TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

Bộ môn: Công nghệ thông tin

BÀI TẬP LỚN

LẬP TRÌNH PYTHON

Sinh viên: Vũ Ngọc Trang

Mssv: K205480106035

Lớp: K56KMT

Giáo viên hướng dẫn: Đỗ Duy Cốp

Thái Nguyên – 2024

TRƯỜNG ĐHKTCN **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

KHOA ĐIỆN TỬ ***Độc lập - Tự do - Hạnh phúc***

PHIẾU GIAO ĐỀ TÀI

Họ và tên: Vũ Ngọc Trang

Mssv: K205480106035

Lớp: K56KMT Ngành: Kỹ thuật máy tính

Giáo viên hướng dẫn: Đỗ Duy Cốp

Ngày giao đề tài: 16/05/2024 Ngày hoàn thành: 26/05/2024

1. Tên đề tài: Xây dựng web theo dõi giá vàng.

2.Nội dung yêu cầu làm:

1. Cơ sở dữ liệu: mssql, hoặc mysql (ko dùng sqlite). có các bảng gì? ghi chú đầy đủ PK, FK. Có các SP_ nào.

2. Module đọc dữ liệu: sử dụng Python + FastAPI , lấy dữ liệu từ cảm biến, hoặc từ web, hoặc từ 1 nguồn nào đó, ko sinh random. Mô tả nguồn dữ liệu và thuật toán để lấy được nó. kết hợp python với FastAPI để tạo thành 1 API cho ứng dụng khác khai thác. đóng gói nó thành service với NSSM.

3. Nodered: Chu trình tự động gọi API python để lấy về dữ liệu, có thể phải xử lý thêm, gọi node tương tác với csdl để đẩy được dữ liệu vào database (nên gọi SP và truyền tham số, ko gọi trực tiếp lệnh INSERT).

4. web: thực hiện việc lấy dữ liệu từ db ra để làm gì đó, vd như vẽ biểu đồ, hoặc làm các thứ khác liên quan đến dữ liệu đã lấy được.

GIÁO VIÊN HƯỚNG DẪN

(ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày….tháng…..năm 2024

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

MỤC LỤC

LỜI CAM ĐOAN

Tôi xin cam đoan đề tài "**Xây dựng web theo dõi giá vàng**" là đề tài nghiên cứu cá nhân của tôi trong thời gian qua. Mọi số liệu sử dụng phân tích trong đề tài và kết quả nghiên cứu là do tôi tự tìm hiểu, phân tích một cách khách quan, trung thực, có nguồn gốc rõ ràng và chưa được công bố dưới bất kỳ hình thức nào.

Tên sinh viên

CHƯƠNG I: GIỚI THIỆU VỀ NGÔN NGỮ PYTHON

Python là một ngôn ngữ lập trình bậc cao, được tạo ra bởi Guido van Rossum và phát hành lần đầu tiên vào năm 1991. Với cú pháp rõ ràng và dễ hiểu, Python đã nhanh chóng trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới. Python được thiết kế để dễ đọc và dễ viết, giúp cho các lập trình viên, kể cả những người mới bắt đầu, có thể nhanh chóng làm quen và sử dụng hiệu quả.

Python là một ngôn ngữ đa năng, hỗ trợ nhiều phong cách lập trình như lập trình hướng đối tượng, lập trình chức năng, và lập trình mệnh lệnh. Nó có một thư viện chuẩn rất phong phú, cung cấp các module và gói phần mềm để giải quyết nhiều vấn đề khác nhau, từ xử lý chuỗi và tính toán số học đến lập trình web và phát triển ứng dụng trí tuệ nhân tạo.

Một trong những điểm mạnh của Python là tính linh hoạt và khả năng mở rộng. Python có thể được sử dụng trong nhiều lĩnh vực khác nhau, từ phát triển web với các framework như Django và Flask, đến phân tích dữ liệu với các thư viện như Pandas và NumPy, và cả trong học máy với TensorFlow và scikit-learn.

Python cũng được cộng đồng lập trình viên ủng hộ mạnh mẽ, với một lượng lớn tài liệu, khóa học, và diễn đàn hỗ trợ. Điều này giúp cho việc học và phát triển với Python trở nên dễ dàng hơn bao giờ hết.

Với những ưu điểm vượt trội, Python đã và đang được sử dụng rộng rãi trong nhiều ngành công nghiệp, từ công nghệ thông tin, tài chính, y tế đến giáo dục, và trở thành một công cụ không thể thiếu đối với các lập trình viên trên toàn thế giới.

CHƯƠNG II: TỔNG QUAN VỀ ĐỀ TÀI

2.1 Giới thiệu đề tài.

Dự án xây dựng website theo dõi giá vàng nhằm cung cấp cho người dùng một công cụ tiện lợi và hiệu quả để cập nhật thông tin giá vàng theo thời gian thực. Trong bối cảnh kinh tế không ngừng biến động, giá vàng luôn là một chỉ số quan trọng được nhiều người quan tâm, từ nhà đầu tư đến người tiêu dùng cá nhân.

Website sẽ tổng hợp dữ liệu từ các nguồn tin cậy và hiển thị giá vàng dưới hình thức vẽ biểu đồ biến động. Mục tiêu của dự án là tạo ra một nền tảng dễ sử dụng, cung cấp thông tin chính xác và kịp thời, giúp người dùng đưa ra quyết định tài chính thông minh.

