**Harry’s notes for software development**

Table of Contents

[1 Microservices 1](#_Toc159348905)

[1.1 Definition 1](#_Toc159348906)

[2 GitHub related 1](#_Toc159348907)

[2.1 Git clone 1](#_Toc159348908)

[2.2 Create gitignore for .NET 2](#_Toc159348909)

[3 API (.NET) 3](#_Toc159348910)

[3.1 Create API 3](#_Toc159348911)

[4 Controller – .NET for API 4](#_Toc159348912)

[4.1 What make the controller to be a controller 4](#_Toc159348913)

[4.2 What make an end point to be an end point 5](#_Toc159348914)

[4.3 Handling return of status codes 5](#_Toc159348915)

[4.4 Document the return status of end point in Swagger 5](#_Toc159348916)

[5 Visual Studio Code 5](#_Toc159348917)

[5.1 Suggested Setting on VS Code 5](#_Toc159348918)

[5.2 If not .vscode folder generated 5](#_Toc159348919)

[6 Dotnet CLI commands 5](#_Toc159348920)

[7 Items is being Looked for solutions 6](#_Toc159348921)

# Microservices

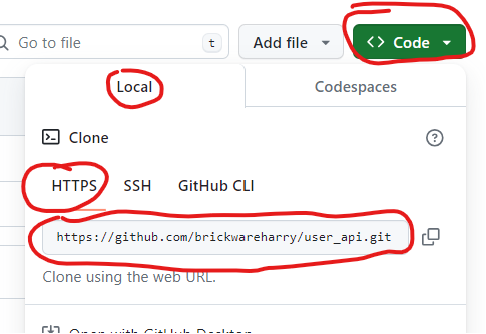
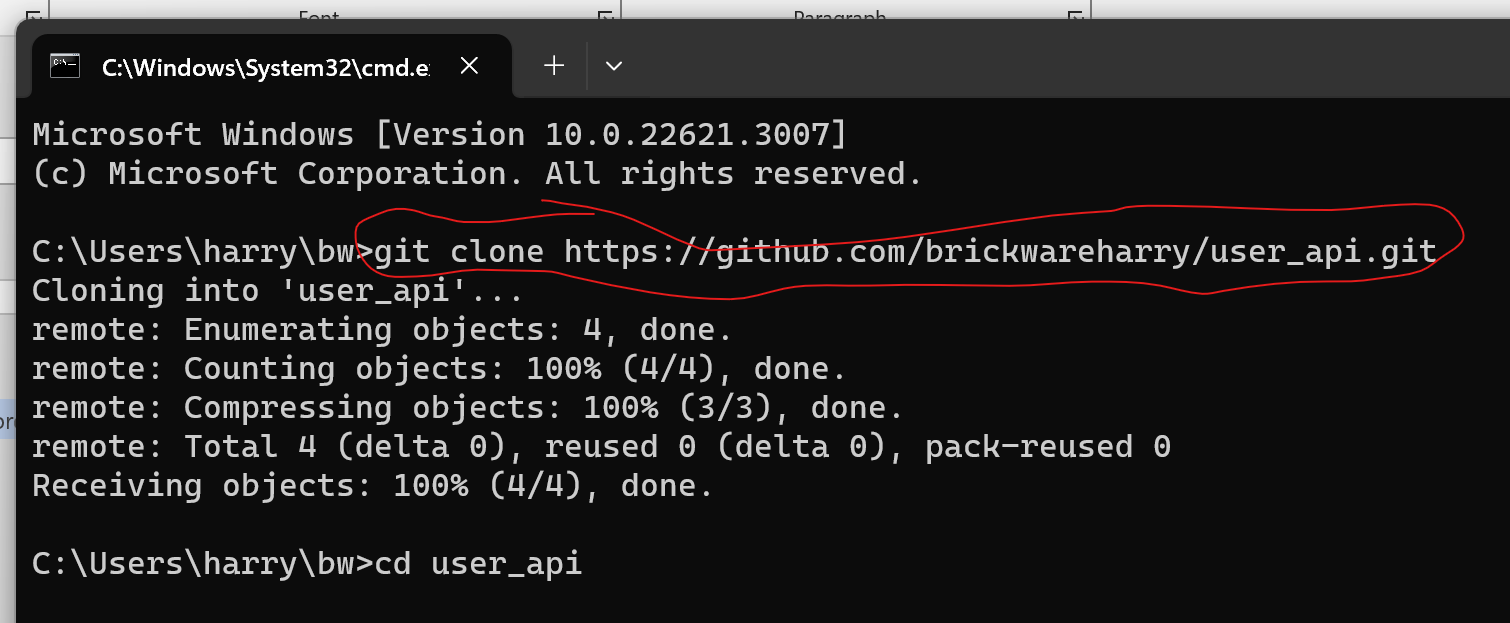
## Definition

Microservices have the architectural style that structures an application as a collection of independently deployable services that are modeled around a business domain and are usually owned by a small team.

