API 🡺 application Programming Interface

Develop backend application 🡺 API can be access through Http/Https Protocol.

HTTP Methods (or) Verbs 🡺 GET,POST,PUT,PATCH,DELETE

GET 🡺 Read the Data

POST 🡺 Create (or) insert new item

PUT 🡺 Update the whole information

PATCH 🡺 Update the Partial information

DELETE 🡺 Remove an item

Why ASP.NET Core Web API 🡪 Cross-Platform, Open-source framework

Web Application 🡺 can have model (Entities or classes), view (presentation layer) ,controller (handle the request and response) (Business logic)

Web API 🡺 Model, (entities) ,

* Controller handling the request and response

Inside Controller we have methods which is called as **Action methods**

Web API is act as bridge between your FrontEnd and Backend (Data Layer)

Web API can be access any Devices such as Laptop,Mobile, tablet,IOT devices

Restful service -> Representational state transfer

As it supports http protocol we are using HTTP verbs (GET,POST,PUT,DELETE)

Http status codes to return the response which is very easy to identify the response, or error and what types of error we are getting based on the status code.

Status code => 200

204

400 -> 401, 404,403

300

500 ->

Web API 🡺 by default it returns the data in JSON format

Media type /content type 🡺text/xml, text/json

Security 🡺 Authentication & Authorization

**1. What is Middleware?**

* **Middleware** is a **software component** that is executed in the **request–response pipeline** of an ASP.NET Core application.
* Every request that comes into the application passes through a **pipeline of middleware components** before reaching the controller.
* Each middleware can:
  1. **Process** the request.
  2. **Pass it** to the next middleware in the pipeline.
  3. **Short-circuit** (stop) the pipeline and return a response immediately.

**Common Middleware in ASP.NET Core Web API**

Some built-in middleware components:

* **Exception Handling Middleware** → Catches and handles errors globally.
* **Routing Middleware** → Matches request URL to controller/action.
* **Authentication Middleware** → Validates user identity (JWT, OAuth, etc.).
* **Authorization Middleware** → Ensures the user has the right permissions.
* **Static Files Middleware** → Serves images, CSS, JS files.
* **CORS Middleware** → Allows cross-origin requests (important for frontend apps).
* **Custom Middleware** → You can write your own middleware for logging, auditing, etc.

public int Id { get; set; }

public string? DepartmentName { get; set; }

public string? Location { get; set; }

get 🡺 Readonly properties

set 🡺 write-only property

get,set 🡺read and write property