2.2 Lý do chọn đề tài.

Việc chọn đề tài xây dựng website theo dõi giá vàng xuất phát từ nhu cầu thực tế và tính ứng dụng cao của công cụ này trong đời sống hiện đại. Dưới đây là một số lý do chính:

1. Nhu cầu Thực tế Cao: Giá vàng luôn là một chỉ số quan trọng trong nền kinh tế, ảnh hưởng đến nhiều lĩnh vực từ đầu tư cá nhân đến chính sách kinh tế vĩ mô. Việc cung cấp thông tin chính xác và kịp thời về giá vàng là cần thiết cho nhiều người, từ nhà đầu tư, doanh nghiệp đến người tiêu dùng cá nhân.
2. Công nghệ Tiện ích: Sự phát triển của công nghệ web và các công cụ phân tích dữ liệu cho phép tạo ra một nền tảng theo dõi giá vàng linh hoạt và thân thiện với người dùng. Website này có thể cập nhật thông tin theo thời gian thực, cung cấp các biểu đồ và phân tích chi tiết, giúp người dùng dễ dàng theo dõi và phân tích biến động giá.
3. Phục vụ Cộng đồng: Website theo dõi giá vàng không chỉ phục vụ cho những cá nhân có nhu cầu đầu tư mà còn có thể hỗ trợ các doanh nghiệp kinh doanh vàng, các cửa hàng trang sức, và nhiều đối tượng khác. Điều này góp phần tăng cường sự minh bạch và hiệu quả trong việc giao dịch vàng.
4. Học hỏi và Phát triển Kỹ năng: Đề tài này cung cấp một cơ hội tuyệt vời để áp dụng và nâng cao các kỹ năng lập trình web, xử lý dữ liệu, và phân tích thông tin. Đây cũng là một thách thức thú vị để giải quyết các vấn đề thực tế bằng công nghệ.

2.3 Mục tiêu nghiên cứu.

Mục tiêu chính của dự án xây dựng website theo dõi giá vàng là phát triển một nền tảng trực tuyến cung cấp thông tin về giá vàng một cách chính xác, kịp thời và dễ hiểu cho người dùng. Các mục tiêu cụ thể bao gồm:

1. **Cung cấp Thông tin Theo Thời gian Thực:** Xây dựng hệ thống cập nhật giá vàng liên tục từ các nguồn tin cậy, đảm bảo rằng người dùng luôn có thông tin mới nhất để đưa ra quyết định chính xác.

2. **Hiển thị Dữ liệu Đa dạng và Trực quan:** Thiết kế giao diện người dùng thân thiện, cho phép hiển thị giá vàng dưới nhiều hình thức như bảng giá, biểu đồ biến động, và các phân tích xu hướng. Điều này giúp người dùng dễ dàng theo dõi và hiểu rõ tình hình thị trường vàng.
3. **Hỗ trợ Quyết định Đầu tư:** Cung cấp các công cụ phân tích và dự đoán giá vàng dựa trên dữ liệu lịch sử và các thuật toán phân tích nâng cao, hỗ trợ người dùng trong việc đưa ra các quyết định đầu tư thông minh.
4. **Khả năng Mở rộng và Tích hợp:** Xây dựng nền tảng với khả năng mở rộng để có thể dễ dàng tích hợp thêm các tính năng mới hoặc kết nối với các dịch vụ tài chính khác trong tương lai.

CHƯƠNG III: CÁC NGÔN NGỮ ĐỂ LẬP TRÌNH

3.1 Cài đặt ngôn ngữ lập trình python gồm có các ngôn ngữ để lập trình python như Pycharm, Visual Studio, Visual code…

Ở đây em dùng Pycharm để sử dụng dự án của đề tài này:

Để sử dụng ngôn ngữ này, ta cần cài ứng dụng này như sau:

Bước 1: Lên web search Pycharm download về và cài đặt

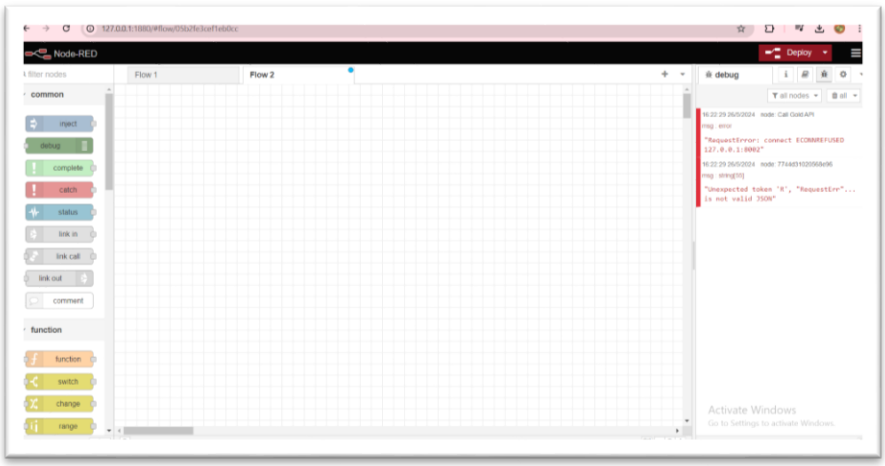
Bước 2: Cài đặt thêm Python phiên bản mới nhất và path ứng dụng này vào trong hệ thống có đường dẫn như này (“D:\python\python.exe”) ở đây em cài phiên bản 3.10.0.

