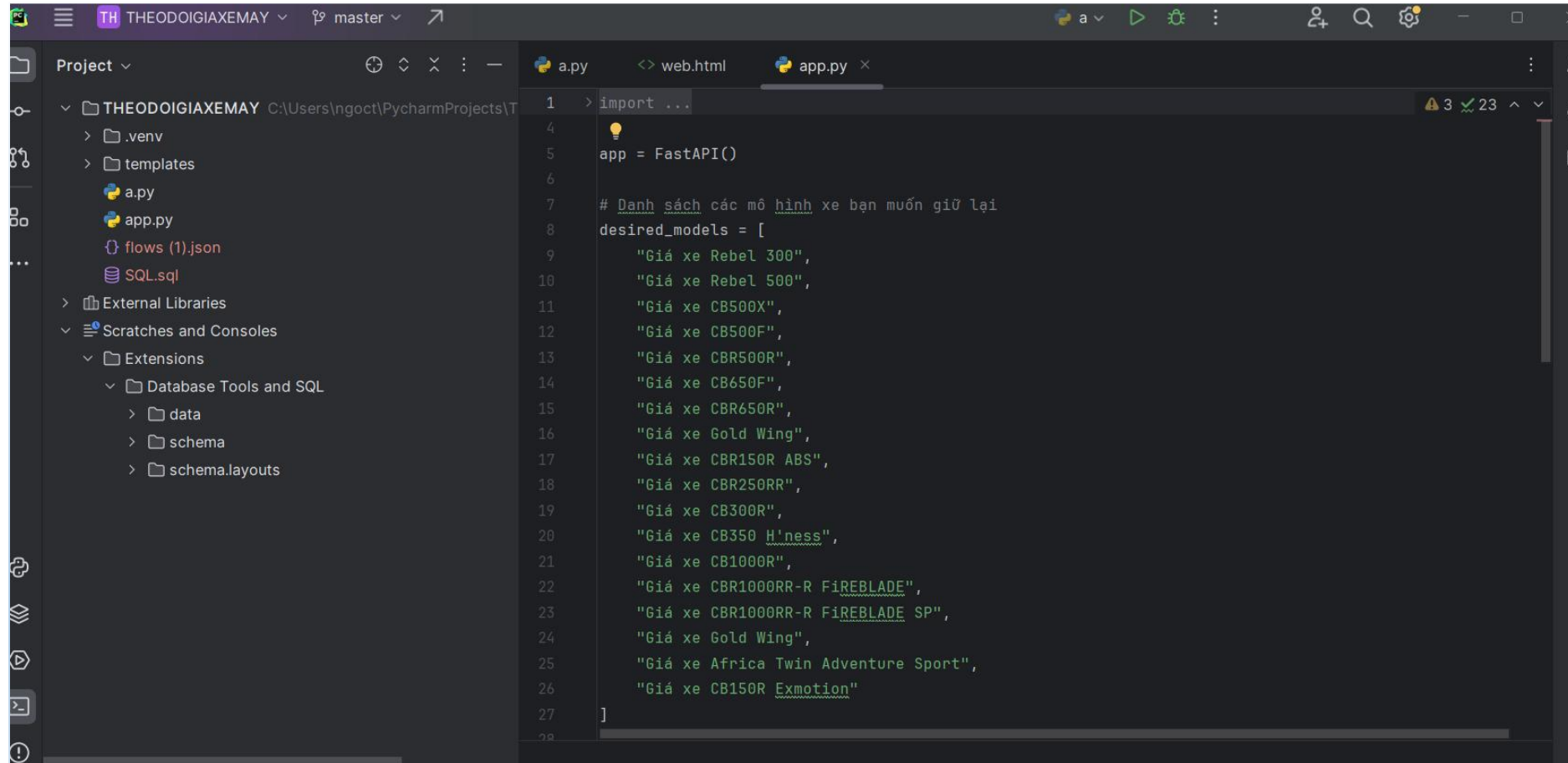


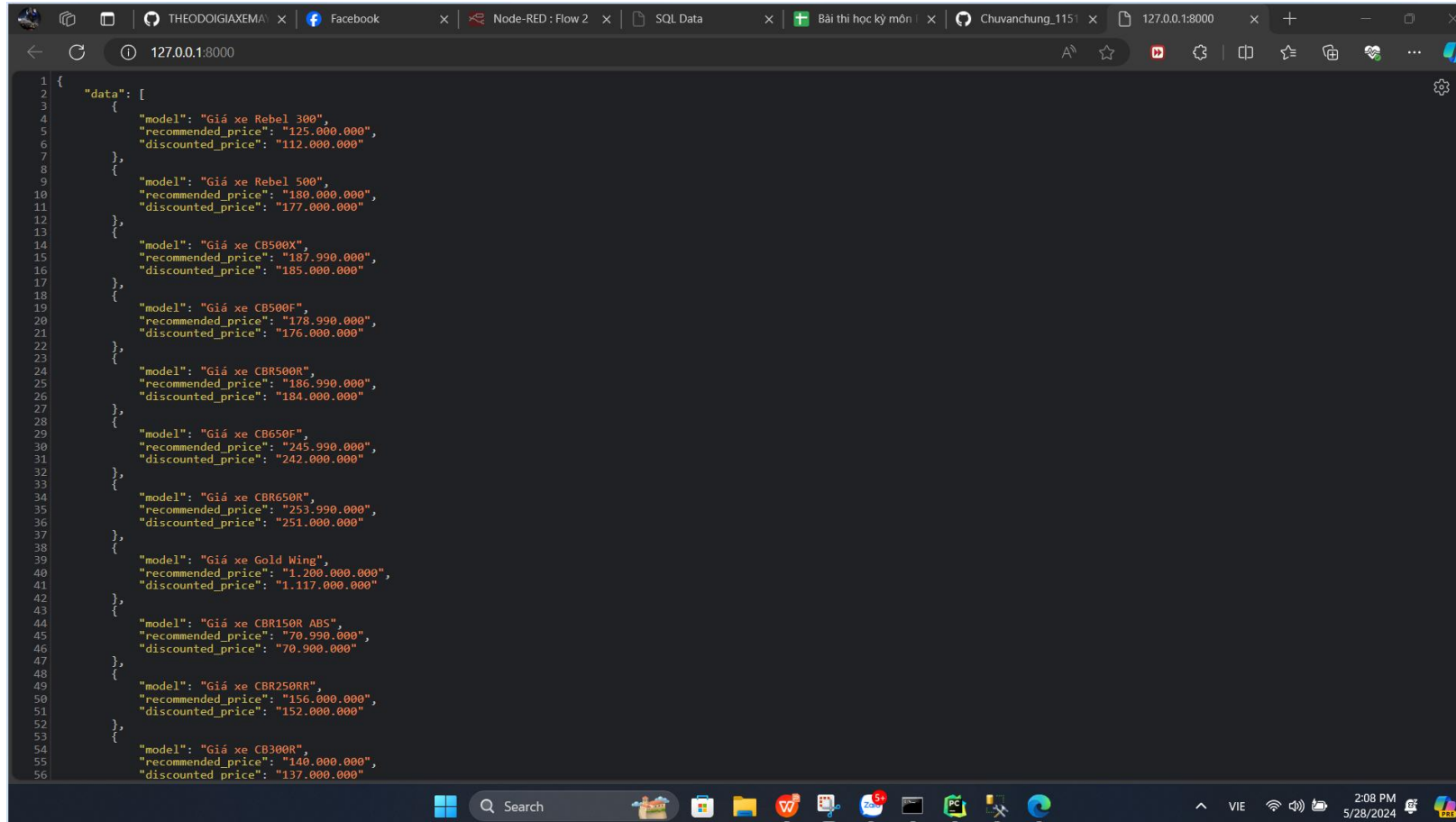
# Bước 1: Tạo hàm giả lập giá xe máy bằng python với việc sử dụng fastAPI tạo ra API



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays the file structure of the project 'THEODOIGIAXEMAY', including folders like '.venv', 'templates', and files like 'a.py', 'app.py', 'flows (1).json', and 'SQL.sql'. The main editor window shows the 'app.py' file with the following Python code:

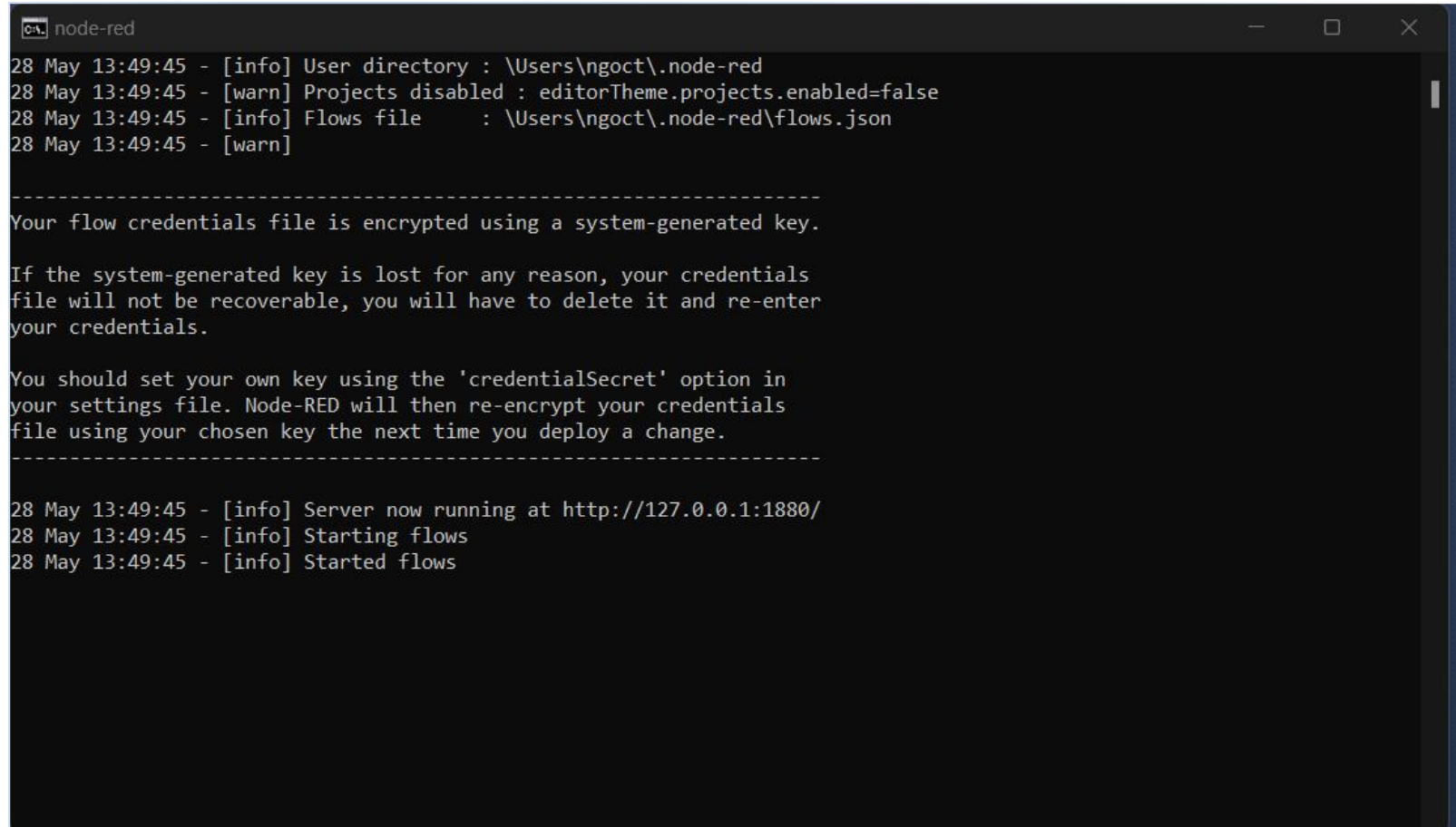
```
1 > import ...
4
5 app = FastAPI()
6
7 # Danh sách các mô hình xe bạn muốn giữ lại
8 desired_models = [
9     "Giá xe Rebel 300",
10    "Giá xe Rebel 500",
11    "Giá xe CB500X",
12    "Giá xe CB500F",
13    "Giá xe CBR500R",
14    "Giá xe CB650F",
15    "Giá xe CBR650R",
16    "Giá xe Gold Wing",
17    "Giá xe CBR150R ABS",
18    "Giá xe CBR250RR",
19    "Giá xe CB300R",
20    "Giá xe CB350 H'ness",
21    "Giá xe CB1000R",
22    "Giá xe CBR1000RR-R FIREBLADE",
23    "Giá xe CBR1000RR-R FIREBLADE SP",
24    "Giá xe Gold Wing",
25    "Giá xe Africa Twin Adventure Sport",
26    "Giá xe CB150R Exmotion"
27 ]
```

