

# BÁO CÁO ANSA TRAINING

05/2023

Nguyễn Trọng Tín – 20214111

## Mục lục

|   |    |
|---|----|
| <i>1. Giới thiệu về Training</i> .....  | 3  |
| <b>1.1. Thông tin cơ bản</b> .....  | 3  |
| <b>1.2. Sơ bộ về sản phẩm</b> .....   | 3  |
| <i>2. Quá trình Training (tuần 1)</i> .....   | 3  |
| <b>2.1. Day 1</b> .....   | 3  |
| <b>2.1.1. Mục tiêu cần đạt được</b> .....   | 3  |
| <b>2.1.2. Kiến thức cần có</b> .....  | 4  |
| a. Command-line Interface .....   | 4  |
| b. Git .....  | 5  |
| c. C# và ASP.Net.....   | 9  |
| <b>2.1.3. Thực hiện</b> .....   | 10 |
| a. Cài đặt Visual Studio Code, DotNet SDK 7, Git.....   | 10 |
| b. In ra màn hình HelloWorld .....  | 12 |
| c. Tạo danh sách HOCSINH.....   | 13 |
| <b>2.2. Day 2</b> .....   | 15 |
| <b>2.2.1. Mục tiêu cần đạt được</b> .....   | 15 |
| <b>2.2.2. Kiến thức cần có</b> .....  | 15 |
| a. HTTP Request và HTTP Response.....   | 15 |
| b. JSON.....  | 17 |
| c. Web Server .....   | 18 |
| <b>2.1.3. Thực hiện</b> .....   | 19 |
| a. Chương trình C#/ASP.NET Web lắng nghe ở localhost:8080, khi truy cập<br>trả về “Hello world” ..... | 19 |

b. Chương trình C#/ASP.NET Web lắng nghe ở localhost:8080, có các API (không yêu cầu làm việc với database).....20

**2.3. Day 3.....26**

**2.3.1. Mục tiêu cần đạt được.....26**

**2.3.2. Kiến thức cần có .....26**

a. SQL .....26

b. Mô hình MVC .....27

**2.3.3. Thực hiện.....28**

a. Database với danh sách HOCSINH.....28

b. Mô hình MVC với danh sách HOCSINH .....33

**3. Code tham khảo .....40**

1. Giới thiệu Training:

*1.1. Thông tin cơ bản:*

- Thời gian thực hiện: 2 tuần.
- Số thành viên thực hiện: 8.
- Mục tiêu: Nắm kiến thức cơ bản về mô hình client-server và OAuth2.
- Sản phẩm: trang web chia sẻ file tạm thời (backend only).
- Yêu cầu: Sử dụng ASP.NET Core (C#), Python, GitHub, Visual Studio CODE.

*1.2. Sơ bộ về sản phẩm:*

Trang web có công dụng lưu trữ tạm thời các tệp tin để người dùng có thể truy cập mà không cần đăng nhập.

Trang web chia sẻ file gồm 2 server nhỏ:

- Server Xác thực (Authentication Server): thực hiện xác thực theo OAuth2.
- Server Chính (Resource Server): thực hiện mọi chức năng của web.
- Front-end: Giao diện người dùng

Kế hoạch ưu tiên giúp các thành viên nắm vững các kiến thức cơ bản và hoàn thành server chính (Resource Server). Server xác thực (Authentication Server) sẽ để dành vào thời gian khác. Phần front-end sẽ để dành cho từng thành viên tự nghiên cứu và thực hiện.

\* Server Xác thực (Authentication Server)

Thực hiện xác thực theo OAuth2.

\* Server Chính (Resource Server)

Thực hiện xử lý nghiệp vụ.

- Xử lý user: đăng kí mới, xoá tài khoản, thay đổi tài khoản.
- Xử lý file: tải file lên, xoá file, sửa thông tin file, đọc file

## 2. Quá trình Training (tuần 1)

### *2.1. Day 1:*

#### **2.1.1. Mục tiêu cần đạt được:**

- Làm quen với command-line;
- Làm quen với git (init, clone, add, remove, reset, branch, push, pull);
- Xử lý git conflicts;
- GitHub và pull requests;
- Tìm hiểu về C# và ASP.NET;
- Cài đặt Visual Studio Code, DotNet SDK 7, Git;
- Viết chương trình C# cơ bản:
  - + In ra màn hình “Hello world”;
  - + Quản lý danh sách HOCSINH.

#### **2.1.2. Kiến thức cần có:**

##### a. Command-line Interface – CLI (Giao diện dòng lệnh):

- Là phương tiện tương tác với chương trình máy tính nơi người dùng đưa ra lệnh cho chương trình dưới dạng các dòng lệnh liên tiếp.
- Một số giao diện dòng lệnh thường dùng: Windows PowerShell (PS), Windows Command Prompts (CMD), Linux Bash,...

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The command executed is "Get-Service -ComputerName Jesusa5b1 | Where-Object {\$\_.name -Like 'W\*'}". The output is a table with three columns: Status, Name, and DisplayName. The table lists various Windows services, including W32Time, wassvc, WcsPlugInService, WdiServiceHost, WdiSystemHost, Wecsvc, wercplsupport, WerSvc, WinHttpAutoProx..., Winmgmt, WinRM, wmiApSrv, WPDBusEnum, WSService, wuauserv, and wudfsvc. The status of each service is indicated in the first column.

| Status  | Name               | DisplayName                            |
|---------|--------------------|--|
| Running | W32Time            | Windows Time                           |
| Running | wassvc             | Windows Assessment Services Service    |
| Stopped | WcsPlugInService   | Windows Color System                   |
| Stopped | WdiServiceHost     | Diagnostic Service Host                |
| Stopped | WdiSystemHost      | Diagnostic System Host                 |
| Stopped | Wecsvc             | Windows Event Collector                |
| Stopped | wercplsupport      | Problem Reports and Solutions Contr... |
| Stopped | WerSvc             | Windows Error Reporting Service        |
| Stopped | WinHttpAutoProx... | WinHTTP Web Proxy Auto-Discovery Se... |
| Running | Winmgmt            | Windows Management Instrumentation     |
| Running | WinRM              | Windows Remote Management (WS-Manag... |
| Stopped | wmiApSrv           | WMI Performance Adapter                |
| Stopped | WPDBusEnum         | Portable Device Enumerator Service     |
| Stopped | WSService          | Windows Store Service (WSService)      |
| Stopped | wuauserv           | Windows Update                         |
| Stopped | wudfsvc            | Windows Driver Foundation - User-mo... |

Hình 2.1.2.a – Windows PowerShell

- Ưu điểm:

- + Có thể kiểm soát chi tiết một hệ điều hành hoặc ứng dụng;
- + Quản lý nhanh hơn số lượng lớn các hệ điều hành;
- + Có khả năng lưu trữ các tập lệnh để tự động hoá các tác vụ thường xuyên;
- + Kiến thức cơ bản về giao diện dòng lệnh giúp khắc phục sự cố, chẳng hạn như sự cố kết nối mạng

- Vì sao cần có CLI?

- + Không phải phần mềm nào cũng có GUI để dùng.
- + Không phải hệ điều hành nào cũng hỗ trợ GUI.
- + CLI đơn giản hơn GUI.

- Tại sao nên sử dụng CLI hơn là GUI?

- + CLI tiêu tốn ít tài nguyên hơn.
- + Độ chính xác cao.
- + Nhiệm vụ lặp đi lặp lại thân thiện.
- + Mạnh mẽ.
- + Tiết kiệm nhiều thời gian nếu thành thạo.
- + Hầu hết các tool cho developer đều là CLI.
- + Developer có thể sẽ phải làm việc nhiều với các server Linux.

- Một số câu lệnh (PowerShell):

| Mục đích                             | Command                     |
|--------------------------------------|-----------------------------|
| Xem vị trí thư mục làm việc hiện tại | pwd                         |
| Đổi vị trí thư mục làm việc          | cd {{đường dẫn}}            |
| Tạo thư mục mới                      | mkdir {{tên thư mục mới}}   |
| Xoá thư mục                          | rm -Recurse {{tên thư mục}} |
| Xoá file                             | rm {{tên file}}             |
| Xem nội dung thư mục                 | ls                          |
| Xoá mọi nội dung trên màn hình       | clear                       |

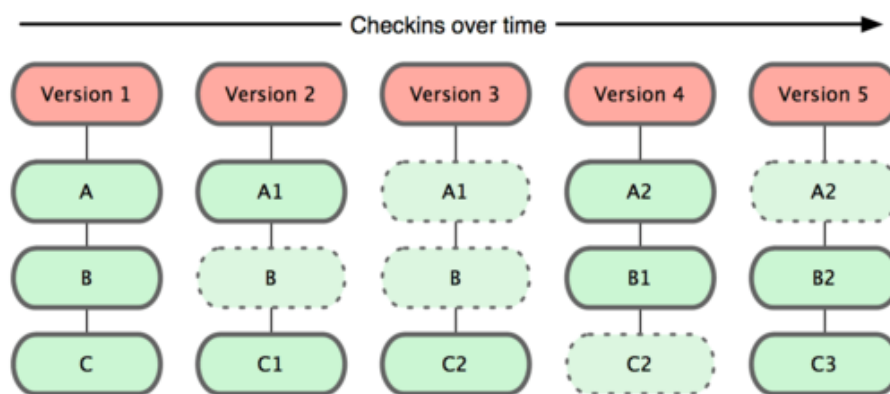
b. Git:

- Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository) riêng chứa toàn bộ lịch sử thay đổi.

- VCS là viết tắt của Version Control System là hệ thống kiểm soát các phiên bản phân tán mã nguồn mở. Các VCS sẽ lưu trữ tất cả các file trong toàn bộ dự án và ghi lại toàn bộ lịch sử thay đổi của file. Mỗi sự thay đổi được lưu lại sẽ được và thành một version (phiên bản).

- Cách hoạt động của Git: Git coi thông tin được lưu trữ là một tập hợp các snapshot – ảnh chụp toàn bộ nội dung tất cả các file tại thời điểm.

Mỗi khi “commit”, Git sẽ “chụp” và tạo ra một snapshot cùng một tham chiếu tới snapshot đó. Để hiệu quả, nếu các tệp không thay đổi, Git sẽ không lưu trữ lại file — chỉ là một liên kết đến tệp giống file trước đó mà nó đã lưu trữ.



Hình 2.1.2.b1 – Nguyên lý hoạt động của Git

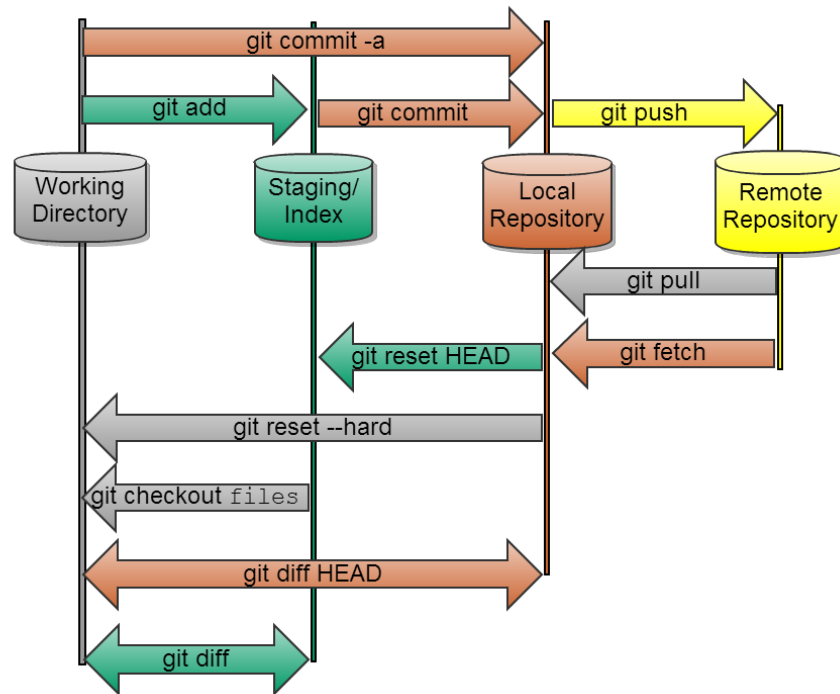
- Lợi ích:

+ Dễ sử dụng, thao tác nhanh, gọn, lẹ và rất an toàn.