Bước 3: Vào Pycharm tạo New Project để sử dụng lập trình trong dự án.

3.2 Cài đặt node-red:

Bước 1: lên web search tìm node.js download về và path ứng dụng này vào trong hệ thống ở máy tính có đường dẫn sau: (“D:\node-red\node.exe”) sau khi path xong thì chúng ta kiểm tra cài đặt dùng lệnh: “node -v” nếu nó xuất hiện ra phiên bản thì thành công.

Bước 2: Mở Terminal hoặc Command Pormpt. ấn dòng lệnh code: node-red, nó sẽ hiện ra đường dẫn http://127.0.0.1:1880/ copy đường dẫn này lên web google hoặc chorme để sử dụng môi trường lập trình của node-red. Nếu thành công nó sẽ hiện giao diện web như này.



3.3 Cài đặt SQL Server:

Bước 1: Cần cài đặt ngôn ngữ này lên web search download SQL Server, cần thiết thì cài phiên bản mới nhất cũng được, tránh trường hợp lỗi.

Bước 2: Sau khi cài xong thì ta cần Tạo tài khoản user name pass word để kế nối dữ liệu với ngôn ngữ khác.

CHƯƠNG IV: LẬP TRÌNH VÀ KIỂM THỬ

4.1 Lập trình:

Bước 1: đầu tiên để xây dựng web site theo dõi giá vàng. Ta cần dữ liệu thô của web api vàng truy cập đường dẫn https://www.goldapi.io/dashboard để lấy dữ liệu.

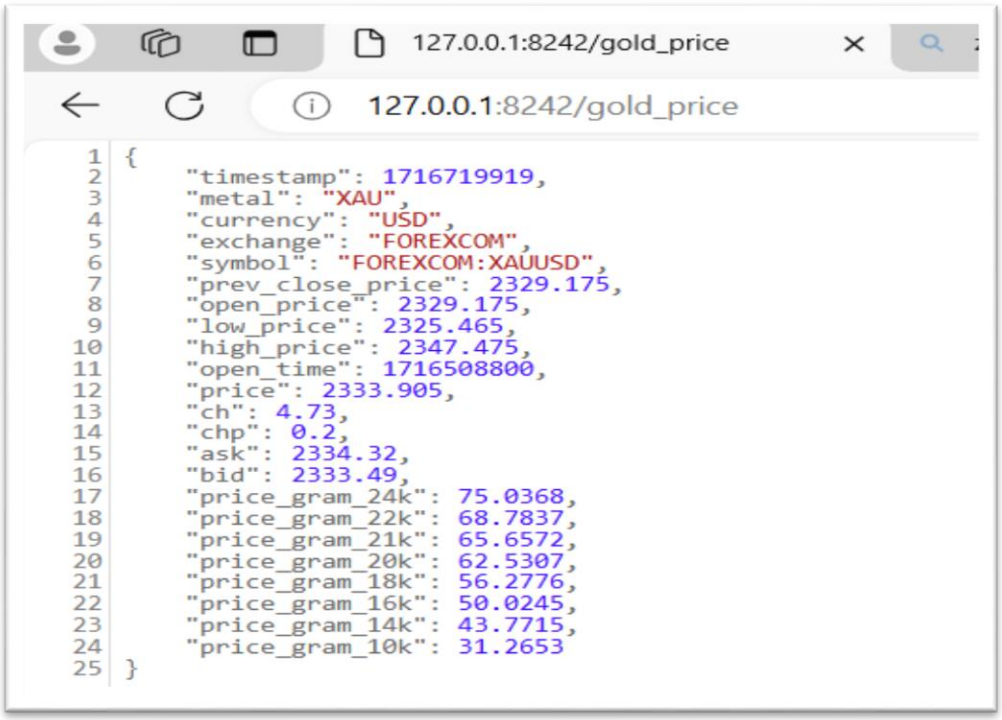
- Sau khi ta có key và url của web dữ liệu mà ta muốn.
  Url:https://www.goldapi.io/api/XAU/USD
  Key: "goldapi-h8ho519lwhhvk0y-io"
- Dùng code py fastAPI để lấy dữ liệu từ web này về:

  Tạo 1 project mới và cài đặt các modul cần thiết ở termial trong Pycharm.

```
from fastapi import FastAPI, HTTPException
import httpx
import asyncio

app = FastAPI()

GOLD_API_URL = "https://www.goldapi.io/api/XAU/USD"
API_KEY = "goldapi-h8ho519lwhhvk0y-io"
UPDATE_INTERVAL = 60  # Thời gian cập nhật (giây)
gold_price = None
async def fetch_gold_price():
    global gold_price
    headers = {"x-access-token": API_KEY}
    try:
        async with httpx.AsyncClient() as client:
            response = await client.get(GOLD_API_URL, headers=headers)
            if response.status_code == 200:
                gold_price = response.json()
            else:
                print(f"Failed to fetch data from Gold API:
{response.status_code}")
                raise HTTPException(status_code=response.status_code,
detail="Failed to fetch data from Gold API")
    except httpx.RequestError as e:
        print(f"Request error: {str(e)}")
        raise HTTPException(status_code=500, detail=f"Request error:
{str(e)}")
async def update_gold_price_periodically():
    while True:
        try:
            await fetch_gold_price()
        except Exception as e:
            print(f"Error while updating gold price: {e}")
        await asyncio.sleep(UPDATE_INTERVAL)
@app.get("/gold_price")
async def get_gold_price():
    if gold_price is None:
        await fetch_gold_price()
    return gold_price
if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.create_task(update_gold_price_periodically())
    import uvicorn
    uvicorn.run(app, host="127.0.0.1", port=8242)
```