## Bước 2: Sử dụng node-red để nhận dữ liệu từ API



```
1 {
2   "data": [
3     {
4       "model": "Giá xe Rebel 300",
5       "recommended_price": "125.000.000",
6       "discounted_price": "112.000.000"
7     },
8     {
9       "model": "Giá xe Rebel 500",
10      "recommended_price": "180.000.000",
11      "discounted_price": "177.000.000"
12    },
13    {
14      "model": "Giá xe CB500X",
15      "recommended_price": "187.990.000",
16      "discounted_price": "185.000.000"
17    },
18    {
19      "model": "Giá xe CB500F",
20      "recommended_price": "178.990.000",
21      "discounted_price": "176.000.000"
22    },
23    {
24      "model": "Giá xe CBR500R",
25      "recommended_price": "186.990.000",
26      "discounted_price": "184.000.000"
27    },
28    {
29      "model": "Giá xe CB650F",
30      "recommended_price": "245.990.000",
31      "discounted_price": "242.000.000"
32    },
33    {
34      "model": "Giá xe CBR650R",
35      "recommended_price": "253.990.000",
36      "discounted_price": "251.000.000"
37    },
38    {
39      "model": "Giá xe Gold Wing",
40      "recommended_price": "1.200.000.000",
41      "discounted_price": "1.117.000.000"
42    },
43    {
44      "model": "Giá xe CBR150R ABS",
45      "recommended_price": "70.990.000",
46      "discounted_price": "70.900.000"
47    },
48    {
49      "model": "Giá xe CBR250RR",
50      "recommended_price": "156.000.000",
51      "discounted_price": "152.000.000"
52    },
53    {
54      "model": "Giá xe CB300R",
55      "recommended_price": "140.000.000",
56      "discounted_price": "137.000.000"
57    }
58  ]
59 }
```