- + Dễ dàng kết hợp các phân nhánh (branch), có thể giúp quy trình làm việc code theo nhóm đơn giản hơn rất nhiều.
- + Chỉ cần clone mã nguồn từ kho chứa hoặc clone một phiên bản thay đổi nào đó từ kho chứa, hoặc một nhánh nào đó từ kho chứa là bạn có thể làm việc ở mọi lúc mọi nơi.
- + Deployment sản phẩm một cách không thể nào dễ dàng hơn.
- Một số câu lệnh của Git

| Mục đích                                       | Command                                     |
|--|---|
| Khởi tạo thư mục                               | git init                                    |
| Copy code từ thư mục Git khác                  | git clone {{repository}} -o {{folder name}} |
| Thêm file cần commit                           | git add {{file}}                            |
| Loại bỏ file ra khỏi danh sách cần commit      | git remove {{file}}                         |
| Reset danh sách cần commit                     | git reset                                   |
| Commit   | git commit -m {{context}}                   |
| Xem các remote                                 | git remote                                  |
| Thêm remote                                    | git remote add {{remote name}} {{path}}     |
| Xoá remote                                     | git remote remove {{remote name}}           |
| Đẩy code lên remote                            | git push {{remote name}} {{branch name}}    |
| Xem các nhánh và nhánh hiện tại                | git branch                                  |
| Thêm nhánh                                     | git branch {{branch name}}                  |
| Chuyển nhánh                                   | git checkout {{branch name}}                |
| Cập nhật nhánh với remote                      | git fetch {{remote name}} {{branch name}}   |
| Cập nhật code từ nhánh khác vào nhánh hiện tại | git rebase {{branch name}}                  |
| Lấy code từ remote                             | git pull {{remote name}}                    |

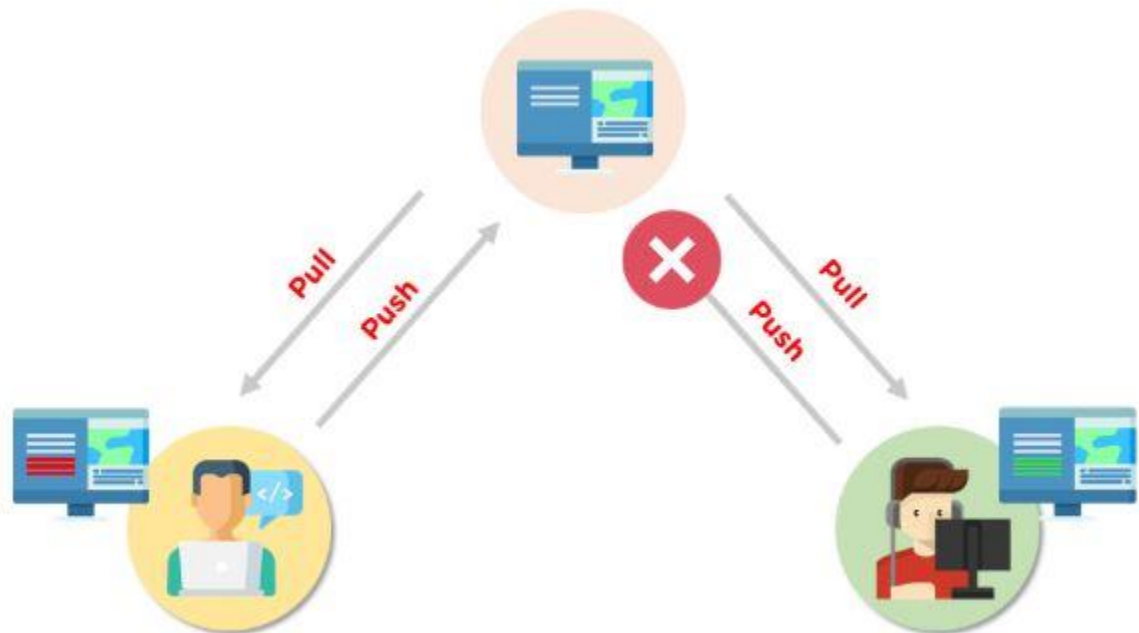
## Git Workflow & Commands



Hình 2.1.2.b2 – Một số câu lệnh của Git

### \* Resolve Merge Conflict:

- Merge Conflict xảy ra khi Git không thể ngay lập tức giải quyết sự khác biệt giữa các code giữa 2 commit. Git có thể merge sự thay đổi này tự động khi các commit này ở các line hoặc branch.



Hình 2.1.2.b3 – Git Merge Conflict.

- Các loại Merge Conflict:

- + Bắt đầu quá trình Merge: Nếu có những thay đổi tại Working Directory trong project hiện tại thì merge sẽ không diễn ra.

Trong trường hợp này, conflict xảy ra do các thay đổi đang chờ xử lý cần được ổn định bằng các lệnh Git khác nhau.

- + Trong quá trình Merge:

Lỗi trong quá trình hợp nhất chỉ ra rằng có xung đột giữa local branch và branch được hợp nhất.

Trong trường hợp này, Git giải quyết hết mức có thể, nhưng có những thứ phải giải quyết thủ công trong các tệp bị conflict.

- Làm thế nào để Resolve Conflict:

- Cách dễ nhất để giải quyết tệp bị conflict là mở tệp đó và thực hiện bất kỳ thay đổi cần thiết nào.

- Sau khi chỉnh sửa tệp, chúng ta có thể sử dụng lệnh git add để tạo nội dung mới được hợp nhất.

- Bước cuối cùng là tạo commit mới với sự trợ giúp của lệnh git commit.

- Git sẽ tạo một merge commit mới để hoàn tất quá trình merge.

- \* GitHub và Pull request:

- GitHub là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. Github có đầy đủ những tính năng của Git, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

- Pull request (PR) được tạo ra để gộp code mới vào trong mã nguồn cũ (merge source) giúp cho mọi người có thể cùng truy cập và review những file source code đó khi các tính năng mới hoàn thành. Đây cũng là một thông báo cho những người làm chung là bạn đã hoàn thành phần việc của mình sẵn sàng để bổ sung tính năng mới vào sản phẩm.

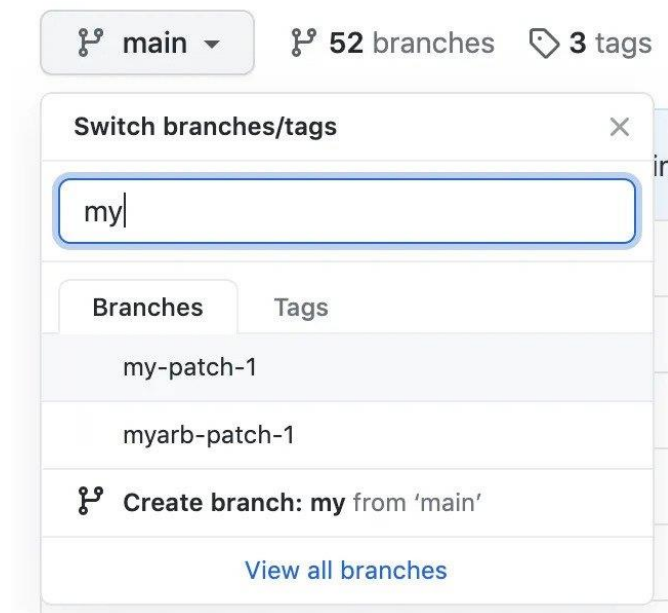
- Cách tạo Pull Request trên GitHub:

- + Thay đổi branch range và destination repository.

- + Trên GitHub.com, tìm đến trang chính của repository.

- + Trong menu "Branch", chọn branch chứa commit của mình.





*Hình 2.1.2.b4 – Menu Branch*

+ Phía trên danh sách tệp, trong biểu ngữ màu vàng, nhấp vào Compare & Pull Request để tạo Pull Request cho branch được liên kết.



*Hình 2.1.2.b5 – Compare & Pull Request*

+ Sử dụng menu nhánh để chọn nhánh mình muốn merge các thay đổi của mình vào, sau đó sử dụng menu compare branch để chọn branch mà bạn đã thực hiện thay đổi.

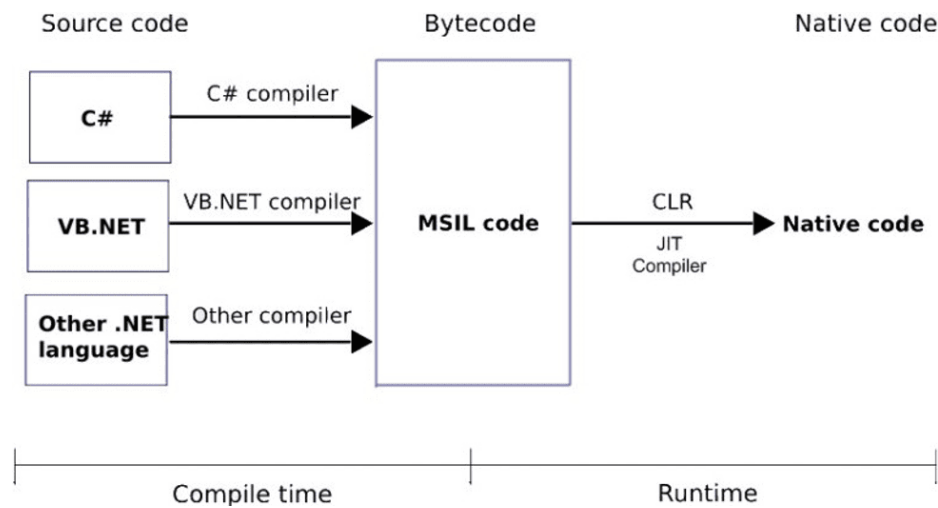
+ Nhập title và description cho pull request của mình.

+ Để tạo pull request đã sẵn sàng để review, hãy nhấp vào Create Pull Request. Để tạo draft pull, hãy sử dụng drop-down và chọn Draft Pull Request, sau đó nhấp vào Draft Pull Request.

c. C# và ASP.Net:

\* C#:

- C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000. C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.



Hình 2.1.2.c1 – Nguyên lý hoạt động của C#

- Đặc trưng:

- + C# là ngôn ngữ đơn giản.
- + C# là ngôn ngữ hiện đại.
- + C# là một ngôn ngữ lập trình thuần hướng đối tượng.
- + C# là một ngôn ngữ ít từ khóa.

\* ASP.Net:

- ASP.Net là một nền tảng dành cho phát triển web, được Microsoft phát hành và cung cấp lần đầu tiên vào năm 2002. Nền tảng được sử dụng để tạo ra các ứng dụng web-based.

- Cấu trúc và các thành phần của ASP.Net:

Cấu trúc của .Net framework dựa trên các thành phần cơ bản sau:

- + Ngôn ngữ (Language)
- + Thư viện (Library)
- + Thời gian chạy ngôn ngữ thông thường (Common Language Runtime – CLR)

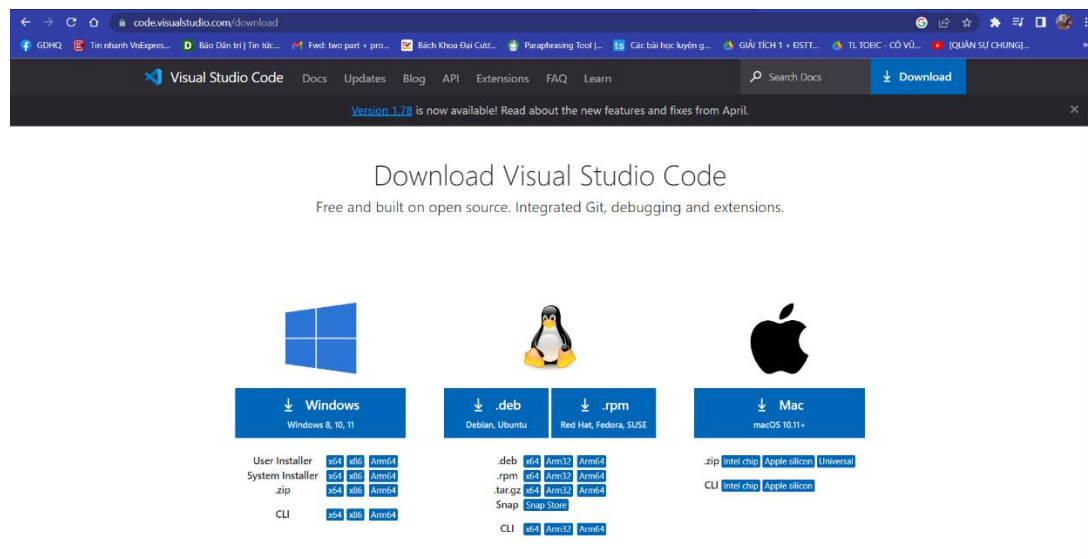
Một số đặc điểm cơ bản của ASP.Net framework:

- + Trạng thái Code rời (Code Behind Mode)
- + Quản lý trạng thái (State Management)
- + Bộ nhớ Cache (Caching)

## 2.1.3 Thực hiện

- a. Cài đặt Visual Studio Code, DotNet SDK 7, Git

\* Visual Studio Code: Tải qua đường link <https://code.visualstudio.com/download>



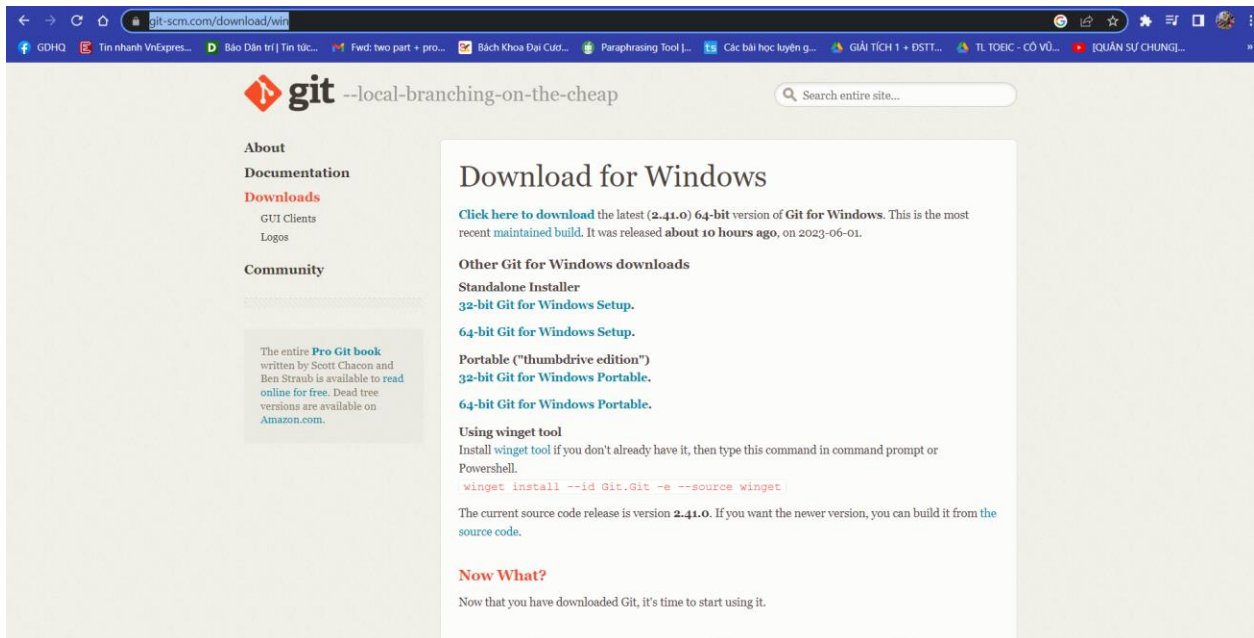
Hình 2.1.3.a1 – Client Download của VSC

\* DotNet SDK 7: Tải qua đường link <https://dotnet.microsoft.com/en-us/download/dotnet/7.0>



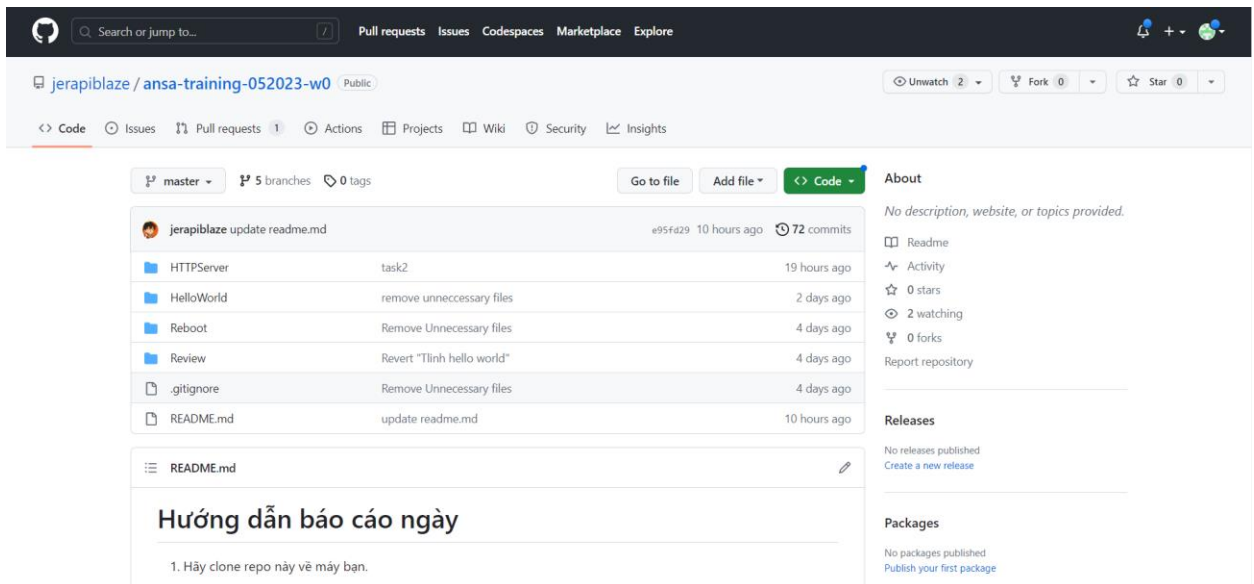
Hình 2.1.3.a2 – Client Download của .Net 7.0

\* Git: Tải qua đường link <https://git-scm.com/download/win>



Hình 2.1.3.a3 – Client Download của Git

b. In ra màn hình HelloWorld:



Hình 2.1.3.b1 – Repos ANSA Training

b. In ra màn hình Hello World:

B0: Add remote về máy: git remote add origin <https://github.com/jerapiblaze/ansa-training-052023-w0>

B1: Clone repo về máy: git clone <https://github.com/jerapiblaze/ansa-training-052023-w0>

B2: Chuyển thư mục làm việc vào thư mục git vừa clone: cd ansa-training-052023-w0

B3: Chuyển thư mục làm việc đến thư mục HelloWorld: cd HelloWorld

B4: Tạo thư mục tên mình và chuyển sang thư mục mới tạo: mkdir Trong Tin  
cd TrongTin

B5: Tạo nhánh mới theo form: git branch TrongTin\_HelloWorld

B6: Chuyển sang nhánh mới tạo: git checkout TrongTin\_HelloWorld

B7: Tạo thư mục Task01: mkdir Task01

B8: Chuyển thư mục làm việc về Task01: cd Task01

B9: Khởi tạo project ASP.NET Console mới: dotnet new console. Tại bước này thì Program.cs đã có sẵn lệnh in “Hello World” nên có thể chạy luôn.

B10: Chạy chương trình: dotnet run.

### c. Tạo danh sách HOCSINH:

- Sau khi hoàn thành xong Task01:

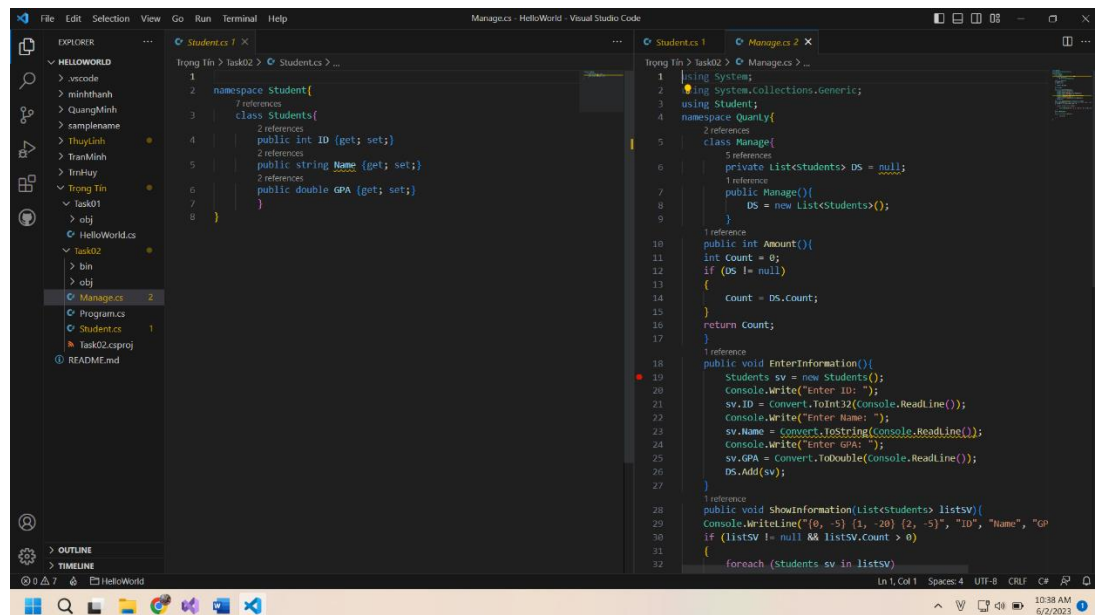
B1: Trở lại thư mục gốc: cd ..

B2: Tạo ra thư mục Task02: mkdir Task02

B3: Chuyển thư mục làm việc vào Task02: cd Task 02:

B4: Khởi tạo project ASP.NET Console mới: dotnet new console.

B5: Chuẩn bị code:

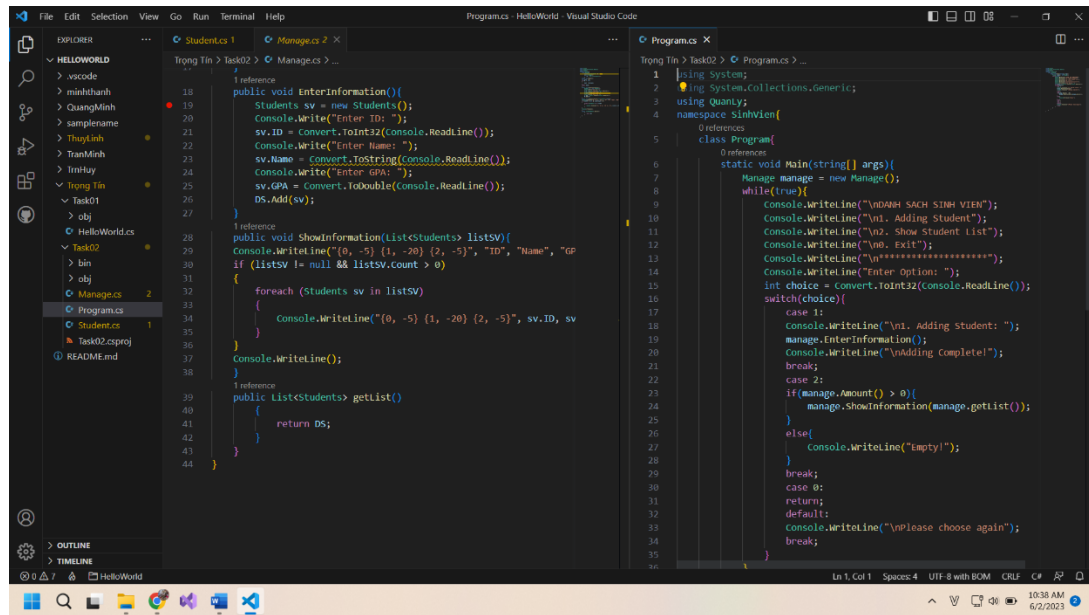


Hình 2.1.3.c1 – Student.cs và Manage.cs

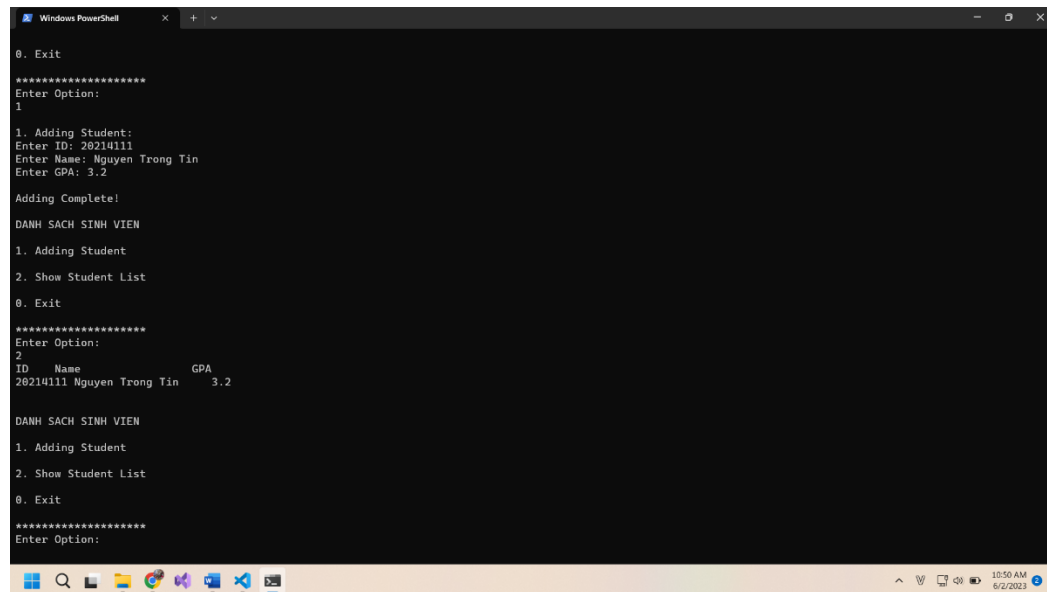
- Student.cs: Khai báo các biến trong Class Student: ID, name, GPA.

