công không
21         if response.status_code == 200:
22             # Lấy dữ liệu JSON từ phản hồi của API tỷ giá hối đoái
23             data = response.json()
24             # Trích xuất thời gian cập nhật và định dạng lại
25             time_last_updated = format_timestamp(data["time_last_updated"])
26             # Trích xuất 25 giá trị đầu tiên có giá trị rates dưới 100.00
27             filtered_rates = {currency: rate for currency, rate in data["rates"].items() if 10 < rate < 100.00}
28             # Lấy 25 giá trị đầu tiên
29             filtered_rates = dict(list(filtered_rates.items())[:25])
30             return {"current_time": current_time, "time_last_updated": time_last_updated, "filtered_rates": filtered_rates}
31         else:
32             # Nếu yêu cầu không thành công, ném ra một HTTPException với mã lỗi tương ứng
33             raise HTTPException(status_code=response.status_code, detail="Failed to fetch exchange rates")
34     except Exception as e:
35         # Xử lý các lỗi khác có thể xảy ra, ví dụ như lỗi mạng
36         raise HTTPException(status_code=500, detail=str(e))
37
38 # Nếu chạy script này trực tiếp, khởi động máy chủ FastAPI
39 if __name__ == "__main__":
40     import uvicorn

```

Ln 37, Col 1 Spaces:4 UTF-8 CRLF Python 3.12.2 64-bit Go Live

Gửi yêu cầu để lấy API:

```

# Gửi yêu cầu GET đến API tỷ giá hối đoái
response = httpx.get("https://api.exchangerate-api.com/v4/latest/USD")

```

```
if response.status_code == 200:  
    # Lấy dữ liệu JSON từ phản hồi của API tỷ giá hối đoái  
    data = response.json()  
    # Trích xuất thời gian cập nhật và định dạng lại  
    time_last_updated = format_timestamp(data["time_last_updated"])  
    # Trích xuất 25 giá trị đầu tiên có giá trị rates dưới 100.00  
    filtered_rates = {currency: rate for currency, rate in data["rates"].items() if 10 < rate < 100.00}  
    # Lấy 25 giá trị đầu tiên  
    filtered_rates = dict(list(filtered_rates.items())[:25])  
    return {"current_time": current_time, "time_last_updated": time_last_updated, "filtered_rates": filtered_rates}  
else:  
    # Nếu yêu cầu không thành công, ném ra một HTTPException với mã lỗi tương ứng  
    raise HTTPException(status_code=response.status_code, detail="Failed to fetch exchange rates")
```

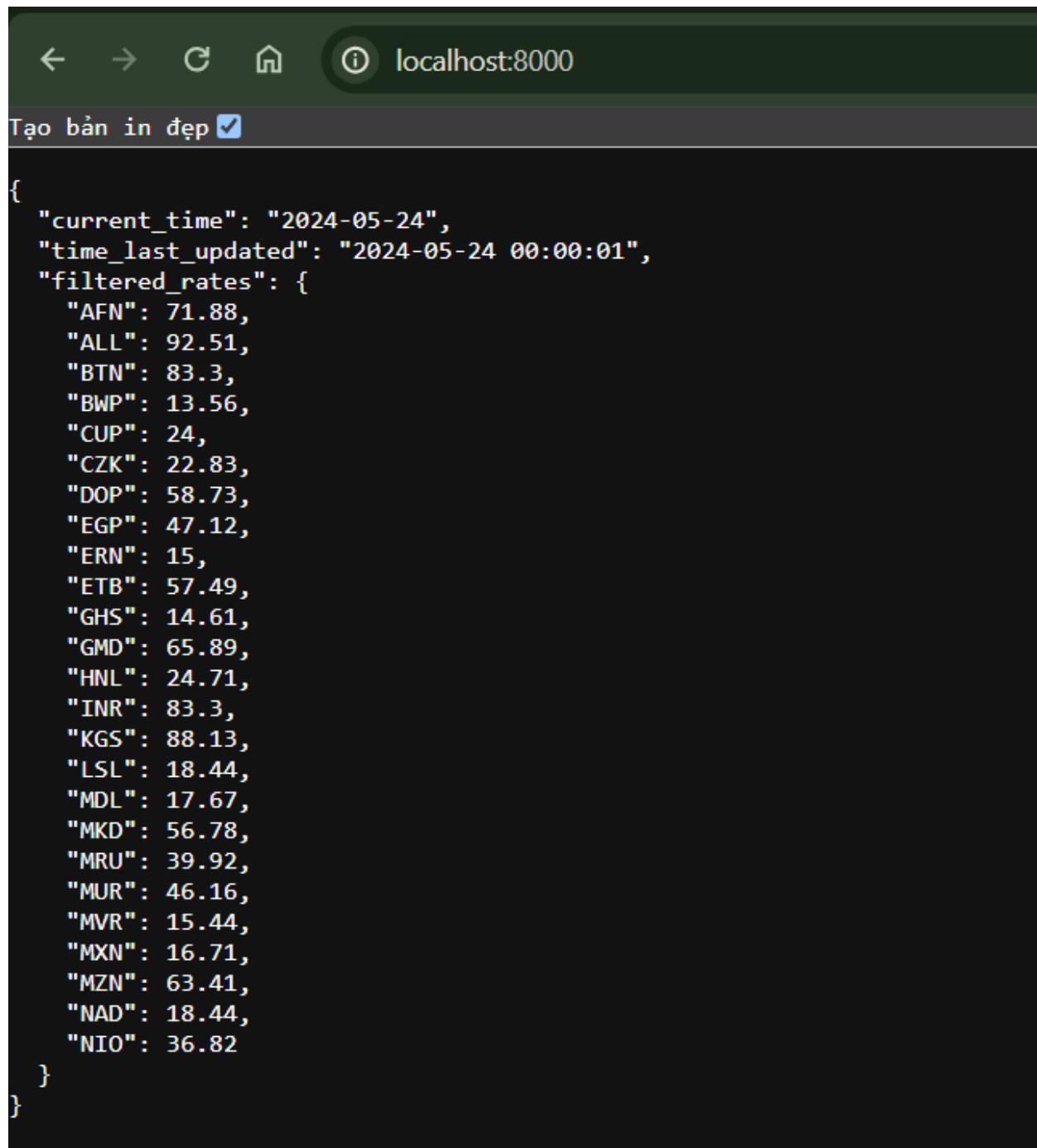
Status_code == 200 => lấy dữ liệu thành công

Vì chuỗi json trả về rất nhiều giá trị tiền tệ của các nước các giá trị từ 1 > 89500 nên em chỉ lấy các giá trị từ 10 < rate < 100 và 25 giá trị đầu tiên để tiện cho việc vẽ biểu đồ biểu diễn các giá trị.

- Chạy uvicorn bằng câu lệnh: **uvicorn main:app --reload**
- Có thể cấu hình cổng ngay khi chạy: **uvicorn main:app -port=8123**