# Tạo node-red bằng cách sử dụng câu lệnh node-red chạy trên cmd

A screenshot of a terminal window titled 'node-red'. The window shows the output of the 'node-red' command. It displays several log messages: '[info] User directory : \Users\ngoct\.node-red', '[warn] Projects disabled : editorTheme.projects.enabled=false', '[info] Flows file : \Users\ngoct\.node-red\flows.json', and '[warn]'. Below these, there is a block of text explaining that the flow credentials file is encrypted using a system-generated key and that users should set their own key using the 'credentialSecret' option in the settings file. At the bottom, it shows '[info] Server now running at http://127.0.0.1:1880/', '[info] Starting flows', and '[info] Started flows'.

```
node-red
28 May 13:49:45 - [info] User directory : \Users\ngoct\.node-red
28 May 13:49:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
28 May 13:49:45 - [info] Flows file      : \Users\ngoct\.node-red\flows.json
28 May 13:49:45 - [warn]

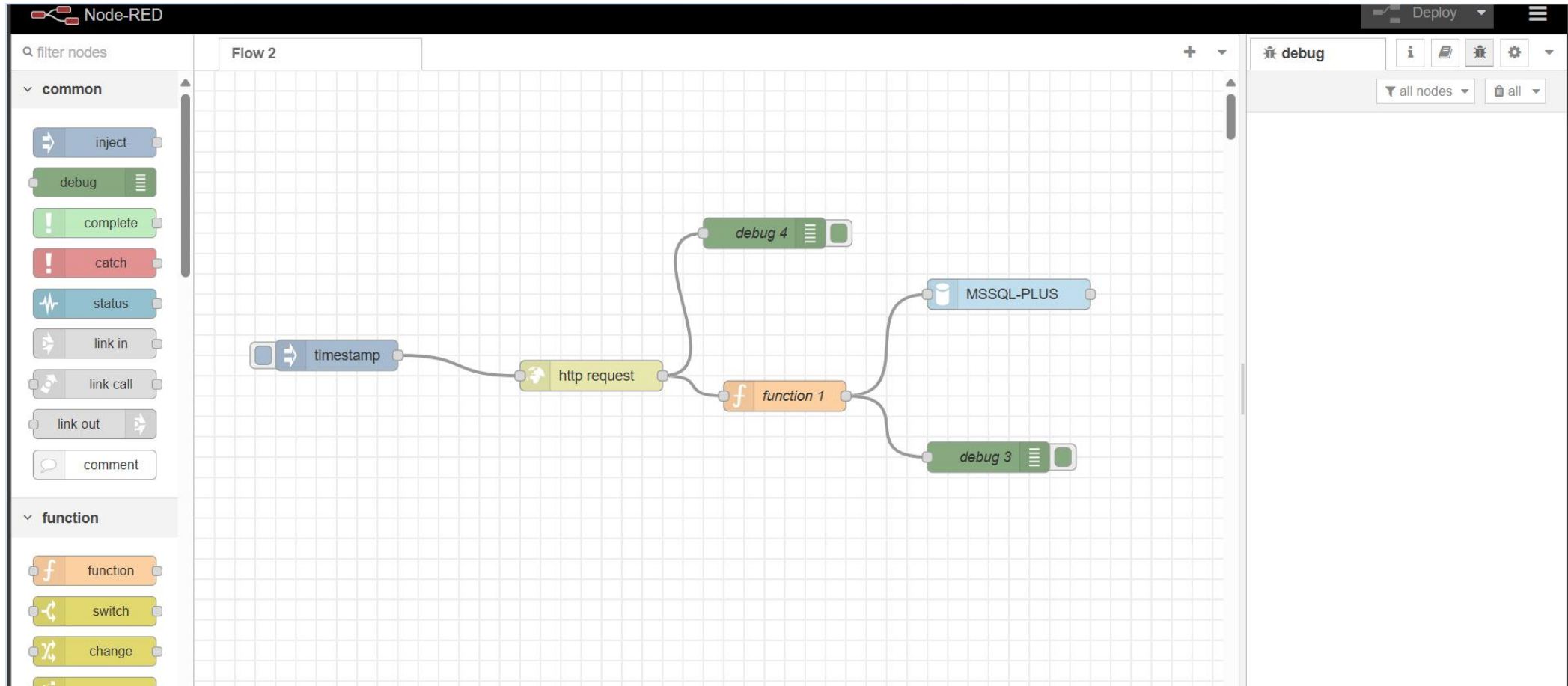
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

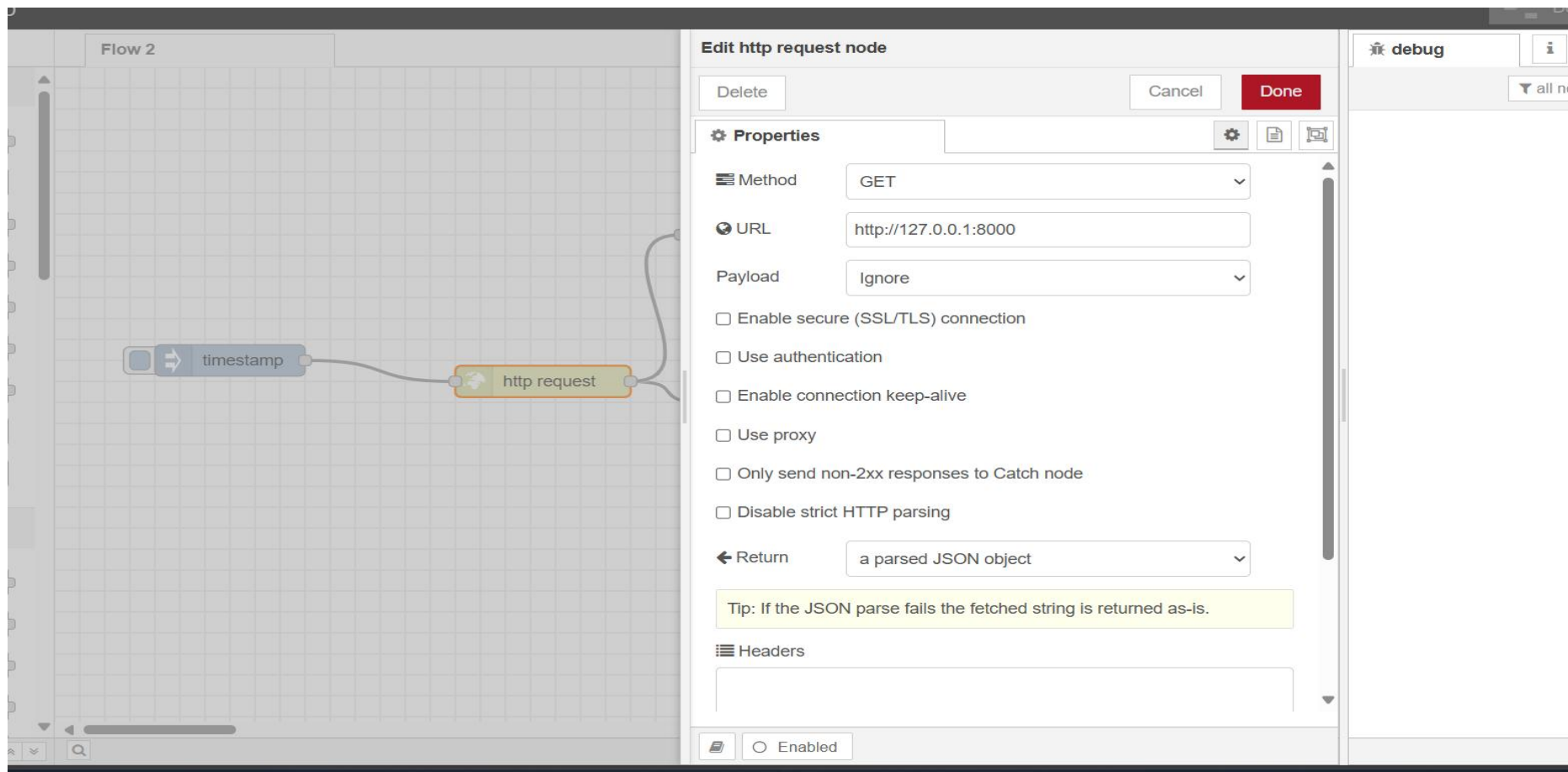
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

28 May 13:49:45 - [info] Server now running at http://127.0.0.1:1880/
28 May 13:49:45 - [info] Starting flows
28 May 13:49:45 - [info] Started flows
```