- Manage.cs: Khai báo các hàm để thực hiện các chức năng như Nhập thông tin sinh viên, Xem danh sách sinh viên

- Program.cs: Code chính để thực hiện các hàm đã khai báo trong Manage.cs



Hình 2.1.3.c2 – Program.cs và Manage.cs



Hình 2.1.3.c3 – Chạy thử chương trình

B6: Commit code:

cd ..

git add .

git commit -m “Pull Request”

B7: Push code lên: git push origin TrongTin\_HelloWorld

B8: Vào repo trên GitHub, vào mục Pull Requests và chọn New Pull Request

B9: Qua mục base, chọn master. Mục compare, chọn nhánh TrongTin\_HelloWorld. Điền nốt thông tin và chọn Create.

\*Trong trường hợp gặp Conflict:

B1: Pull code từ remote về Local: git pull origin TrongTin\_HelloWorld

B2: Vào phần Git của VSCode, ta sẽ thấy một danh sách các file có conflict. Xử lý lần lượt từng file.

B3: Chọn File cần xử lý và chọn Resolve in Merge Editor

B4: Sau khi hoàn thành các thay đổi, nhập Complete Merge.

B5: Commit lại và push code lên.

## 2.2. Day 2

### 2.2.1 Mục tiêu:

- Hiểu biết cơ bản về HTTP Request, HTTP Response
- Hiểu biết cơ bản về JSON
- Hiểu biết cơ bản về Web Server
- Làm quen với ASP.NET Core Web Server

### 2.2.2 Kiến thức cần có:







a. HTTP Request và HTTP Response:

\*HTTP Request:

- HTTP request là thông tin được gửi từ client lên server, để yêu cầu server tìm hoặc xử lý một số thông tin, dữ liệu mà client muốn. HTTP request có thể là một file text dưới dạng XML hoặc Json mà cả hai đều có thể hiểu được.

- Các phương thức của HTTP Request:

# HTTP Request Methods

|  |   |  |   |  |   |
|--|---|--|---|--|---|
|  <b>GET</b> |  <b>POST</b> |  <b>PUT</b> |  <b>DELETE</b> |  <b>PATCH</b> |  <b>HEAD</b> |
| retrieve data from server  | add data to an existing file or resource  | update(replace) an existing file or resource in server                                       | delete data from server   | update a resource partially (modify)   | retrieve the resource's headers   |

- **CONNECT** is used to open a two-way socket connection to the remote server;
- **OPTIONS** is used to describe the communication options for specified resource;
- **TRACE** is designed for diagnostic purposes during the development.
- **HEAD** retrieves the resource's headers, without the resource itself.

*Hình 2.2.2.a1 – Các phương thức của HTTP Request*

- + **GET**: là phương thức được Client gửi dữ liệu lên Server thông qua đường dẫn URL nằm trên thanh địa chỉ của Browser. Server sẽ nhận đường dẫn đó và phân tích trả về kết quả cho bạn.
- + **POST**: là phương thức gửi dữ liệu đến server giúp bạn có thể thêm mới dữ liệu hoặc cập nhật dữ liệu đã có vào database.
- + **PUT**: Cách hoạt động tương tự như Post nhưng nó chỉ được sử dụng để cập nhật dữ liệu đã có trong database.
- + **PATCH**: Tương tự như Post và Put, nhưng Patch được sử dụng khi phải cập nhật một phần dữ liệu của đối tượng.
- + **DELETE**: Xóa các dữ liệu của server về tài nguyên thông qua URI.
- + **HEAD**: Gần giống với GET, tuy nhiên nó không có response body.
- Cấu trúc của HTTP Request:
  - + Request Line
  - + Request Header
  - + Request Body.
- \* HTTP Response:
  - HTTP Response là kết quả server trả về cho Client.
  - Cấu trúc:
 

```
<http-version> <status> <reason-phrase>
```



<headers>

<body>

+ Giải thích:

HTTP-version: phiên bản HTTP cao nhất mà server hỗ trợ.

Status-Code: mã kết quả trả về.

Reason-Phrase: mô tả về Status-Code.

- Ý nghĩa một số Status-Code: Là một số nguyên 3 ký tự. Ký tự đầu tiên của mã hoá trạng thái định nghĩa loại phản hồi và 2 ký tự cuối không có bất cứ vai trò phân loại nào. Có 5 giá trị của ký tự đầu tiên:

- **1xx**: Thông tin. Mã này nghĩa là yêu cầu đã được nhận và tiến trình đang tiếp tục.
- **2xx**: Thành công. Mã này nghĩa là hoạt động đã được nhận, được hiểu, và được chấp nhận một cách thành công.
- **3xx**: Sự điều hướng lại. Mã này nghĩa là hoạt động phải được thực hiện để hoàn thành yêu cầu.
- **4xx**: Lỗi Client. Mã này nghĩa là yêu cầu chứa cú pháp không chính xác hoặc không được thực hiện.
- **5xx**: Lỗi Server. Mã này nghĩa là Server thất bại với việc thực hiện một yêu cầu nhìn như có vẻ khả thi.

b. JSON:

- JSON là viết tắt của **J**ava**S**cript **O**bject **N**otation, là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON là một tiêu chuẩn mở để trao đổi dữ liệu trên web.

- Định dạng JSON sử dụng các cặp *key – value* để dữ liệu sử dụng. Nó hỗ trợ các cấu trúc dữ liệu như đối tượng và mảng.

```
1 [
2   {
3     "first_name": "John",
4     "last_name": "Doe",
5     "gender": "Male",
6     "email": "johndoe@gmail.com"
7   },
8   {
9     "first_name": "John",
10    "last_name": "Wick",
11    "gender": "Male",
12    "email": "johnwick@gmail.com"
13  }
14 ]
15
```

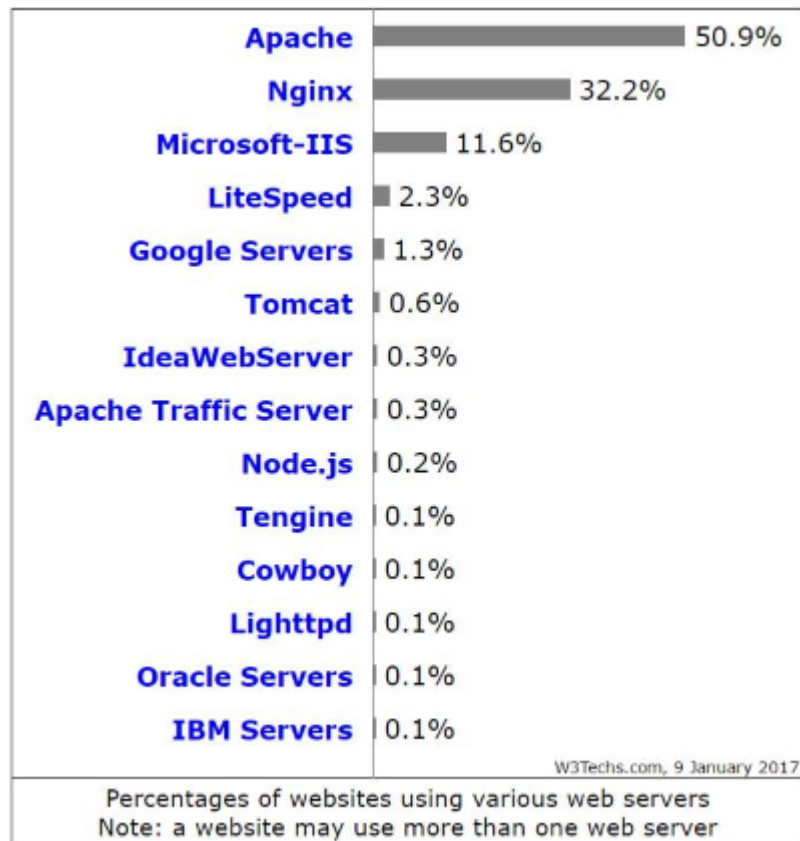
*Hình 2.2.2.b1 – Định dạng JSON*

- JSON sử dụng khi nào?

Đó là khi bạn muốn lưu trữ dữ liệu đơn thuần dưới dạng metadata (dữ liệu về dữ liệu) ở phía server. Chuỗi JSON sẽ được lưu vào database và sau đó khi cần dữ liệu thì sẽ được giải mã.

c. Web Server:

- Web server là máy chủ cài đặt các chương trình phục vụ các ứng dụng web. Web Server có khả năng tiếp nhận request từ các trình duyệt web và gửi phản hồi đến client thông qua giao thức HTTP hoặc các giao thức khác.



*Hình 2.2.2.c1 – Các Web Server thông dụng nhất hiện nay*

- Cách hoạt động của Web Server:

- + Trình duyệt phân giải tên miền thành địa chỉ IP.
- + Web Server gửi lại client Trang được yêu cầu.
- + Trình duyệt hiển thị trang web.

### 2.2.3. Thực hiện:

a. Chương trình C#/ASP.NET Web lắng nghe ở localhost:8080, khi truy cập trả về “Hello world”

- Trước khi thực hiện, cần thực hiện đồng bộ local repo với remote repo

```
git fetch --prune origin
```

```
git reset --hard origin/master
```

```
git clean -f -d
```

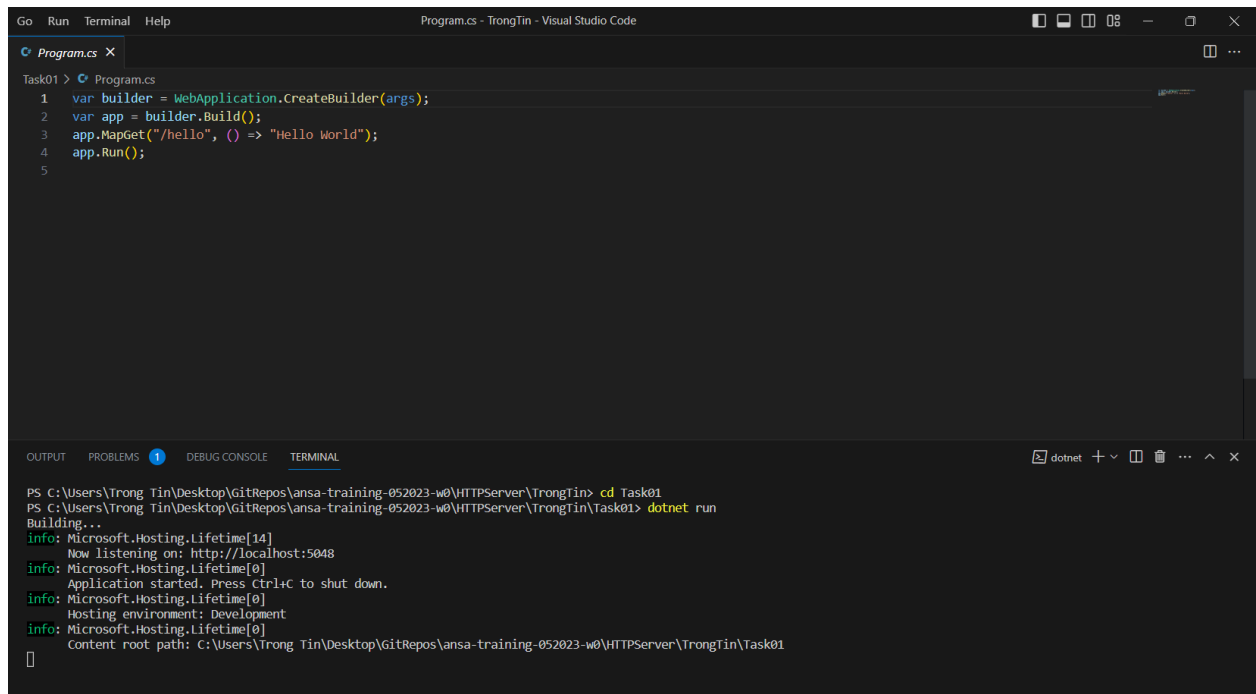
Sau đó, thực hiện tạo branch mới như đã thực hiện trong bài “Hello World”