```
PS D:\BTL> uvicorn main:app --reload  
INFO:     Will watch for changes in these directories: ['D:\\BTL']  
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)  
INFO:     Started reloader process [14312] using Watchfiles  
INFO:     Started server process [20940]  
INFO:     Waiting for application startup.  
INFO:     Application startup complete.
```

Giá trị trả về:



A screenshot of a web browser window titled "localhost:8000". The address bar shows the URL. Below the title bar, there is a toolbar with icons for back, forward, search, and refresh. The main content area displays a JSON object. At the top of the JSON object, there is a checkbox labeled "Tạo bản in đẹp" which is checked.

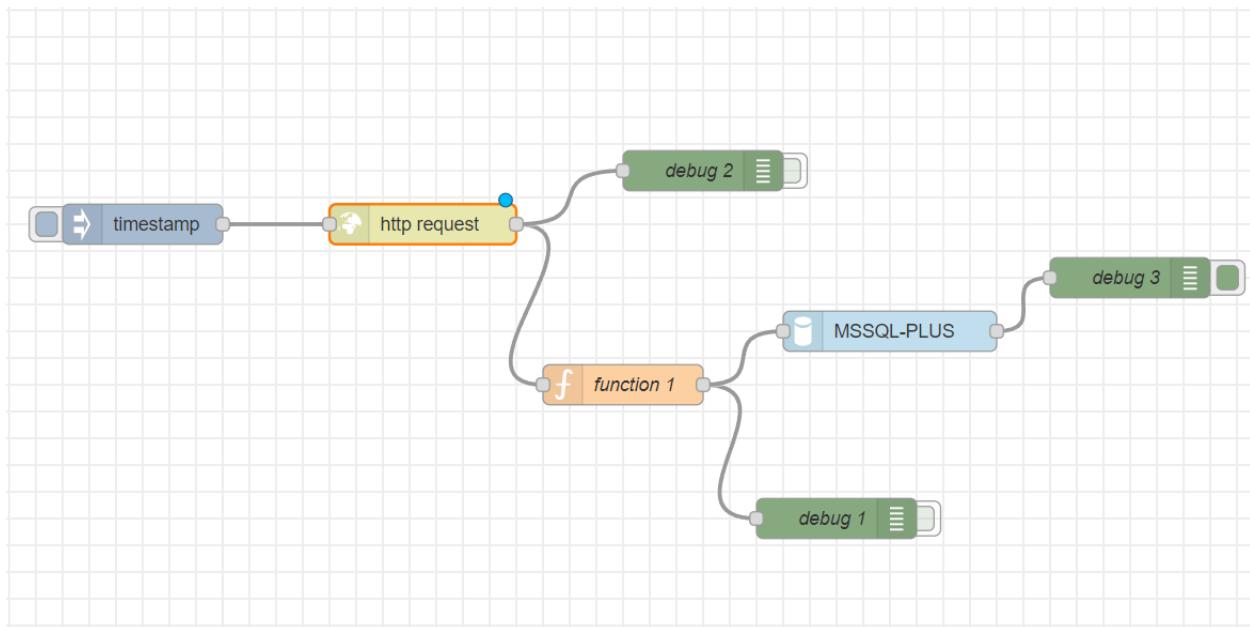
```
{  
    "current_time": "2024-05-24",  
    "time_last_updated": "2024-05-24 00:00:01",  
    "filtered_rates": {  
        "AFN": 71.88,  
        "ALL": 92.51,  
        "BTN": 83.3,  
        "BWP": 13.56,  