# Thiết lập các luồng làm việc của node-red



# Gán đường dẫn URL của API đã tạo



# Bước 3: Thêm dữ liệu vào SQL sever

The screenshot displays the Node-RED web interface. On the left, the 'common' nodes palette is visible, containing nodes like inject, debug, complete, catch, status, link in, link call, link out, and comment. The central workspace, labeled 'Flow 2', shows a 'timestamp' node connected to an 'http request' node. On the right, the 'Edit MSSQL node' configuration panel is open. It includes a 'Delete' button, 'Cancel', and 'Done' buttons. The 'Properties' section shows the 'Connection' set to 'NNTHAI\SQLEXPRESS'. The 'Name' field is empty. The 'Query mode' is set to 'Query' and the 'Query' editor is set to 'Editor'. The 'Query' field contains the following SQL statement:

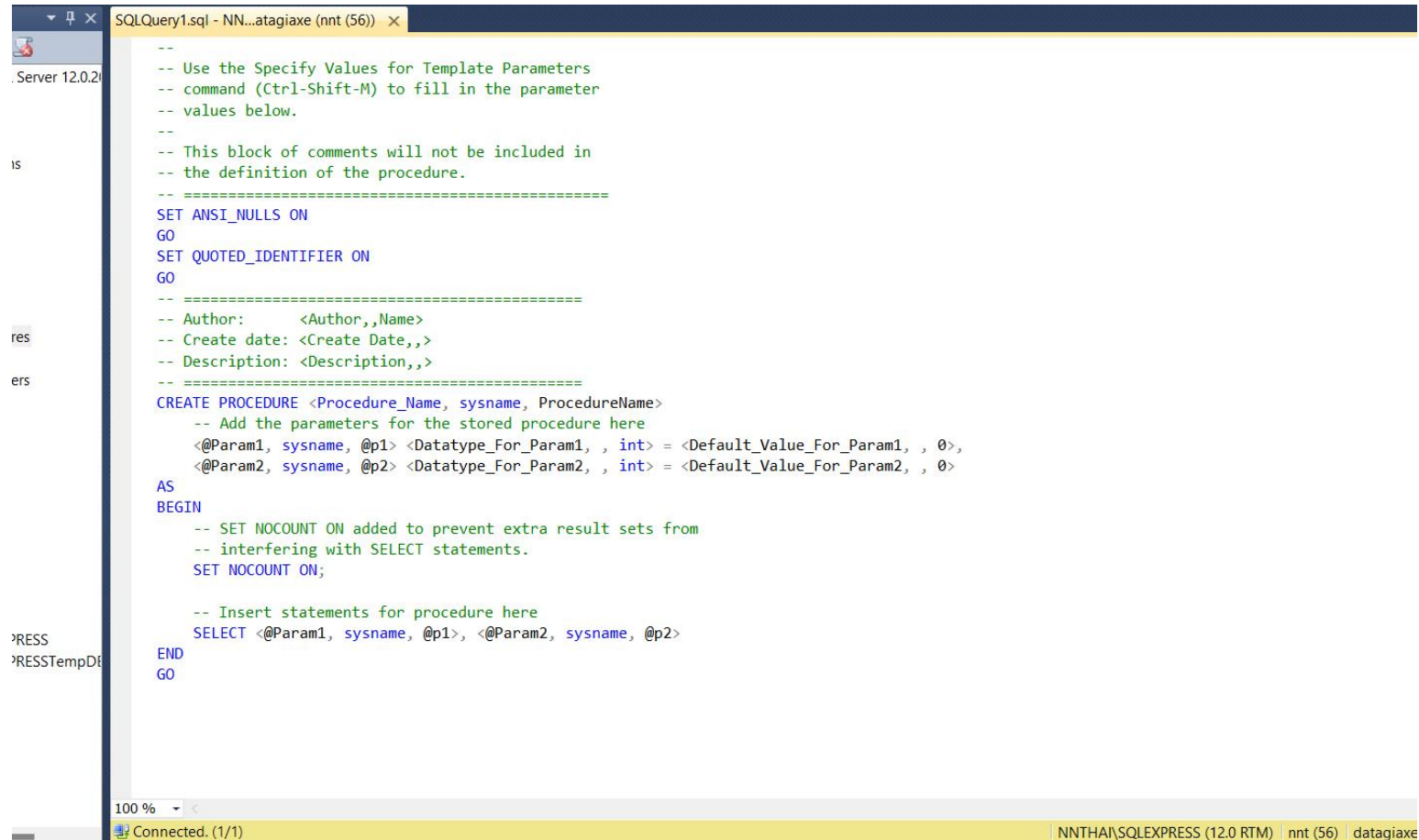
```
1 EXEC SP_giablan @mauxe = @mauxe , @giadexuat= @param2 ,
```

The 'Parameters' section is set to 'Editor' and lists three input parameters:

Input	Name	Type	Value
▼ Input	mauxe	NVarChar	msg.payload.model
▼ Input	param2	NVarChar	msg.payload.recommend
▼ Input	param3	NVarChar	msg.payload.discounted

At the bottom, the 'Parse Mustache' checkbox is checked, and the 'Output property' is set to 'msg.payload'. The rightmost panel shows the 'debug' console with a dropdown menu set to 'all nodes'.