- Để tạo một project Web Server, thực hiện lệnh sau:

```
dotnet new web
```

- Để chạy thử, thực hiện lệnh:

## dotnet run



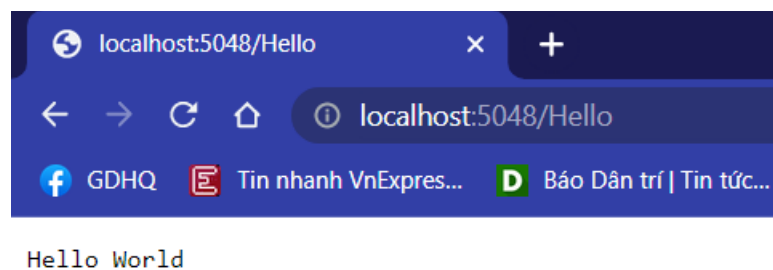
The screenshot shows the Visual Studio Code interface. The top editor pane displays a C# file named `Program.cs` with the following code:

```
1 var builder = WebApplication.CreateBuilder(args);
2 var app = builder.Build();
3 app.MapGet("/hello", () => "Hello World");
4 app.Run();
5
```

The bottom pane shows the `TERMINAL` output, which includes the command `dotnet run` and the following log messages:

```
PS C:\Users\Trong Tin\Desktop\GitRepos\ansa-training-052023-w0\HTTPServer\TrongTin> cd Task01
PS C:\Users\Trong Tin\Desktop\GitRepos\ansa-training-052023-w0\HTTPServer\TrongTin\Task01> dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5048
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Trong Tin\Desktop\GitRepos\ansa-training-052023-w0\HTTPServer\TrongTin\Task01
```

*Hình 2.2.3.a1 – Chạy dotnet run*



*Hình 2.2.3.a2 – Hello World trên web*

b. Chương trình C#/ASP.NET Web lắng nghe ở `localhost:8080`, có các API (không yêu cầu làm việc với database):

GET / (trả về dòng chữ “Hello another world”);

GET /students/{id} (id==null trả về toàn bộ danh sách);

POST /students (tải lên danh sách sinh viên);

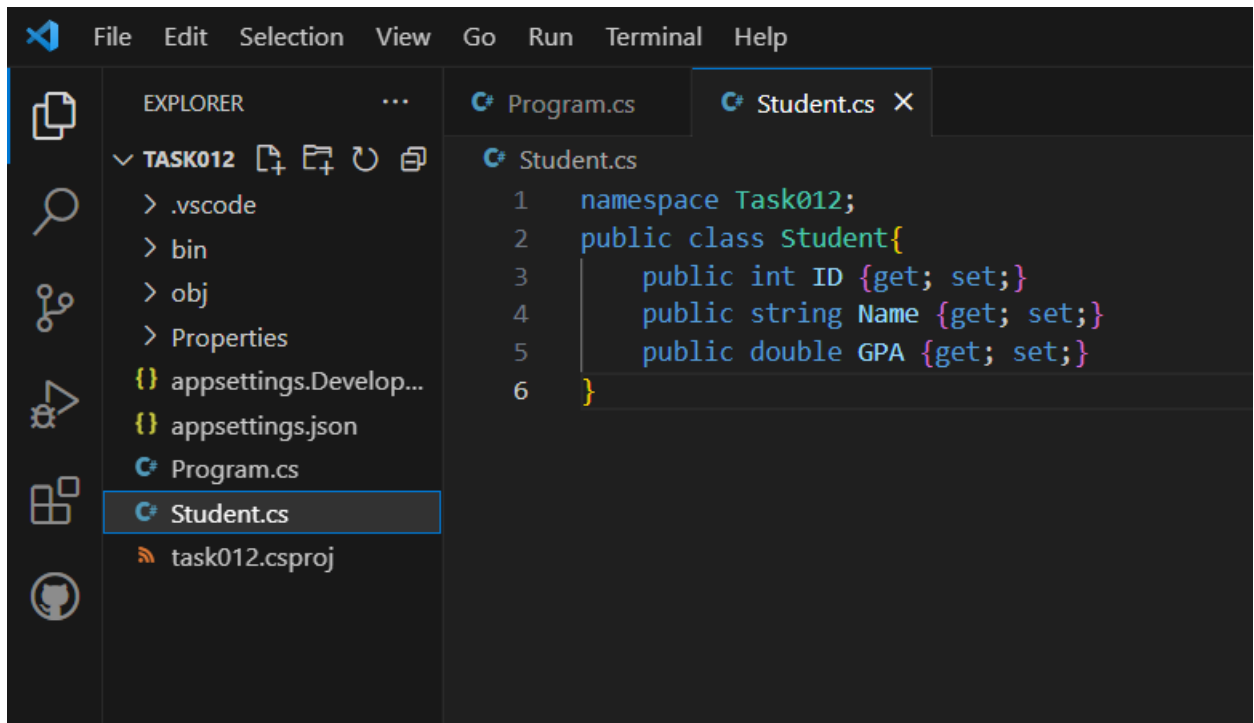
PUT /students/{id} (sửa thông tin sinh viên);

DELETE /students/{id} (xoá sinh viên).

- Tạo thư mục task012: `mkdir task012`

- Tạo một Web Server: `dotnet new Web`

- Tạo một class Student.cs



*Hình 2.2.3.b1 – Class Student*

- Trong Program.cs, tạo các phương thức của HTTP Server:

```
using Task012;
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();
var studentDb = new List<Student>();
void addStudent(List<Student> DS, Student sv){
    DS.Add(sv);
}

app.MapGet("/", () => "Hello another world");
```

*Hình 2.2.3.b2 – Tạo hàm thêm sinh viên  
và phương thức GET ra Hello another world*

```
app.MapGet("/students", () =>{
    return Results.Ok(studentDb);
});
```

*Hình 2.2.3.b3 – Phương thức GET trả về danh sách sinh viên*

```
app.MapGet("/students/{id}", (int id) =>{
    Student? selection = studentDb.Find(s => s.ID == id);
    if (selection == null){
        return Results.NotFound(id);
    }
    return Results.Ok(selection);
});
```

Hình 2.2.3.b4 – Phương thức GET trả về thông tin của sinh viên theo ID

```
app.MapPost("/students", (Student stu) => {
    addStudent(studentDb, stu);
    return Results.Ok(stu);
});
```

Hình 2.2.3.b5 – Phương thức POST

```
app.MapPut("/students/{id}", (int ID, Student sv) => {
    if (ID != sv.ID){
        return Results.BadRequest("Mismatch");
    }
    Student? selection = studentDb.Find(s => s.ID == ID);
    if (selection == null){
        return Results.NotFound("No ID found");
    }
    studentDb.Remove(sv);
    studentDb.Add(sv);
    return Results.Ok(sv);
});
```

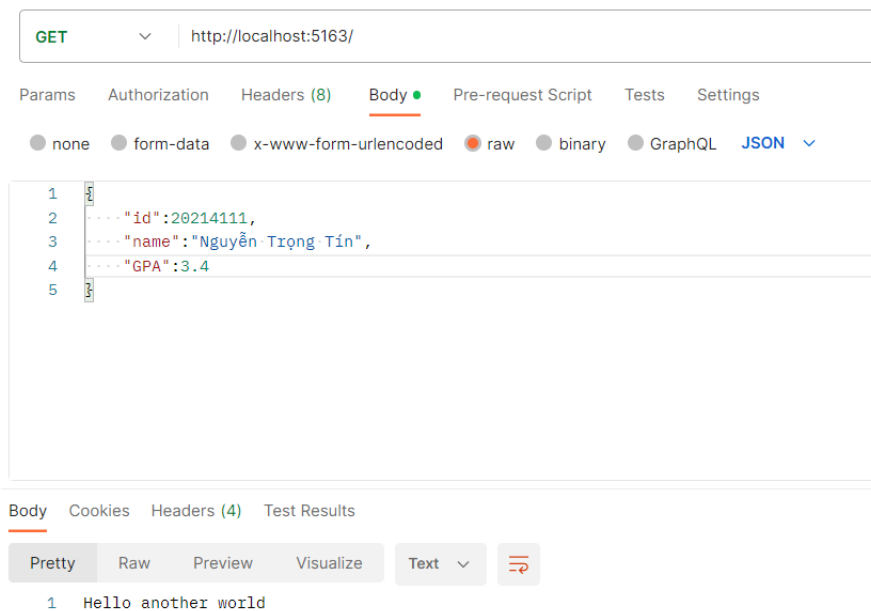
Hình 2.2.3.b6 – Phương thức PUT

```
app.MapDelete("/students/{id}", (int ID) =>{
    Student? selection = studentDb.Find(s => s.ID == ID);
    if(selection != null){
        studentDb.Remove(selection);
    }
    return Results.Ok(selection);
});

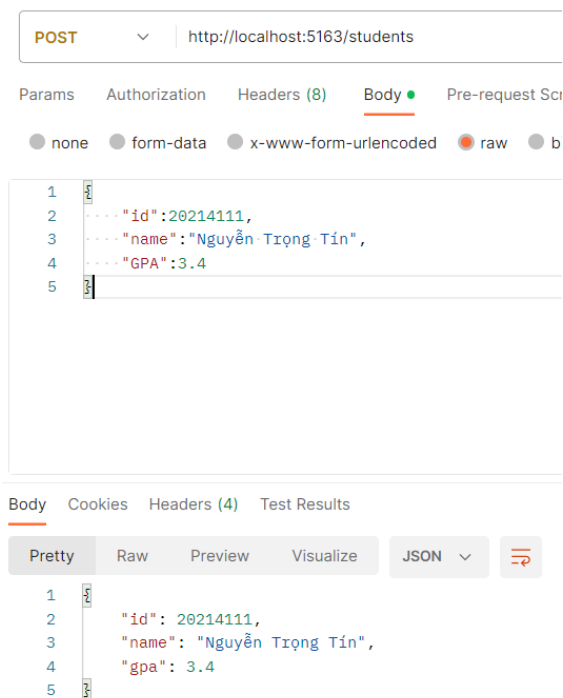
app.Run();
```

Hình 2.2.3.b7 – Phương thức DELETE và lệnh chạy

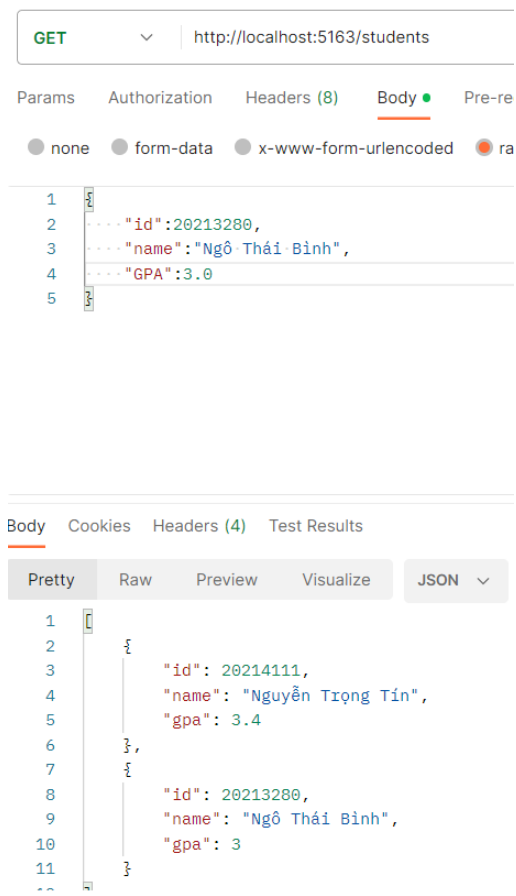
- Chạy thử trên Postman:



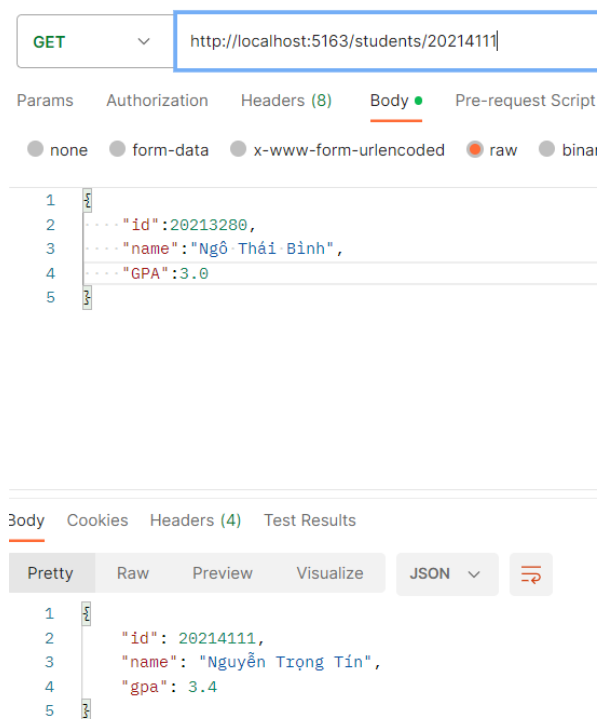
*Hình 2.2.3.b8 – GET Hello another world*



*Hình 2.2.3.b9 – POST*

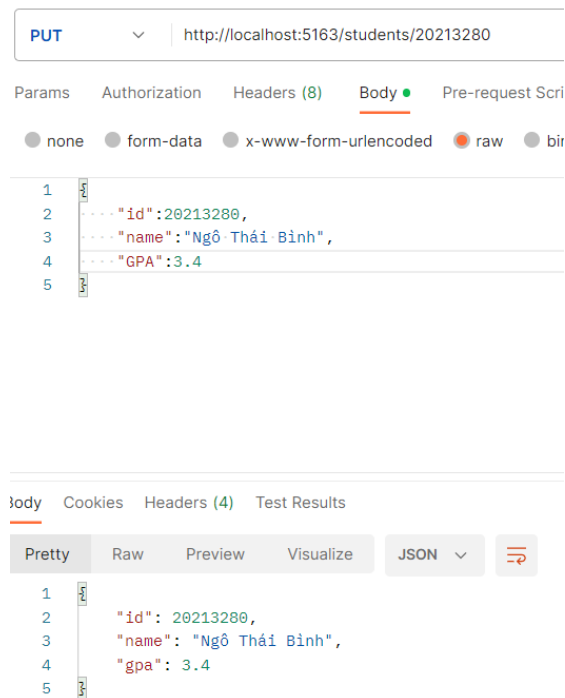


Hình 2.2.3.b10 – GET trả về danh sách sinh viên

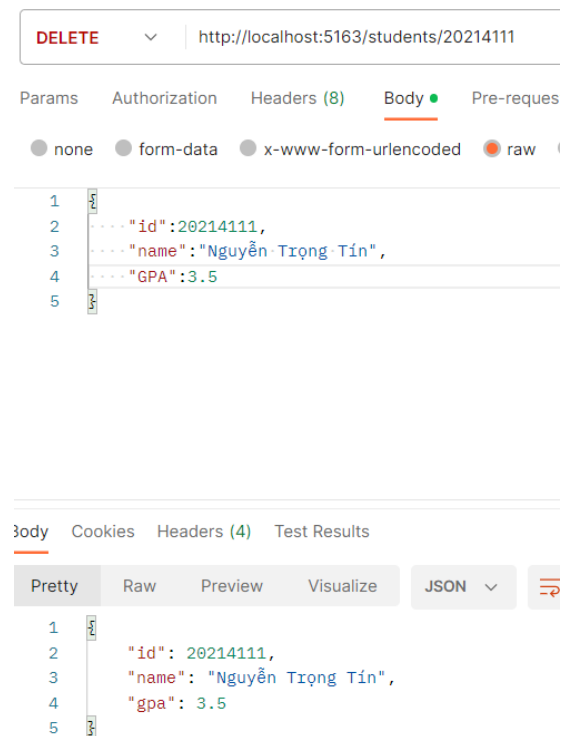




### Hình 2.2.3.b11 – GET lấy thông tin của 1 sinh viên



### Hình 2.2.3.b12 – PUT



- Commit và Push code lên GitHub. Giải quyết các conflict nếu cần thiết.

### 2.3 Day 3:

#### 2.3.1 Mục tiêu:

- Hiểu biết cơ bản về SQL
- Hiểu biết cơ bản về mô hình MVC
- Thực hành viết chương trình đã làm ở Day 2 với Database và mô hình MVC

#### 2.3.2 Kiến thức cần biết:

##### a. SQL:

- Ngôn ngữ truy vấn có cấu trúc (SQL) là một ngôn ngữ lập trình phục vụ việc lưu trữ và xử lý thông tin trong cơ sở dữ liệu quan hệ. Cơ sở dữ liệu quan hệ lưu trữ thông tin dưới dạng bảng có các hàng và cột đại diện cho những thuộc tính dữ liệu và nhiều mối quan hệ khác nhau giữa các giá trị dữ liệu.

- Tại sao SQL lại quan trọng?

+ Là một ngôn ngữ truy vấn phổ biến thường được sử dụng trong tất cả các loại ứng dụng.

+ Các nhà phân tích và phát triển dữ liệu tìm hiểu và sử dụng SQL do ngôn ngữ này tích hợp hiệu quả với nhiều ngôn ngữ lập trình khác nhau.

- Các từ khoá SQL:

+ Lấy dữ liệu:

- SELECT: Sử dụng để lấy dữ liệu từ một hoặc nhiều bảng trong CSDL.
- FROM: Dùng để chỉ định dữ liệu sẽ được lấy ra từ những bảng nào, và các bảng đó quan hệ với nhau như thế nào.
- WHERE: Dùng để xác định những bản ghi nào sẽ được lấy ra, hoặc áp dụng với GROUP BY.
- GROUP BY: Dùng để kết hợp các bản ghi có những giá trị liên quan với nhau thành các phần tử của một tập hợp nhỏ hơn các bản ghi
- HAVING: Dùng để xác định những bản ghi nào là kết quả lấy từ từ khoá GROUP BY, sẽ được lấy ra.
- ORDER BY: Dùng để xác định dữ liệu lấy ra sẽ được sắp xếp theo những cột nào.

+ Sửa đổi dữ liệu:

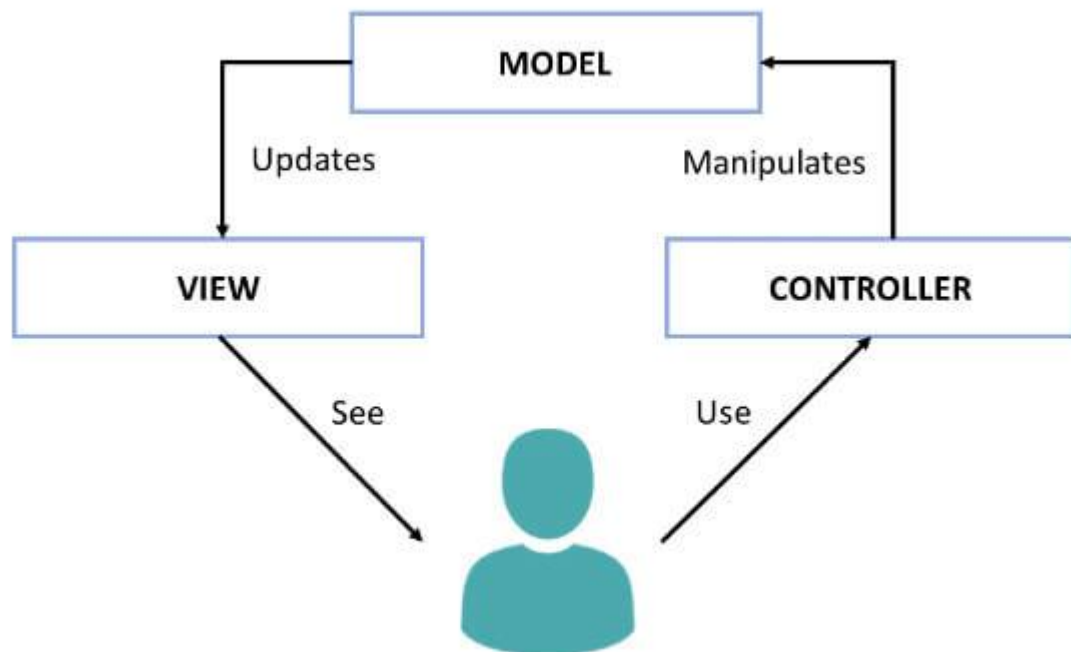
- INSERT: Dùng để thêm dữ liệu vào một bảng đã tồn tại.
- UPDATE: Dùng để thay đổi giá trị của một tập hợp các bản ghi trong một bảng.
- MERGE: Dùng để kết hợp dữ liệu của nhiều bảng. Nó được dùng như việc kết hợp giữa hai phân tử INSERT và UPDATE.
- DELETE: Xóa những bản ghi tồn tại trong một bảng.
- TRUNCATE: Xóa toàn bộ dữ liệu trong một bảng (không phải là tiêu chuẩn, nhưng là một lệnh SQL phổ biến).

b. Mô hình MVC:

- MVC là viết tắt của cụm từ “Model-View-Controller”.

- MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau và mỗi thành phần đều có một nhiệm vụ riêng của nó độc lập với các thành phần khác. Tên gọi 3 thành phần:

- Model (dữ liệu): Quản lý xử lý các dữ liệu. Là cầu nối giữa 2 thành phần View và Controller.
- View (giao diện): Nơi hiển thị dữ liệu cho người dùng.
- Controller (bộ điều khiển): Điều khiển sự tương tác của hai thành phần Model và View. Nó nhận các input và thực hiện các update tương ứng.



*Hình 2.3.2.b1 – Nguyên lý hoạt động của mô hình MVC*

- Nguyên lý hoạt động của MVC:

+ Khi một yêu cầu của từ máy khách (Client) gửi đến Server, thì bị Controller trong MVC chặn lại để xem đó là URL request hay sự kiện.

- + Sau đó, Controller xử lý input của user rồi giao tiếp với Model trong MVC.
- + Model chuẩn bị data và gửi lại cho Controller.
- + Cuối cùng, khi xử lý xong yêu cầu thì Controller gửi dữ liệu trở lại View để hiển thị cho người dùng trên trình duyệt.

### 2.3.3 Thực hiện:

a. Database với danh sách HOCSINH:

- Tạo Web Server: dotnet new web
- Khởi tạo Database:

```
using Microsoft.EntityFrameworkCore;
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<StudentDb>(opt => opt.UseInMemoryDatabase("StudentList"));

//builder.Services.AddEndpointsApiExplorer();
builder.Services.AddDatabaseDeveloperPageExceptionFilter();
var app = builder.Build();
```

*Hình 2.3.3.a1 – Database*

- Khai báo các class sử dụng:

```
public class Student
{
    3 references
    public int id { get; set; }
    3 references
    public string? name { get; set; }
    3 references
    public double GPA { get; set; }
}

6 references
public class StudentDTO
{
    2 references
    public int id { get; set; }
    3 references
    public string? name { get; set; }
    3 references
    public double GPA { get; set; }

    0 references
    public StudentDTO() { }
    4 references
    public StudentDTO(Student student) =>
        (id, name, GPA) = (student.id, student.name, student.GPA);
}

7 references
class StudentDb : DbContext
{
    0 references
    public StudentDb(DbContextOptions<StudentDb> options) : base(options) { }
    6 references
    public DbSet<Student> Students => Set<Student>();
}
```

*Hình 2.3.3.a2 – Class Student và StudentDTO*

- Khai báo các phương thức:

```
app.MapGet("/", () => "Hello another world!");
```

*Hình 2.3.3.a3 – GET ra Hello another world*

```
app.MapGet("/student", async (StudentDb db) => await db.Students.Select(x=> new StudentDTO(x)).ToListAsync());
```

*Hình 2.3.3.a4 – GET ra hết bảng sinh viên*

```
app.MapGet("/student/{id}", async (int id, StudentDb db) =>
    await db.Students.FindAsync(id)
        is Student stu
        ? Results.Ok(new StudentDTO(stu))
        : Results.NotFound());
```

*Hình 2.3.3.a5 – GET ra thông tin sinh viên theo ID*

```
app.MapPost("/student", async(StudentDTO studentDTO, StudentDb db) =>
{
    var sv = new Student
    {
        id = studentDTO.id,
        name = studentDTO.name,
        GPA = studentDTO.GPA
    };
    db.Students.Add(sv);
    await db.SaveChangesAsync();
    return Results.Created($"/student/{sv.id}", new StudentDTO(sv));
});
```