        "CUP": 24,  
        "CZK": 22.83,  
        "DOP": 58.73,  
        "EGP": 47.12,  
        "ERN": 15,  
        "ETB": 57.49,  
        "GHS": 14.61,  
        "GMD": 65.89,  
        "HNL": 24.71,  
        "INR": 83.3,  
        "KGS": 88.13,  
        "LSL": 18.44,  
        "MDL": 17.67,  
        "MKD": 56.78,  
        "MRU": 39.92,  
        "MUR": 46.16,  
        "MVR": 15.44,  
        "MXN": 16.71,  
        "MZN": 63.41,  
        "NAD": 18.44,  
        "NIO": 36.82  
    }  
}
```

Phân II: NODE-RED + SQL

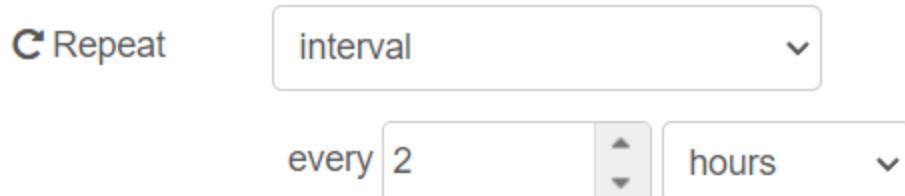
B1: Cài Đặt Node-red

Hướng dẫn cài và cách dùng: <https://www.youtube.com/watch?v=p74GGc5CZoc>

B2: Sử dụng Node-red để lấy dữ liệu



- Block **inject**: để thời gian request lại lấy dữ liệu là 2h nếu như Node-red được bật.



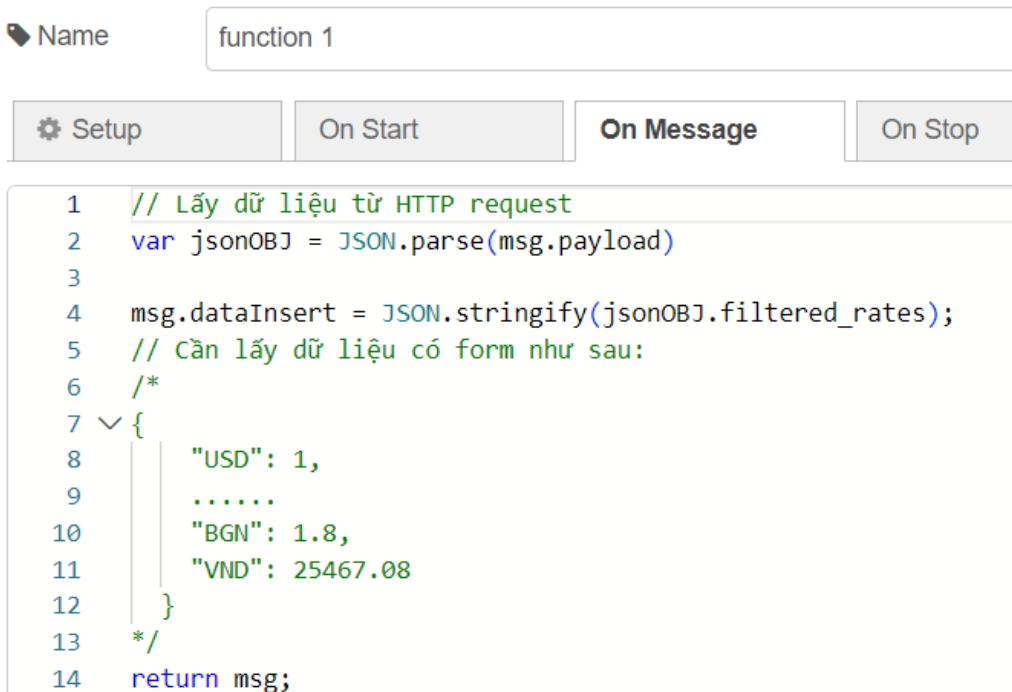
- Block **http request:** lấy dữ liệu từ fastapi

Method	GET
URL	http://localhost:8000
Payload	Ignore

Lấy được dữ liệu:

```
11:14:10 24/5/2024 node: debug 2
msg.payload : string[382]
{
  "current_time": "2024-05-24",
  "time_last_updated": "2024-05-24 00:00:01",
  "filtered_rates": {
    "AFN": 71.88,
    "ALL": 92.51,
    "BTN": 83.3,
    "BWP": 13.56,
    "CUP": 24,
    "CZK": 22.83,
    "DOP": 58.73,
    "EGP": 47.12,
    "ERN": 15,
    "ETB": 57.49,
    "GHS": 14.61,
    "GMD": 65.89,
    "HNL": 24.71,
    "INR": 83.3,
    "KGS": 88.13,
    "LSL": 18.44,
    "MDL": 17.67,
    "MKD": 56.78,
    "MRU": 39.92,
    "MUR": 46.16,
    "MVR": 15.44,
    "MXN": 16.71,
    "MZN": 63.41,
    "NAD": 18.44,
    "NIO": 36.82
}
```

- Block **Function**: xử lý dữ liệu lấy được => lưu vào SQL

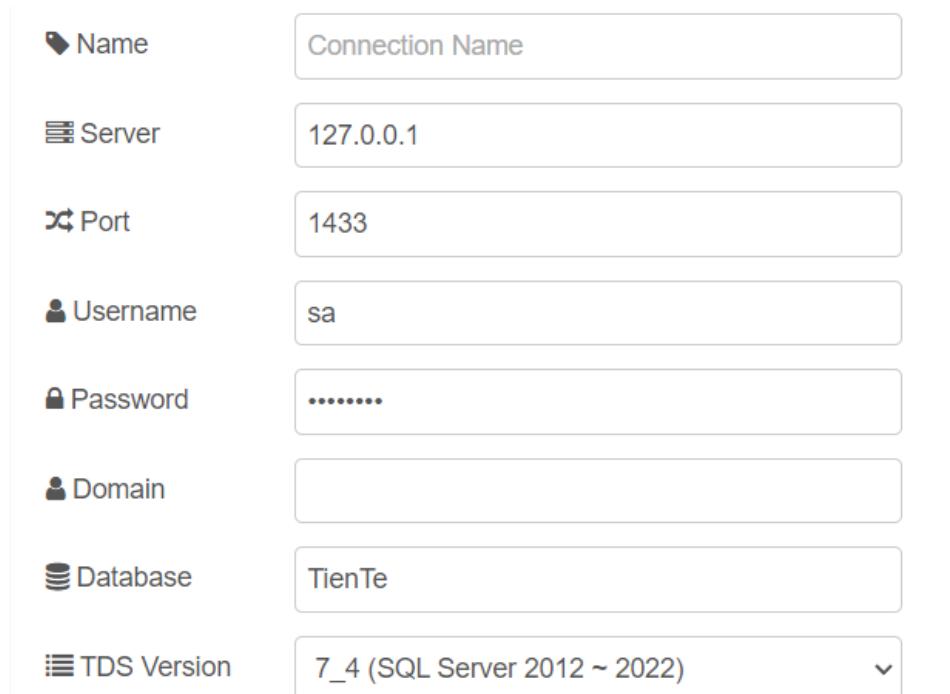