# Lấy dữ liệu từ SQL sever đẩy lên web



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query script for creating a stored procedure. The script includes comments for template parameters and a CREATE PROCEDURE statement with parameters. The bottom pane shows the status bar indicating the connection to the NNTHAI\SQLEXPRESS (12.0 RTM) instance.

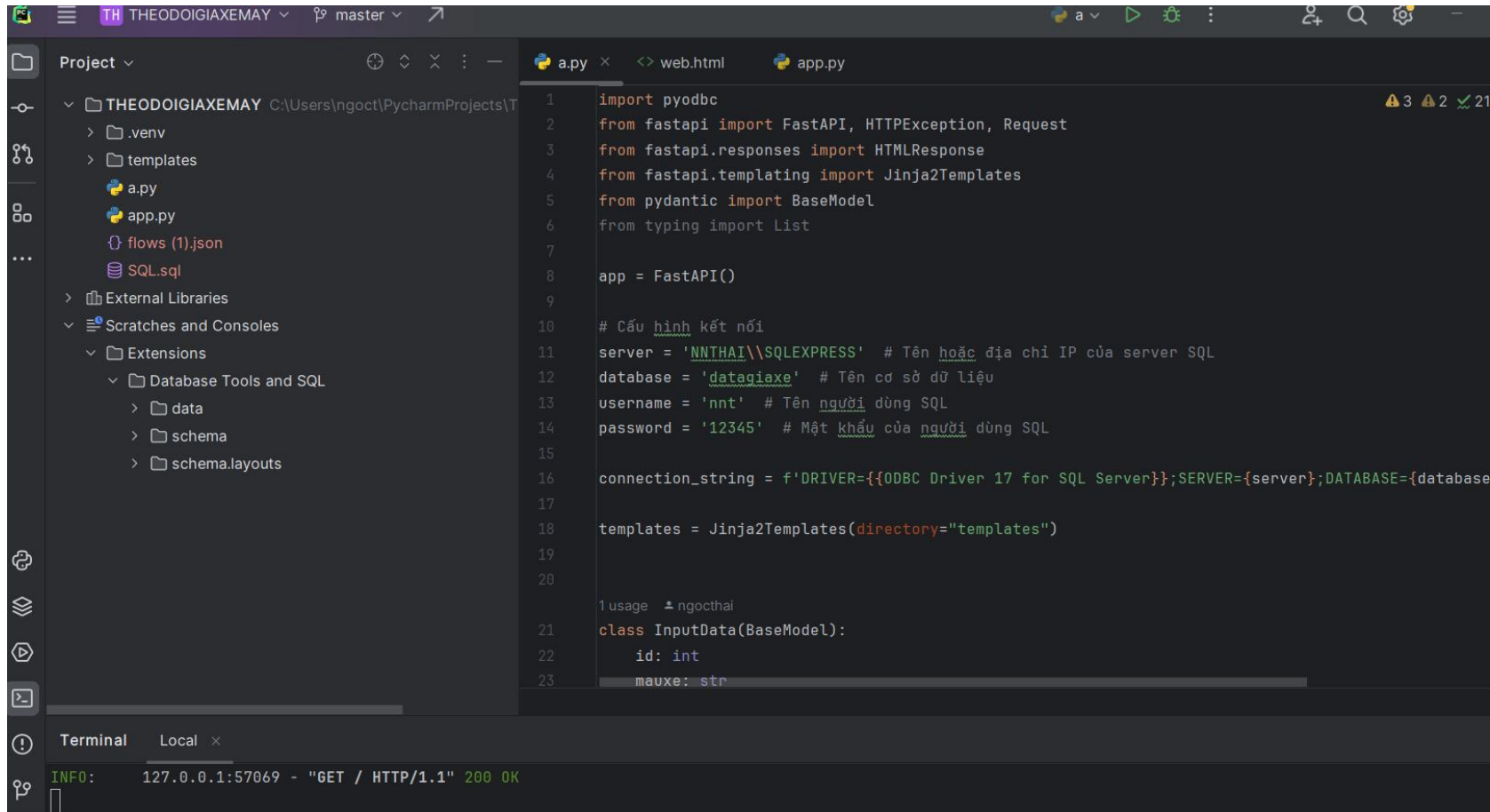
```
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- =====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
-- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

100 %  
Connected. (1/1) | NNTHAI\SQLEXPRESS (12.0 RTM) | nnt (56) | datagiaxe



# Tạo a.py để kết nối với SQL sever



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' sidebar displays the file structure of 'THEODOIGIAXEMAY', including folders like '.venv', 'templates', and files like 'a.py', 'app.py', 'flows (1).json', and 'SQL.sql'. The main editor window shows the code for 'a.py'. The code imports necessary libraries (pyodbc, FastAPI, HTTPException, Request, HTMLResponse, Jinja2Templates, BaseModel, List) and sets up a FastAPI application. It defines database connection parameters (server, database, username, password) and a connection string. A Jinja2Templates object is