Tạo một cái endpoint để truy cập mà mình lấy kết quả dữ liệu giá vàng cập nhật liên tục 60 giây 1 dữ liệu mới lấy về lấy về. Sau khi chạy xong thì có địa chỉ ta truy cập : http://127.0.0.1:8242 /gold_price

Thì ra kết quả dữ liệu json mà ta muốn lấy về:



Bước 2: Ta tạo bảng Table và Stored Procedure:

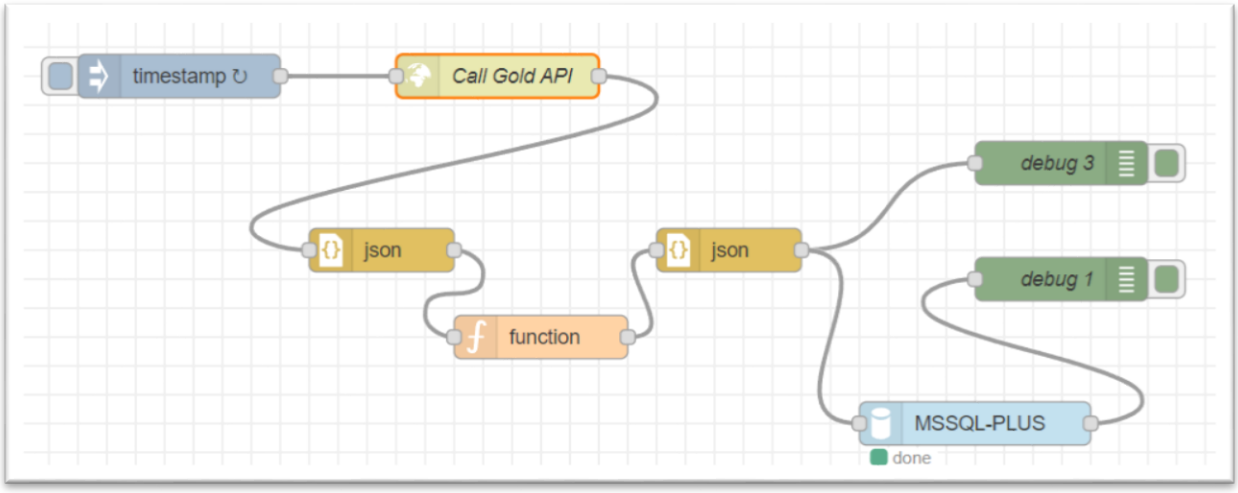
```
CREATE TABLE GoldPrices (
    timestamp BIGINT,
    metal VARCHAR(10),
    currency VARCHAR(10),
    exchange VARCHAR(50),
    symbol VARCHAR(50),
    prev_close_price FLOAT,
    open_price FLOAT,
    low_price FLOAT,
    high_price FLOAT,
    open_time BIGINT,
    price FLOAT,
    ch FLOAT,
    chp FLOAT,
    ask FLOAT,
    bid FLOAT,
    price_gram_24k FLOAT,
    price_gram_22k FLOAT,
    price_gram_21k FLOAT,
    price_gram_20k FLOAT,
    price_gram_18k FLOAT,
    price_gram_16k FLOAT,
    price_gram_14k FLOAT,
    price_gram_10k FLOAT
);
CREATE PROCEDURE InsertGoldPrice
    @timestamp BIGINT,
```

```
    @metal VARCHAR(10),
    @currency VARCHAR(10),
    @exchange VARCHAR(50),
    @symbol VARCHAR(50),
    @prev_close_price FLOAT,
    @open_price FLOAT,
    @low_price FLOAT,
    @high_price FLOAT,
    @open_time BIGINT,
    @price FLOAT,
    @ch FLOAT,
    @chp FLOAT,
    @ask FLOAT,
    @bid FLOAT,
    @price_gram_24k FLOAT,
    @price_gram_22k FLOAT,
    @price_gram_21k FLOAT,
    @price_gram_20k FLOAT,
    @price_gram_18k FLOAT,
    @price_gram_16k FLOAT,
    @price_gram_14k FLOAT,
    @price_gram_10k FLOAT
AS
BEGIN
    INSERT INTO GoldPrices (
        timestamp, metal, currency, exchange, symbol,
        prev_close_price, open_price, low_price, high_price, open_time,
        price, ch, chp, ask, bid,
        price_gram_24k, price_gram_22k, price_gram_21k,
        price_gram_20k, price_gram_18k, price_gram_16k,
        price_gram_14k, price_gram_10k
    ) VALUES (
        @timestamp, @metal, @currency, @exchange, @symbol,
        @prev_close_price, @open_price, @low_price, @high_price, @open_time,
        @price, @ch, @chp, @ask, @bid,
        @price_gram_24k, @price_gram_22k, @price_gram_21k,
        @price_gram_20k, @price_gram_18k, @price_gram_16k,
        @price_gram_14k, @price_gram_10k
    );
END;
```