```

1 // Lấy dữ liệu từ HTTP request
2 var jsonOBJ = JSON.parse(msg.payload)
3
4 msg.dataInsert = JSON.stringify(jsonOBJ.filtered_rates);
5 // Cần lấy dữ liệu có form như sau:
6 /*
7 {
8   "USD": 1,
9   .....
10  "BGN": 1.8,
11  "VND": 25467.08
12 }
13 */
14 return msg;

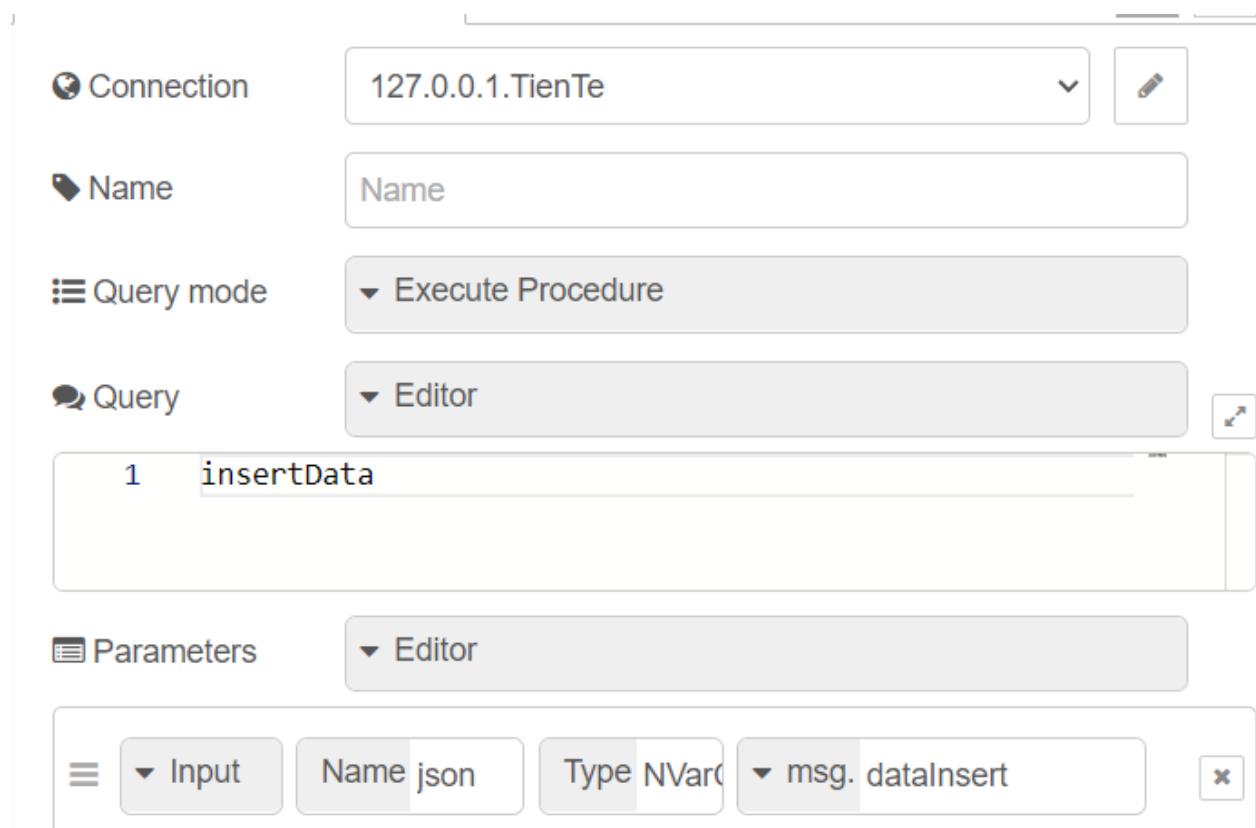
```

- Block **MSSQL-PLUS**: cấu hình để lưu được dữ liệu vào SQL.



Name	Connection Name
Server	127.0.0.1
Port	1433
Username	sa
Password
Domain	
Database	TienTe
TDS Version	7_4 (SQL Server 2012 ~ 2022)

Gọi đúng Store Procedure trong SQL và truyền các giá trị đúng với function.



B3: Sử dụng SQL để lưu dữ liệu

- Tạo 1 bảng để lưu trữ dữ liệu.(id, currency_code,rate, date_updated)

DESKTOP-4TQI2HI\...bo.exchange_rates				X	SQLQuery1.sql - DE...-4TQI2HI\Pio (70)
	Column Name	Data Type	Allow Nulls		
PK	id	int	<input type="checkbox"/>		
	currency_code	varchar(10)	<input checked="" type="checkbox"/>		
	rate	decimal(18, 2)	<input checked="" type="checkbox"/>		
	date_updated	datetime	<input checked="" type="checkbox"/>		
			<input type="checkbox"/>		

- Tạo 1 strore procedure tên là [insertData] để insert dữ liệu vào database