*Hình 2.3.3.a6 – POST*

```
app.MapPut("/student/{id}", async (StudentDb db, StudentDTO studentDTO, int id) =>
{
    var student = await db.Students.FindAsync(id);
    if (student is null) return Results.NotFound();
    student.name = studentDTO.name;
    student.GPA = studentDTO.GPA;
    await db.SaveChangesAsync();
    return Results.NoContent();
});
```

*Hình 2.3.3.a7 – PUT*

```

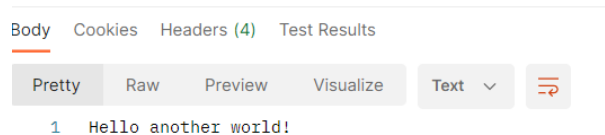
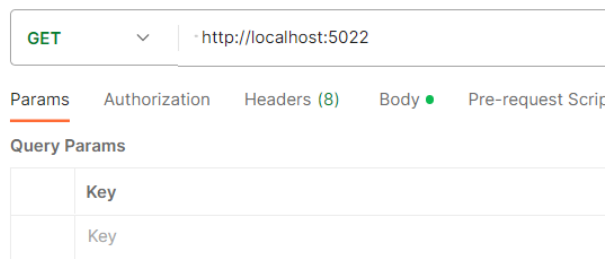
app.MapDelete("/student/{id}", async (int id, StudentDb db) =>
{
    if (await db.Students.FindAsync(id) is Student student)
    {
        db.Students.Remove(student);
        await db.SaveChangesAsync();
        return Results.Ok(new StudentDTO(student));
    }

    return Results.NotFound();
});

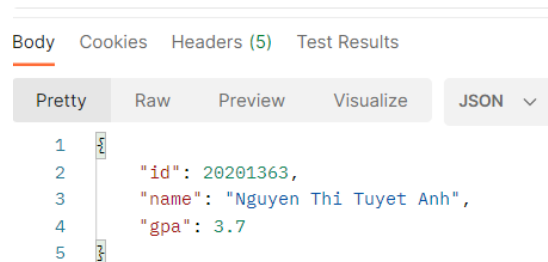
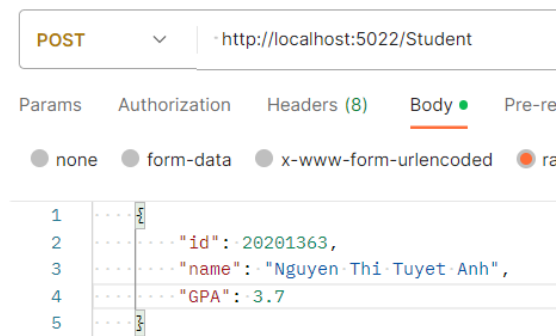
```

*Hình 2.3.3.a8 – DELETE*

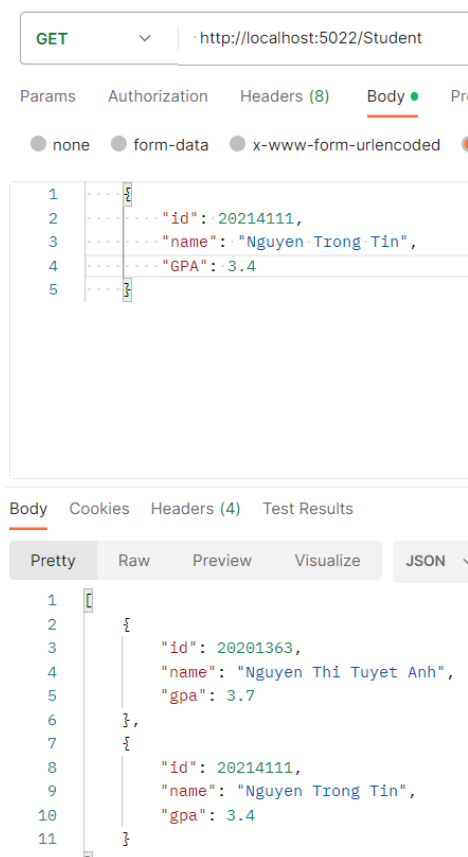
- Chạy thử code trên Postman:



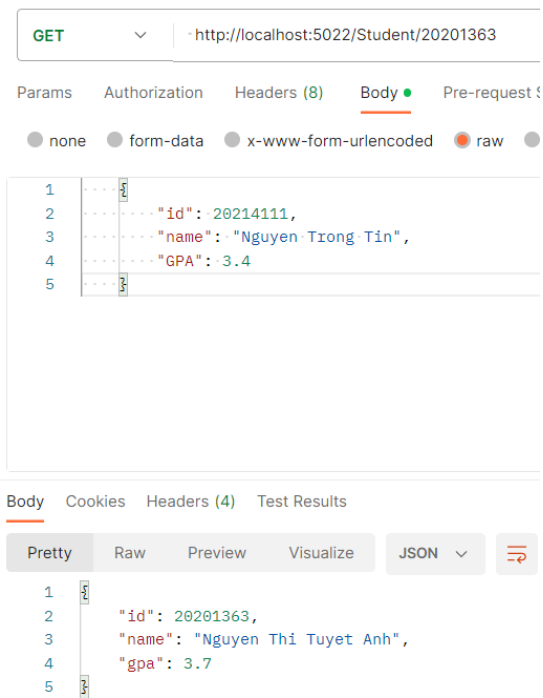
*Hình 2.3.3.a9 – GET ra Hello World*



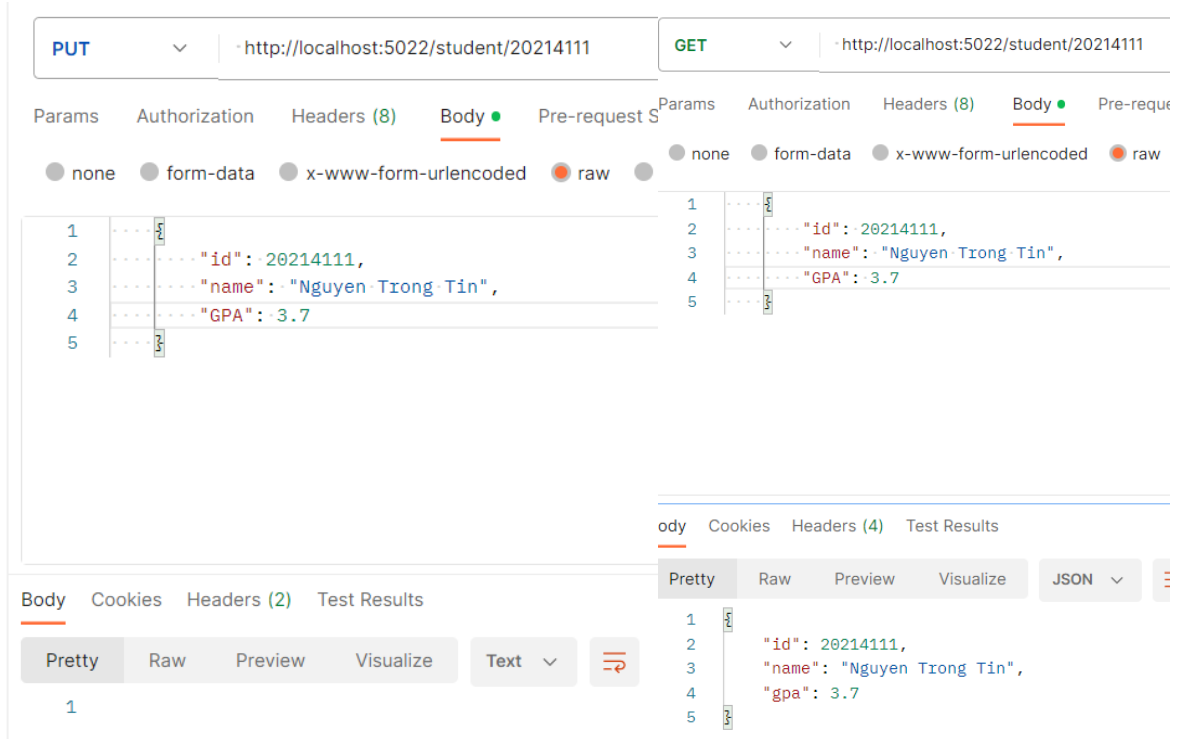
*Hình 2.3.3.a10 – POST*



*Hình 2.3.3.a11 – GET ra danh sách toàn sinh viên*



Hình 2.3.3.a12 – GET ra thông tin 1 sinh viên



Hình 2.3.3.a13 – PUT





*Hình 2.2.3.a14 – DELETE*

- Commit và push code lên GitHub. Giải quyết các conflict nếu cần thiết.
- b. Mô hình MVC với danh sách HOCSINH:
  - Tạo Web Server: `dotnet new web`
  - Models: Khai báo hai class `DataSeed.cs` và `Student.cs`.

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System;
namespace MvcStudent.Models{
    21 references
    public class Student{
        [Key]
        9 references
        public int id {get; set;}
        15 references
        public string? name {get; set;}

        15 references
        public double GPA {get; set;}
    }
}

```

*Hình 2.3.3.b1 – Student.cs*

```

1 using Microsoft.EntityFrameworkCore;
2 using Microsoft.Extensions.DependencyInjection;
3 using MvcStudent.Data;
4 using System;
5 using System.Linq;
6
7 namespace MvcStudent.Models;
8
9 1 reference
10 public static class SeedData
11 {
12     1 reference
13     public static void Initialize(IServiceProvider serviceProvider)
14     {
15         using (var context = new MvcStudentContext(
16             serviceProvider.GetRequiredService<
17                 DbContextOptions<MvcStudentContext>>())
18         {
19             // Look for any movies.
20             if (context.Student.Any())
21             {
22                 return; // DB has been seeded
23             }
24             context.Student.AddRange(
25                 new Student
26                 {
27                     id = 20212860,
28                     name = "Trương Đức Nhật",
29                     GPA = 3.5
30                 },
31                 new Student
32                 {
33                     id = 20211793,
34                     name = "Tùng Linh",
35                     GPA = 3.2
36                 },
37                 new Student
38                 {
39                     id = 20214111,
40                     name = "Nguyễn Trọng Tín",
41                     GPA = 3.4
42                 }
43             );
44             context.SaveChanges();
45         }
46     }
47 }

```

Hình 2.3.3.b2 – DataSeed.cs

- Controller: Tạo StudentController với các phương thức

```

1 using Microsoft.AspNetCore.Mvc;
2 using MvcStudent.Data;
3 using System.Text.Encodings.Web;
4 using MvcStudent.Models;
5
6 namespace MvcStudent.Controllers;
7
8 [Route("[controller]")]
9 0 references
10 public class StudentController : Controller
11 {
12     //
13     12 references
14     private readonly MvcStudentContext _context;
15
16     0 references
17     public StudentController(MvcStudentContext context)
18     {
19         _context = context;
20     }
21 }

```

Hình 2.3.3.b3 – Khởi tạo StudentController

```

18 [HttpGet("/")]
19 0 references
20 public IActionResult HelloWorld(){
21     return Results.Ok("Hello another world");
22 }

```

Hình 2.3.3.b4 – GET trả về Hello another world

```

22 | [HttpGet("{id}")]
    | 0 references
23 | public IActionResult Get([FromRoute]int id) {
24 |     return Results.Ok(_context.Student.Find(id));
25 | }

```

Hình 2.3.3.b5 – GET trả về thông tin sinh viên theo ID

```

[HttpGet]
0 references
public IActionResult GetAll(){
    return Results.Ok(_context.Student.ToList());
}

```

Hình 2.3.3.b6 – GET trả về toàn bộ danh sách sinh viên

```

[HttpPost]
0 references
public IActionResult Post([FromBody]List<Student> Students){
    _context.Student.AddRange(Students);
    _context.SaveChanges();
    return Results.Ok(Students);
}

```

Hình 2.3.3.b7 – POST

```

[HttpPut("{id}")]
0 references
public IActionResult Put([FromRoute]int id, [FromBody]Student stu){
    var old = _context.Student.Find(id);
    if (old == null){
        return Results.NotFound();
    }
    _context.Student.Remove(old);
    _context.Student.Add(stu);
    _context.SaveChanges();
    return Results.Ok(stu);
}

```

Hình 2.3.3.b8 – PUT