Hai bảng này có công dụng:

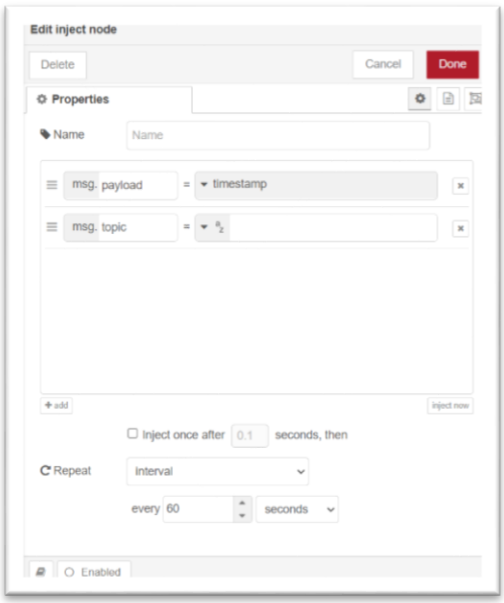
- Bảng GoldPrices lưu trữ thông tin chi tiết về giá vàng.
- Thủ tục InsertGoldPrice giúp thêm bản ghi mới vào bảng GoldPrices một cách có cấu trúc và dễ dàng hơn.

Bước 3: Tạo cấu hình node-red để lưu dữ liệu từ code py FastAPI về SQL Server.

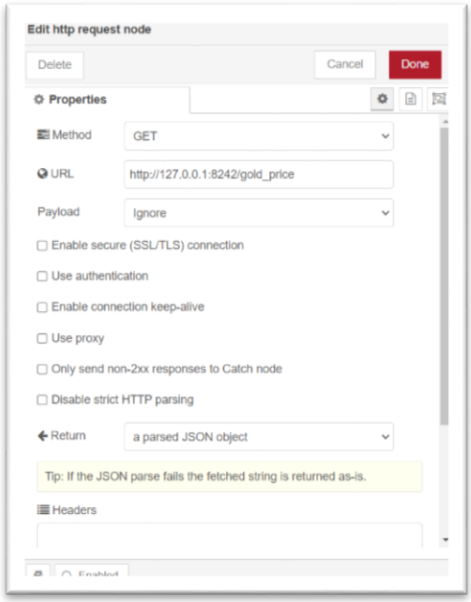


- **Gồm có:**

1. **Tab "Flow 1":** Đây là tab chứa luồng của bạn.
2. **Inject Node ("inject"):** Node này có nhiệm vụ kích hoạt luồng mỗi 60 giây một lần. Khi được kích hoạt, nó sẽ gửi một thông điệp có payload là thời gian hiện tại.

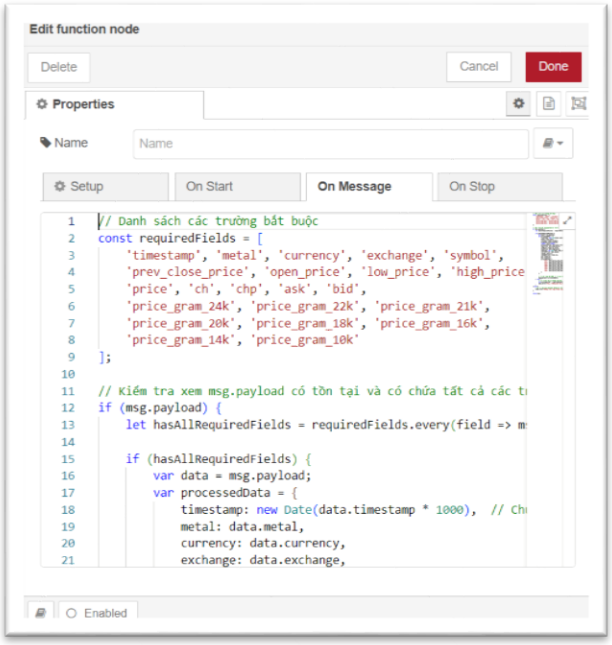


3. **HTTP Request Node ("Call Gold API"):** Node này thực hiện một yêu cầu HTTP GET tới API http://127.0.0.1:8242/gold_price để lấy dữ liệu giá vàng. Dữ liệu trả về sẽ là một đối tượng JSON.



4. **JSON Node:** Node này chuyển đổi dữ liệu trả về từ chuỗi JSON sang đối tượng JavaScript để có thể dễ dàng xử lý trong Node-RED.