```

ALTER PROCEDURE [dbo].[insertData]
    @json NVARCHAR(MAX)
AS
BEGIN
    -- Sử dụng OPENJSON để phân tích chuỗi JSON và chèn vào bảng, kèm theo giá trị date_updated bằng GETDATE()
    INSERT INTO exchange_rates (currency_code, rate, date_updated)
    SELECT [key], [value], GETDATE()
    FROM OPENJSON(@json);
END

```

Dữ liệu được insert database mỗi khi node-red gọi store procedure

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer tree view lists various database objects like 'Tables', 'Stored Procedures', and 'Triggers'. In the center, the 'Results' tab of a query editor window displays the contents of the 'exchange_rates' table. The table has four columns: 'id', 'currency_code', 'rate', and 'date_updated'. The data consists of 33 rows, each representing a currency with its code, rate, and the timestamp of the update (May 23, 2024). The 'date_updated' column contains values such as '2024-05-23 20:42:33.580'.

	id	currency_code	rate	date_updated
1	6927	AFN	71.91	2024-05-23 20:42:33.580
2	6928	ALL	92.49	2024-05-23 20:42:33.580
3	6929	BTN	83.29	2024-05-23 20:42:33.580
4	6930	BWP	13.50	2024-05-23 20:42:33.580
5	6931	CUP	24.00	2024-05-23 20:42:33.580
6	6932	CZK	22.83	2024-05-23 20:42:33.580
7	6933	DOP	58.74	2024-05-23 20:42:33.580
8	6934	EGP	46.75	2024-05-23 20:42:33.580
9	6935	ERN	15.00	2024-05-23 20:42:33.580
10	6936	ETB	57.46	2024-05-23 20:42:33.580
11	6937	GHS	14.52	2024-05-23 20:42:33.580
12	6938	GMD	66.20	2024-05-23 20:42:33.580
13	6939	HNL	24.72	2024-05-23 20:42:33.580
14	6940	INR	83.29	2024-05-23 20:42:33.580
15	6941	KGS	88.66	2024-05-23 20:42:33.580
16	6942	LSL	18.26	2024-05-23 20:42:33.580
17	6943	MDL	17.67	2024-05-23 20:42:33.580
18	6944	MKD	56.72	2024-05-23 20:42:33.580
19	6945	MRU	39.90	2024-05-23 20:42:33.580
20	6946	MUR	46.04	2024-05-23 20:42:33.580
21	6947	MVR	15.45	2024-05-23 20:42:33.580
22	6948	MXN	16.66	2024-05-23 20:42:33.580
23	6949	MZN	63.45	2024-05-23 20:42:33.580
24	6950	NAD	18.26	2024-05-23 20:42:33.580
25	6951	NIO	36.84	2024-05-23 20:42:33.580
26	6952	AFN	71.91	2024-05-23 20:43:26.830
27	6953	ALL	92.49	2024-05-23 20:43:26.830
28	6954	BTN	83.29	2024-05-23 20:43:26.830
29	6955	BWP	13.50	2024-05-23 20:43:26.830
30	6956	CUP	24.00	2024-05-23 20:43:26.830
31	6957	CZK	22.83	2024-05-23 20:43:26.830
32	6958	DOP	58.74	2024-05-23 20:43:26.830
33	6959	EGP	46.75	2024-05-23 20:43:26.830

- 1 store procedure để lấy dữ liệu ra và trả về chuỗi json.

```
ALTER PROCEDURE [dbo].[GetLast16ExchangeRates]
AS
BEGIN
    SET NOCOUNT ON;

    SELECT TOP 25
        currency_code,
        rate,
        date_updated
    FROM
        exchange_rates
    ORDER BY id DESC
    FOR JSON PATH, ROOT('');
END
```

Chuỗi json được trả về mỗi khi thực hiện store procedure



The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top bar, there is a dropdown set to '121 %' and a 'Messages' button. Below the top bar, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and contains a single row of JSON data. The 'Messages' tab is empty. The query window has a title bar with the text 'exec GetLast16ExchangeRates'. The results are as follows:

1	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	[{"currency_code": "NIO", "rate": 36.82, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "NAD", "rate": 18.44, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "MZN", "rate": 63.41, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "VND", "rate": 1000.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "EUR", "rate": 22.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "USD", "rate": 23.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "JPY", "rate": 200.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "CNY", "rate": 18.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "MXN", "rate": 15.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "IDR", "rate": 1200.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "BRL", "rate": 10.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "AED", "rate": 6.0, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "KWD", "rate": 0.3, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "ILS", "rate": 0.2, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "IQD", "rate": 0.1, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "OMR", "rate": 0.05, "date_updated": "2024-05-24T11:14:10.650"}, {"currency_code": "QAR", "rate": 0.02, "date_updated": "2024-05-24T11:14:10.650"}]

PHẦN 3: C# + WEB

B1: Tạo 1 file có tên là api.aspx.cs để kết nối đến cơ sở dữ liệu

```

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
    
```

No issues found | Error List | Entire Solution | 0 Errors | 0 Warnings | 0 of 1 Message | Build + IntelliSense | Search Error List

Mở chuỗi kết nối đến cơ sở dữ liệu

```
string connectionString = "Data Source=DESKTOP-4TQI2HI\SQLEXPRESS;Initial Catalog=TienTe;Integrated Security=True";
```

Gọi store procedure và sử dụng store lấy 25 giá trị cuối cùng được cập nhật