```

public IActionResult Delete([FromRoute]int id){
    var sv = _context.Student.Find(id);
    if(sv == null){
        return Results.NotFound();
    }
    _context.Student.Remove(sv);
    _context.SaveChanges();
    return Results.Ok(sv);
}
}

```

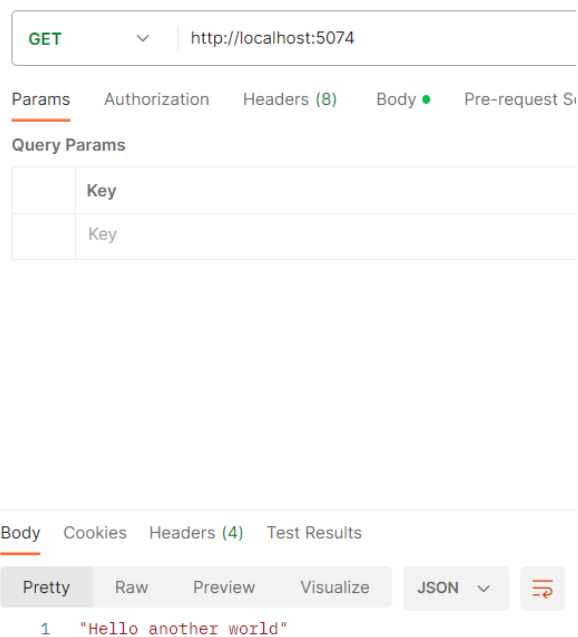
Hình 2.3.3.b9 – DELETE

- Tạo file Migrations (Nhằm khởi tạo và cập nhật database khớp với dữ liệu Model):  
dotnet ef migrations add InitialCreate  
dotnet ef database update
- Cập nhật Database vào file Program.cs:

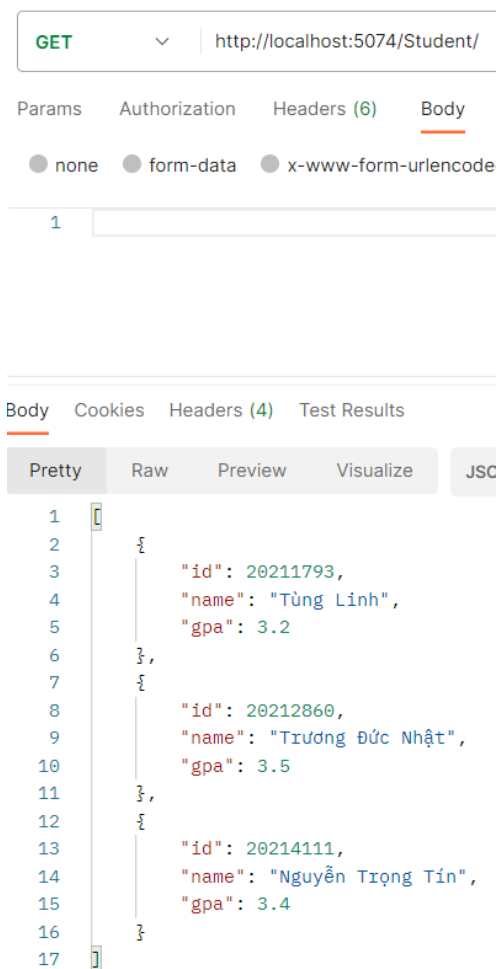
```
1  using Microsoft.EntityFrameworkCore;
2  using Microsoft.Extensions.DependencyInjection;
3  using MvcStudent.Data;
4  using MvcStudent.Models;
5
6  var builder = WebApplication.CreateBuilder(args);
7
8  builder.Services.AddDbContext<MvcStudentContext>(options =>
9  |      options.UseSqlite(builder.Configuration.GetConnectionString("MvcStudentContext")));
10 ▶ builder.Services.AddControllersWithViews();
11
12 var app = builder.Build();
13
14 using (var scope = app.Services.CreateScope())
15 {
16     var services = scope.ServiceProvider;
17
18     SeedData.Initialize(services);
19 }
20
21 ▶ if (!app.Environment.IsDevelopment())
22 {
23     app.UseExceptionHandler("/Home/Error");
24     app.UseHsts();
25 }
26
27 ▶ app.UseStaticFiles();
28
29 app.UseRouting();
30
31 ▶ app.MapControllerRoute(
32     name: "default",
33     pattern: "{controller=Home}/{action=Index}/{id?}");
34
35 app.Run();
36
```

*Hình 2.3.3.b10 – Program.cs*

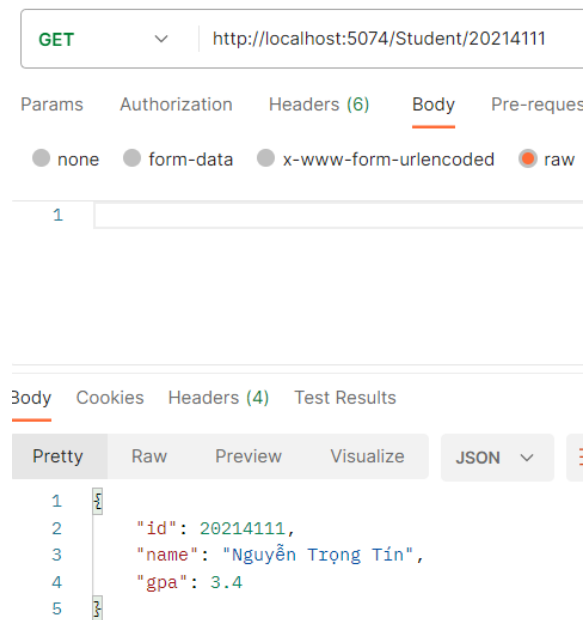
- Chạy thử trên Postman:



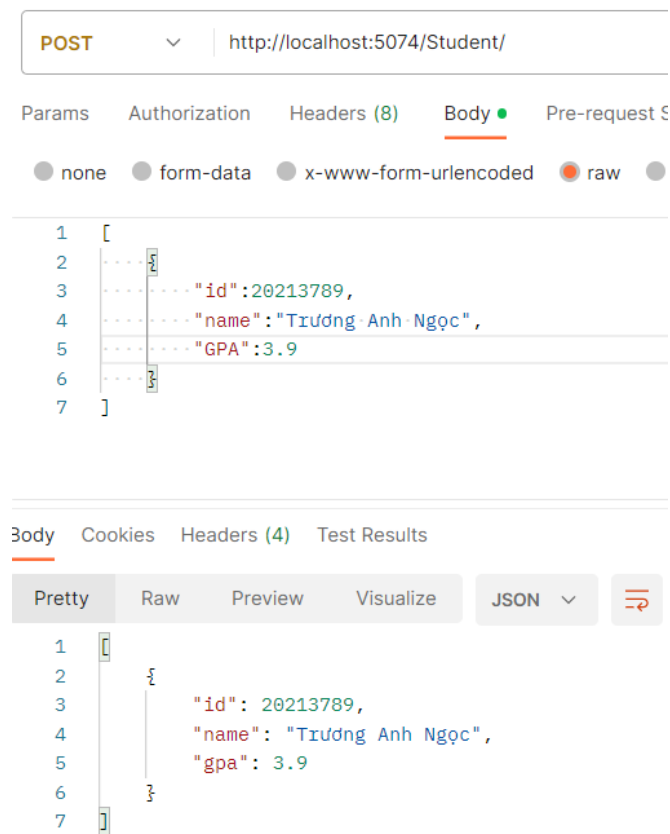
*Hình 2.3.3.b11 – GET ra Hello another world*



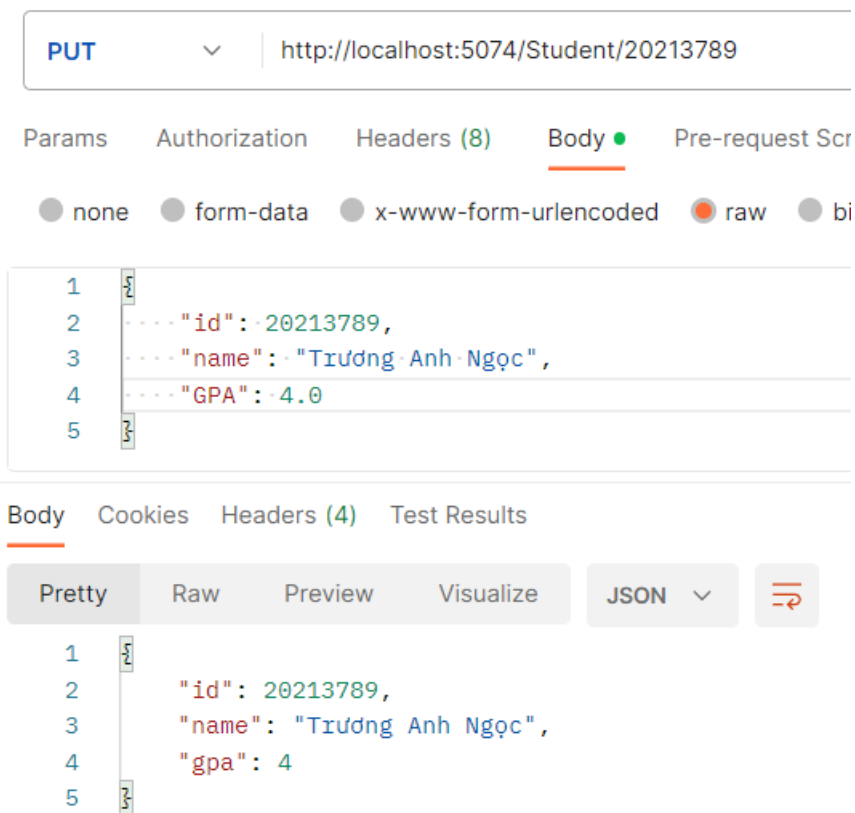
*Hình 2.3.3.b12 – GET ra danh sách sinh viên*



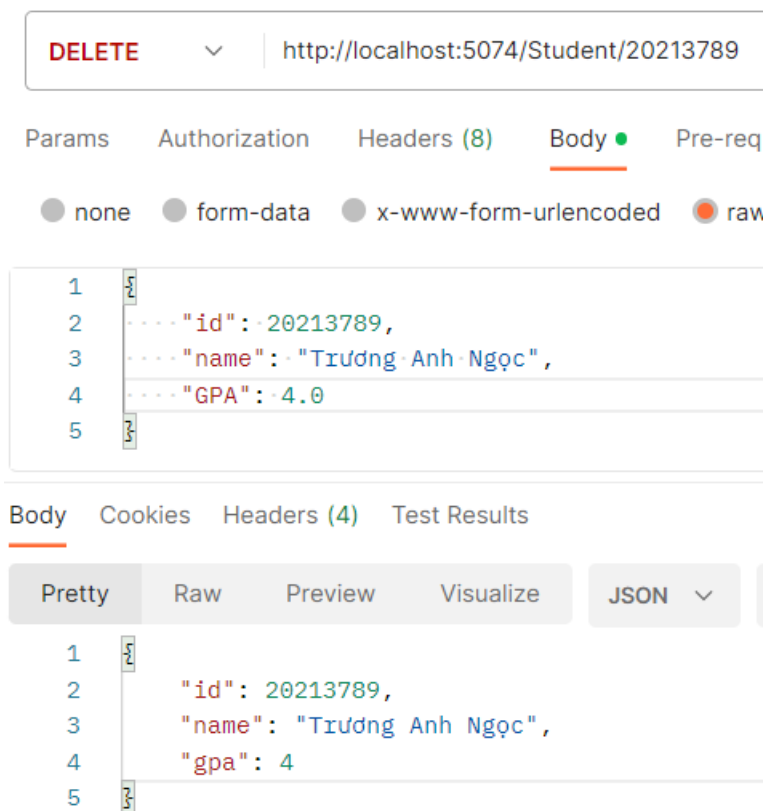
*Hình 2.3.3.b13 – GET ra thông tin một sinh viên*



*Hình 2.3.3.b14 – POST*



Hình 2.3.3.b15 – PUT



Hình 2.3.3.b16 – DELETE

- Commit và push code lên GitHub. Giải quyết các conflict nếu cần thiết.

### 3. Code tham khảo:

- Task 02 Day 1: <https://viettuts.vn/bai-tap-csharp/bai-tap-quan-ly-sinh-vien-trong-csharp>
- Task 01 Day 3:  
[https://github.com/slthomason/StartupHakk/tree/main/02\\_Mininum\\_WebApi\\_Aspnet\\_Core](https://github.com/slthomason/StartupHakk/tree/main/02_Mininum_WebApi_Aspnet_Core)
- Task 02 Day 3: <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?source=recommendations&view=aspnetcore-7.0&tabs=visual-studio-code>