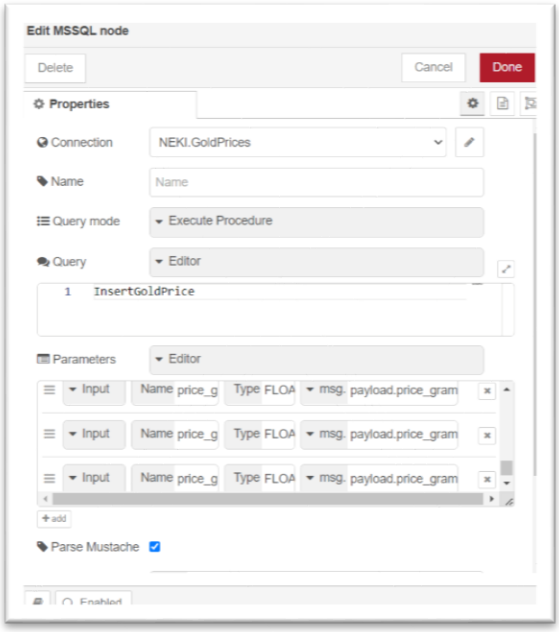
5. **Function Node:** Node này kiểm tra xem dữ liệu nhận được từ API có đầy đủ các trường cần thiết không. Nếu có, nó sẽ chuyển đổi một số trường dữ liệu (như timestamp và open_time) từ định dạng UNIX timestamp sang định dạng ngày tháng và tạo một đối tượng mới chứa các trường dữ liệu đã được xử lý. Nếu thiếu bất kỳ trường nào, nó sẽ ghi lỗi và không chuyển tiếp dữ liệu.



6. **JSON Node:** Node này chuyển đổi đối tượng JavaScript thành chuỗi JSON để có thể sử dụng trong các node tiếp theo.

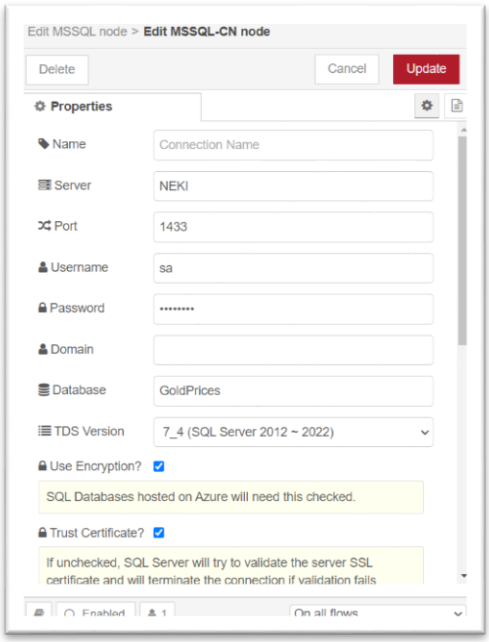
7. **Debug Node ("debug 3"):** Node này hiển thị dữ liệu sau khi đã được xử lý trong bảng debug của Node-RED để dễ dàng kiểm tra và xác minh dữ liệu.

8. **MSSQL Node:** Node này thực hiện lưu dữ liệu đã được xử lý vào cơ sở dữ liệu Microsoft SQL Server. Nó sử dụng một stored procedure có tên InsertGoldPrice để chèn các trường dữ liệu vào cơ sở dữ liệu.



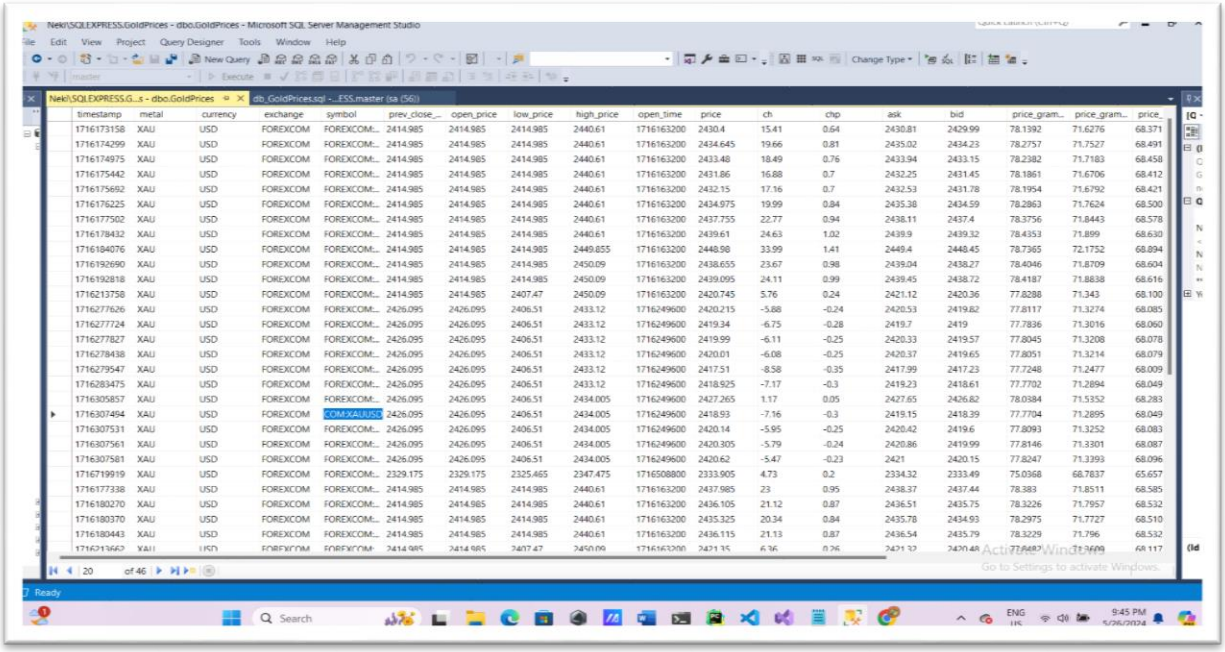
9. **Debug Node ("debug 1"):** Node này hiển thị kết quả của việc lưu dữ liệu vào cơ sở dữ liệu trong bảng debug của Node-RED.