```

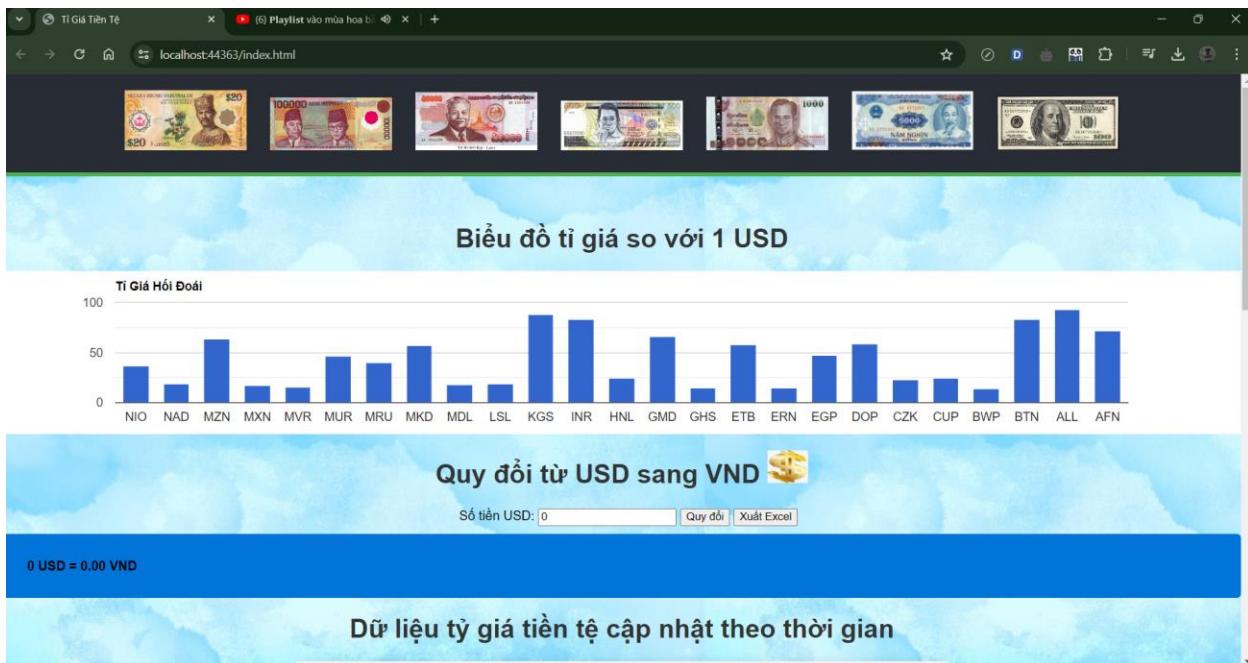
using (SqlCommand command = new SqlCommand("GetLast16ExchangeRates", connection))// gọi store procedure
{
    command.CommandType = CommandType.StoredProcedure;

    try
    {
        connection.Open(); // mở chuỗi kết nối đến csdl
        object kq = command.ExecuteScalar(); // thực hiện sp và trả về dữ liệu( sp này trả về chuỗi json)
        string json = (string)kq;
        this.Response.ContentType = "application/json";
        this.Response.Clear();
        this.Response.Write(json);
        Console.WriteLine(json);
    }
    catch (Exception ex)
    {
        this.Response.ContentType = "application/json";// đặt nd phản hồi thành kiểu json
        this.Response.Write("{\"ok\":0,\"msg\":\"" + ex.Message + "\"}");
    }
}
    