created for the 'templates' directory. A Pydantic model 'InputData' is defined with fields 'id' (int) and 'mauxe' (str). The bottom terminal window shows a successful GET request to '127.0.0.1:57069' with a '200 OK' status.

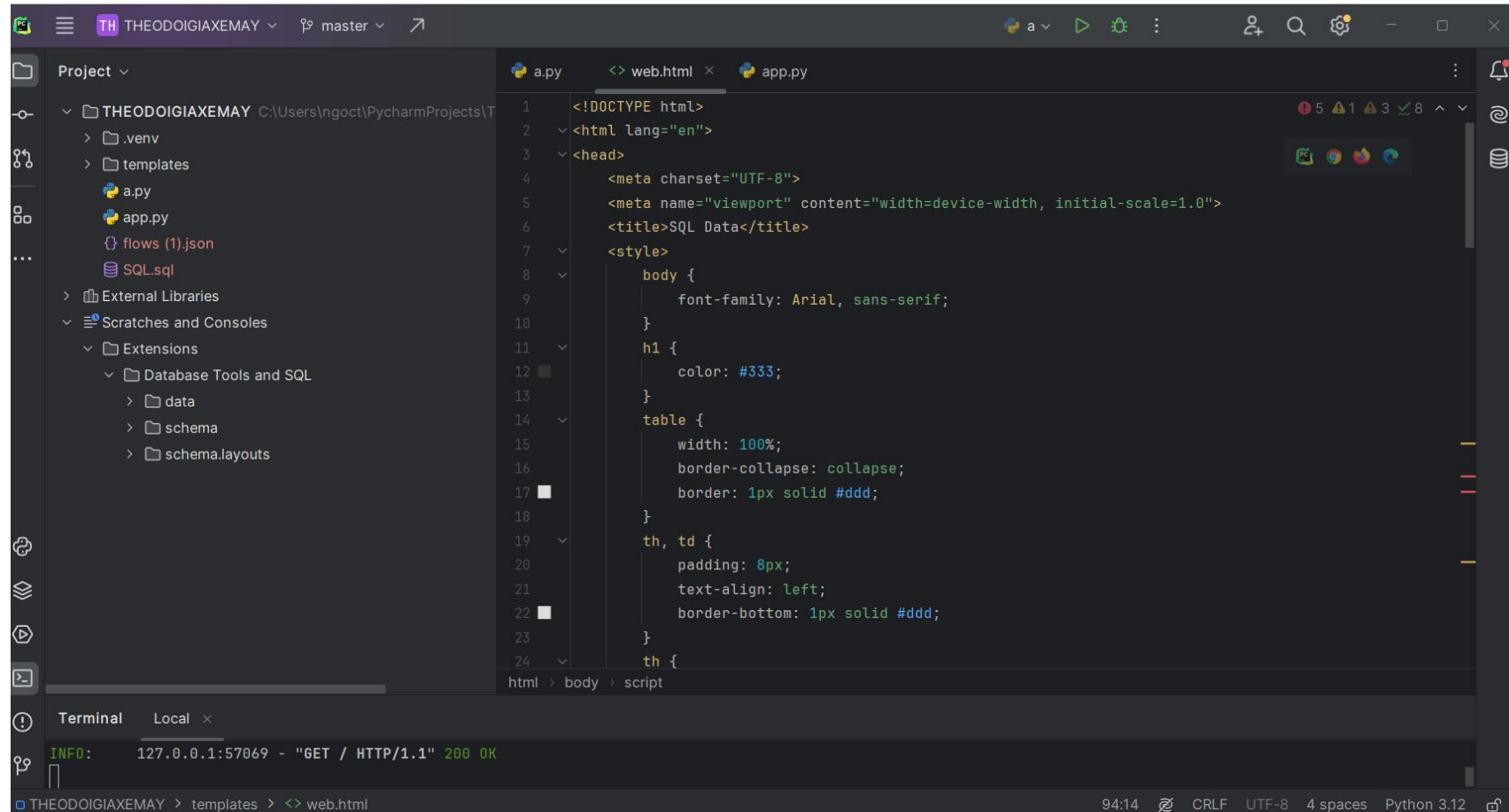
```
1 import pyodbc
2 from fastapi import FastAPI, HTTPException, Request
3 from fastapi.responses import HTMLResponse
4 from fastapi.templating import Jinja2Templates
5 from pydantic import BaseModel
6 from typing import List
7
8 app = FastAPI()
9
10 # Cấu hình kết nối
11 server = 'NNTTHAI\\SQLEXPRESS' # Tên hoặc địa chỉ IP của server SQL
12 database = 'datagiaxe' # Tên cơ sở dữ liệu
13 username = 'nnt' # Tên người dùng SQL
14 password = '12345' # Mật khẩu của người dùng SQL
15
16 connection_string = f'DRIVER={{ODBC Driver 17 for SQL Server}};SERVER={server};DATABASE={database}'
17
18 templates = Jinja2Templates(directory="templates")
19
20
21 usage: ngocthai
22 class InputData(BaseModel):
23     id: int
24     mauxe: str
```

Terminal Local x

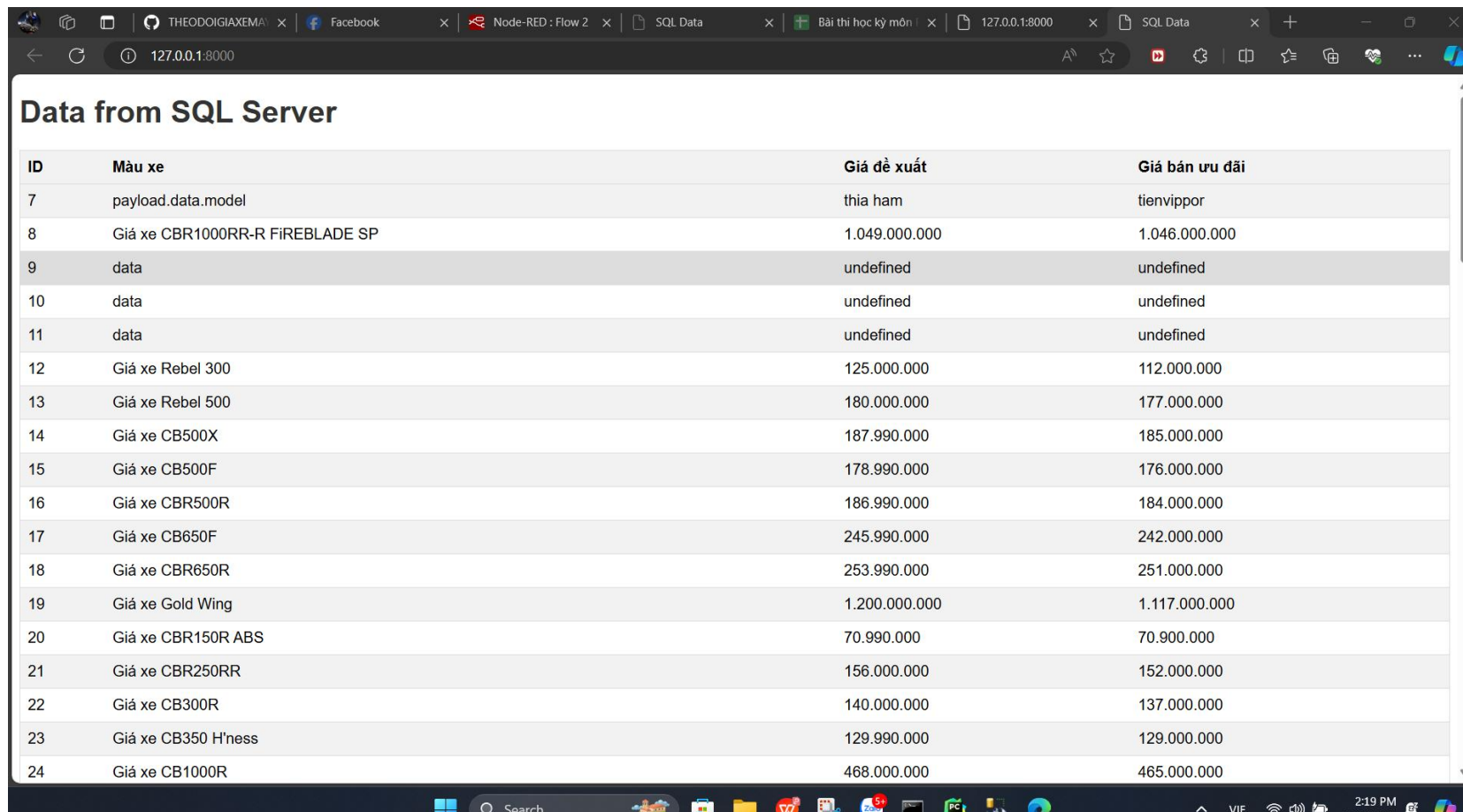
INFO: 127.0.0.1:57069 - "GET / HTTP/1.1" 200 OK



# add 1 file index.html để hiển thị



# Dữ liệu đã được hiển thị



ID	Màu xe	Giá đề xuất	Giá bán ưu đãi
7	payload.data.model	thia ham	tienvippor