10. **MSSQL-CN Configuration Node:** Node này chứa thông tin cấu hình để kết nối với cơ sở dữ liệu Microsoft SQL Server.



- Sau khi lưu về SQL Server ta có bảng thông tin dữ liệu update 60s liên tục giá vàng ở trong SQL Server.

  Kết quả lưu được về SQL Server:



Bước 4: Tạo một file.py sử dụng ứng dụng Flask để lấy dữ liệu ở trong SQL Server mà đã lưu về được. Xây dựng web biểu đồ và xem thông tin ở web site.

- Tạo thêm một folder chứa code giao diện html: Gồm 3 giao diện.

1. Index.html: Giao diện chính.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Giá Vàng</title>
</head>
<body>
    <h1>Gold Prices Dashboard</h1>
    <button onclick="window.location.href='/table'">View Table</button>
    <button onclick="window.location.href='/chart'">View Chart</button>
</body>
</html>
```

2. Inde.html : Giao diện hiển thị bảng dữ liệu.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gold Prices</title>
```

```
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Gold Prices</h1>
        <table class="table table-bordered mt-3" id="gold-prices-table">
            <thead>
                <tr>
                    <th>Timestamp</th>
                    <th>Metal</th>
                    <th>Currency</th>
                    <th>Exchange</th>
                    <th>Symbol</th>
                    <th>Previous Close Price</th>
                    <th>Open Price</th>
                    <th>Low Price</th>
                    <th>High Price</th>
                    <th>Open Time</th>
                    <th>Price</th>
                    <th>Change</th>
                    <th>Change Percent</th>
                    <th>Ask</th>
                    <th>Bid</th>
                    <th>Price Gram 24k</th>
                    <th>Price Gram 22k</th>
                    <th>Price Gram 21k</th>
                    <th>Price Gram 20k</th>
                    <th>Price Gram 18k</th>
                    <th>Price Gram 16k</th>
                    <th>Price Gram 14k</th>
                    <th>Price Gram 10k</th>
                </tr>
            </thead>
            <tbody></tbody>
        </table>
    </div>
    <script>
        function fetchData() {
            $.getJSON("/data", function(data) {
                var tableBody = $("#gold-prices-table tbody");
                tableBody.empty();
                data.forEach(function(row) {
                    var tr = $("<tr>");
                    Object.values(row).forEach(function(value) {
                        tr.append($("<td>").text(value));
                    });
                    tableBody.append(tr);
                });
            });
        }

        $(document).ready(function() {
            fetchData();
            setInterval(fetchData, 60000); // Tự động cập nhật mỗi 60 giây
        });
    </script>
</body>
</html>
```

3. Chart.html: Giao diện vẽ biểu đồ theo dữ liệu.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gold Price Chart</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <div class="container">
        <h1 class="mt-5">Biểu Đồ Giá Vàng</h1>
        <!-- Biểu đồ cho giá vàng -->
        <canvas id="goldPriceChart" width="800" height="400"></canvas>
    </div>
    <script>
        // Function to fetch data from the server
        async function fetchData() {
            const response = await fetch('/data');
            const data = await response.json();
            return data;
        }

        // Function to update the chart with new data
        async function updateChart() {
            const data = await fetchData();
            const labels = data.map(entry => new Date(entry.timestamp *
1000).toLocaleString());
            const prices = data.map(entry => entry.price);

            // Get the chart context
            const ctx =
document.getElementById('goldPriceChart').getContext('2d');

            // Check if the chart exists
            if (window.myChart) {
                // Update existing chart with new data
                window.myChart.data.labels = labels;
                window.myChart.data.datasets[0].data = prices;
                window.myChart.update();
            } else {
                // Create new chart if it doesn't exist
                window.myChart = new Chart(ctx, {
                    type: 'line',
                    data: {
                        labels: labels,
                        datasets: [{
                            label: 'Gold Price (USD)',
                            data: prices,
                            borderColor: 'rgba(255, 99, 132, 1)',
                            borderWidth: 1
                        }]
                    },
                    options: {
                        scales: {
                            y: {
                                beginAtZero: false
                            }
                        }
                    }
                });
            }
        }

        // Function to update the chart periodically
        async function updateChartPeriodically() {
```

```
            await updateChart();
            setInterval(updateChart, 60000); // Update every minute (adjust
interval as needed)
        }