```

B2: Tạo các file html, css, js để làm web.

Link githuub: <https://github.com/vanninh2002>

Giao diện của trang web



Các chức năng:

- Xem biểu đồ tỉ giá hối đoái
- Quy đổi từ USD sang VND
- Xuất ra file Excel
- Xem lịch sử dữ liệu cập nhật.

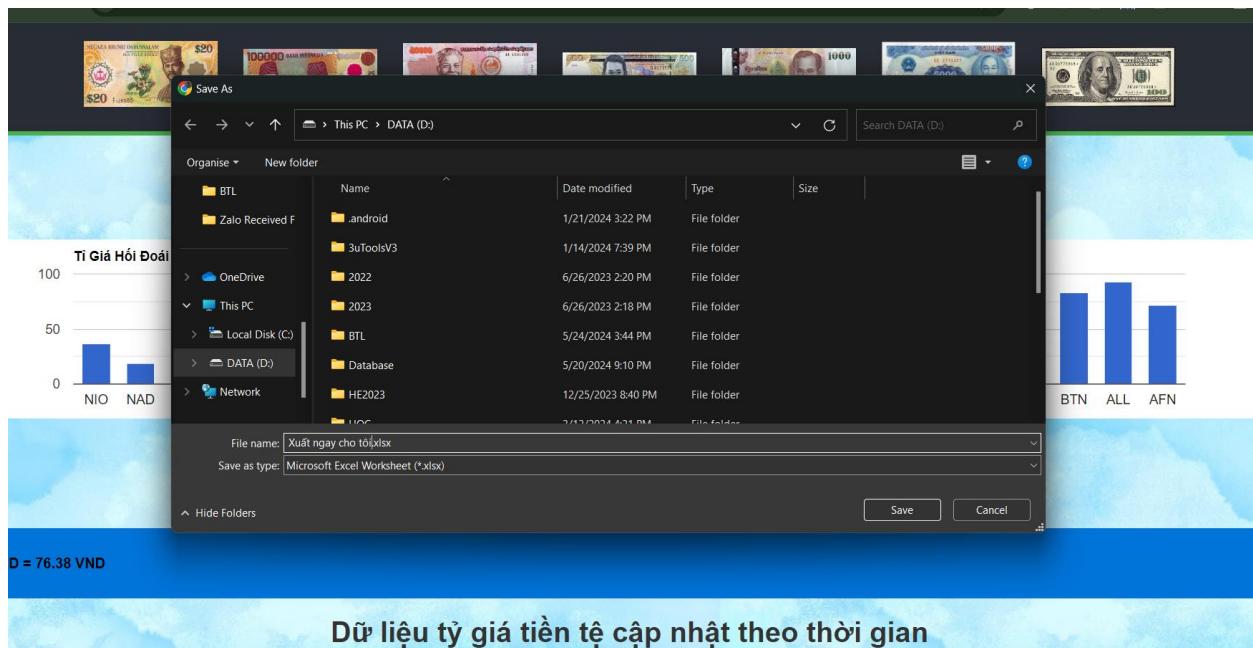
1. Xem biểu đồ tỉ giá hối đoái.



2. Quy đổi từ USD sang VND



3. Xuất ra file Excel.



4. Xem lịch sử Cập nhật

Dữ liệu tỷ giá tiền tệ cập nhật theo thời gian			
Mã Tiền Tệ	Đơn Vị Tiền Tệ Của	Tỷ Giá	Thời Gian Cập Nhật
NIO	Cordoba (Nicaragua)	36.82	2024-05-24 16:07
NAD	Dollar Namibia (Namibia)	18.44	2024-05-24 16:07
MZN	Metical (Mozambique)	63.41	2024-05-24 16:07
MXN	Peso Mexicano (Mexico)	16.71	2024-05-24 16:07
MVR	Rufiyaa (Maldives)	15.44	2024-05-24 16:07
MUR	Rupee (Mauritius)	46.16	2024-05-24 16:07
MRU	Ouguiya (Mauritania)	39.92	2024-05-24 16:07
MKD	Denar (North Macedonia)	56.78	2024-05-24 16:07
MDL	Leu (Moldova)	17.67	2024-05-24 16:07
LSL	Loti (Lesotho)	18.44	2024-05-24 16:07
KGS	Som (Kyrgyzstan)	88.13	2024-05-24 16:07
INR	Indian Rupee (India)	83.30	2024-05-24 16:07
HNL	Lempira (Honduras)	24.71	2024-05-24 16:07
GMD	Dalasi (Gambia)	65.89	2024-05-24 16:07
GHS	Cedi (Ghana)	14.61	2024-05-24 16:07
ETB	Birr (Ethiopia)	57.49	2024-05-24 16:07
ERN	Nakfa (Eritrea)	15.00	2024-05-24 16:07
EGP	Egyptian Pound (Egypt)	47.12	2024-05-24 16:07
DOP	Dominican Peso (Dominican Republic)	58.73	2024-05-24 16:07
CZK	Czech Koruna (Czech Republic)	22.83	2024-05-24 16:07
CUP	Cuban Peso (Cuba)	24.00	2024-05-24 16:07
BWP	Pula (Botswana)	13.56	2024-05-24 16:07
BTN	Ngultrum (Bhutan)	83.30	2024-05-24 16:07
ALL	Lek (Albania)	92.51	2024-05-24 16:07
AFN	Afghani (Afghanistan)	71.88	2024-05-24 16:07

PHẦN 4: KẾT LUẬN

- Qua bài tập lớn em đã học được cách lấy dữ liệu từ một trang web cung cấp api
- sử dụng node red để lưu được dữ liệu từ api vào cơ sở dữ liệu
- Lấy dữ liệu từ cơ sở dữ liệu và đưa lên web