8	Giá xe CBR1000RR-R FIREBLADE SP	1.049.000.000	1.046.000.000
9	data	undefined	undefined
10	data	undefined	undefined
11	data	undefined	undefined
12	Giá xe Rebel 300	125.000.000	112.000.000
13	Giá xe Rebel 500	180.000.000	177.000.000
14	Giá xe CB500X	187.990.000	185.000.000
15	Giá xe CB500F	178.990.000	176.000.000
16	Giá xe CBR500R	186.990.000	184.000.000
17	Giá xe CB650F	245.990.000	242.000.000
18	Giá xe CBR650R	253.990.000	251.000.000
19	Giá xe Gold Wing	1.200.000.000	1.117.000.000
20	Giá xe CBR150R ABS	70.990.000	70.900.000
21	Giá xe CBR250RR	156.000.000	152.000.000
22	Giá xe CB300R	140.000.000	137.000.000
23	Giá xe CB350 H'ness	129.990.000	129.000.000
24	Giá xe CB1000R	468.000.000	465.000.000

## Kết luận

Việc Kết hợp ba công nghệ này có thể tạo ra các giải pháp mạnh mẽ và linh hoạt cho nhiều loại dự án. Python có thể được sử dụng để xây dựng các ứng dụng và phân tích dữ liệu phức tạp, trong khi Node-RED giúp đơn giản hóa việc tích hợp các hệ thống và tự động hóa. SQL, với khả năng quản lý dữ liệu mạnh mẽ, sẽ đảm bảo rằng dữ liệu được tổ chức và truy vấn một cách hiệu quả. Sự kết hợp này không chỉ tăng cường khả năng phát triển mà còn mở ra nhiều cơ hội cho sự sáng tạo và tối ưu hóa quy trình làm việc trong các dự án công nghệ hiện đại.