        // Call the function to start updating the chart
        updateChartPeriodically();
    </script>
</body>
</html>
```

Code: Tạo ứng dụng xây dựng web site và lấy dữ liệu ở SQL Server để vẽ biểu đồ và bảng hiện thị số liệu:

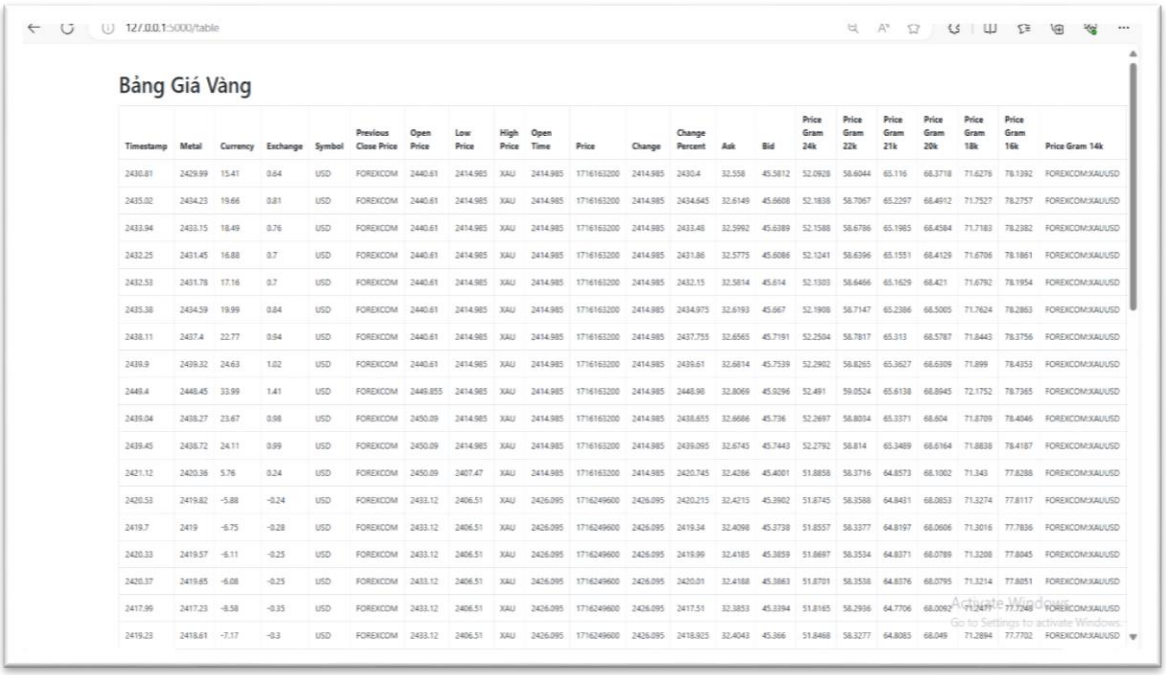
```
from flask import Flask, render_template, jsonify
import pyodbc
app = Flask(__name__)
def get_db_connection():
    try:
        conn = pyodbc.connect(
            'DRIVER={ODBC Driver 17 for SQL Server};'
            'SERVER=Neki;'
            'DATABASE=GoldPrices;'
            'UID=sa;'
            'PWD=2402'
        )
        return conn
    except pyodbc.Error as e:
        print(f"Error connecting to database: {e}")
        return None
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/data')
def get_data():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Could not connect to the database."})
    try:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM GoldPrices")
        columns = [column[0] for column in cursor.description]
        data = [dict(zip(columns, row)) for row in cursor.fetchall()]
        return jsonify(data)
    except Exception as e:
        return jsonify({"error": str(e)})
    finally:
        conn.close()
@app.route('/table')
def show_table():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Could not connect to the database."})

    try:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM GoldPrices")
        columns = [column[0] for column in cursor.description]
        data = [dict(zip(columns, row)) for row in cursor.fetchall()]
        return render_template('inde.html', columns=columns, data=data)
    except Exception as e:
        return jsonify({"error": str(e)})
    finally:
```

```
        conn.close()
@app.route('/chart')
def show_chart():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Could not connect to the database."})

    try:
        cursor = conn.cursor()
        cursor.execute("SELECT Metal, Currency, Exchange, Open_Price,
Low_Price, High_Price FROM GoldPrices")
        rows = cursor.fetchall()

        metals = [row.Metal for row in rows]
        currencies = [row.Currency for row in rows]
        exchanges = [row.Exchange for row in rows]
        open_prices = [row.Open_Price for row in rows]
        low_prices = [row.Low_Price for row in rows]
        high_prices = [row.High_Price for row in rows]

        return render_template('chart.html',
                               metals=metals,
                               currencies=currencies,
                               exchanges=exchanges,
                               open_prices=open_prices,
                               low_prices=low_prices,
                               high_prices=high_prices)
    except Exception as e:
        return jsonify({"error": str(e)})
    finally:
        conn.close()
 if __name__ == '__main__':
  app.run(debug=True)
```

4.2 Ảnh kết quả:

1. Giao diện chính web site:

2. Giao diện hiển thị bảng dữ liệu đã lưu được.



3. Giao diện vẽ biểu đồ.


127.0.0.1:5000/chart
Biểu Đồ Giá Vàng
Gold Price (USD)
2,460
2,440
2,420
2,400
2,380
2,360
2,340
2,320
Activate Windows

CHƯƠNG V: KẾT LUẬN

Xây dựng một trang web site " Theo dõi giá vàng" không chỉ là một dự án hấp hẫn và còn khám phá giá vàng cập nhật lên xuống. Và hiểu biết các ngôn ngữ lập trình và đặt tay làm việc tự code lập trình trên các ngôn ngữ như là python, mssql server, node-red…, tiếp thu được những kinh nghiệm lập trình tạo lên và sửa lỗi để hoàn thiện dự án xây dựng web site này và rút ra kỹ năng và kinh nghiệm trong tương lai. Cuối cùng em cảm ơn thầy Cốp đã chỉ dạy em kinh nghiệm trong môn học lập trình này.