# GitHub related

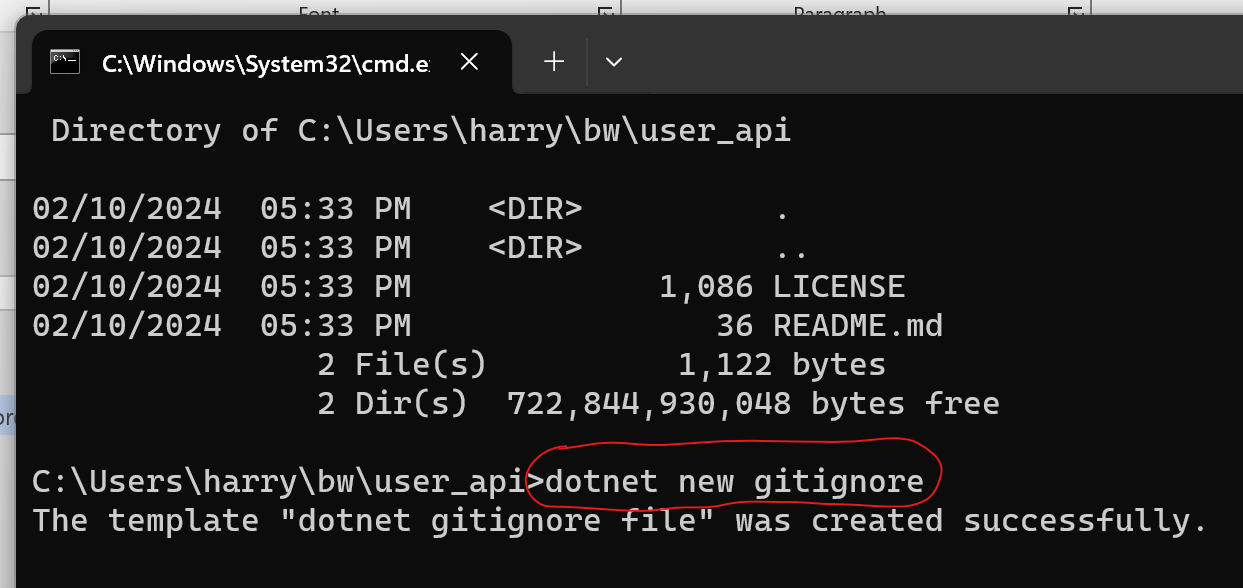
## Git clone

git clone <<URL of git repository, see screenshot below>>

## Create gitignore for .NET

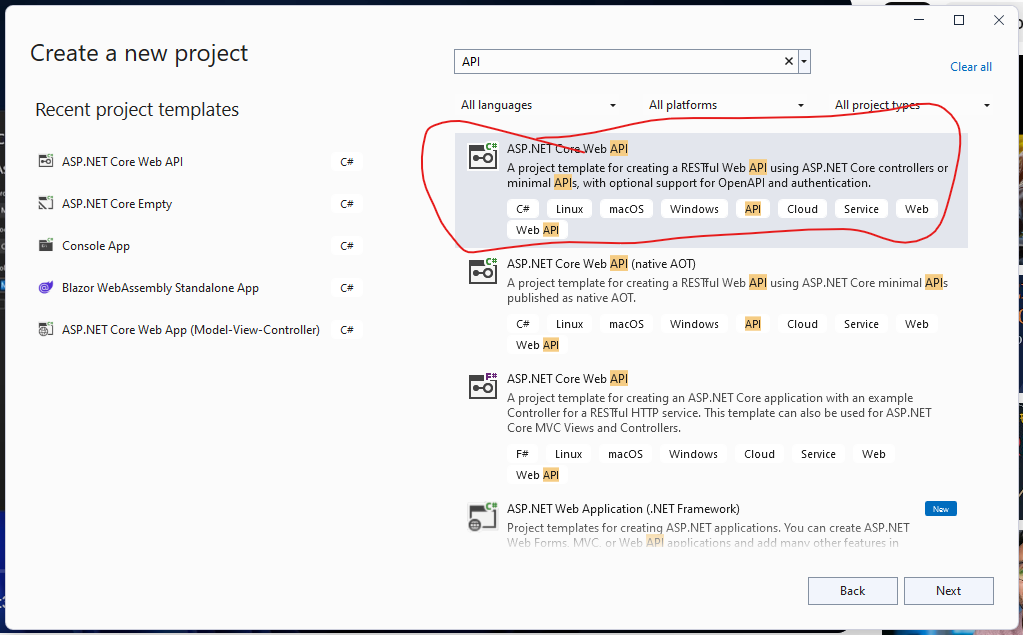
Give command “dotnet new gitignore” in the Command Prompt of the local repository, see screenshot below

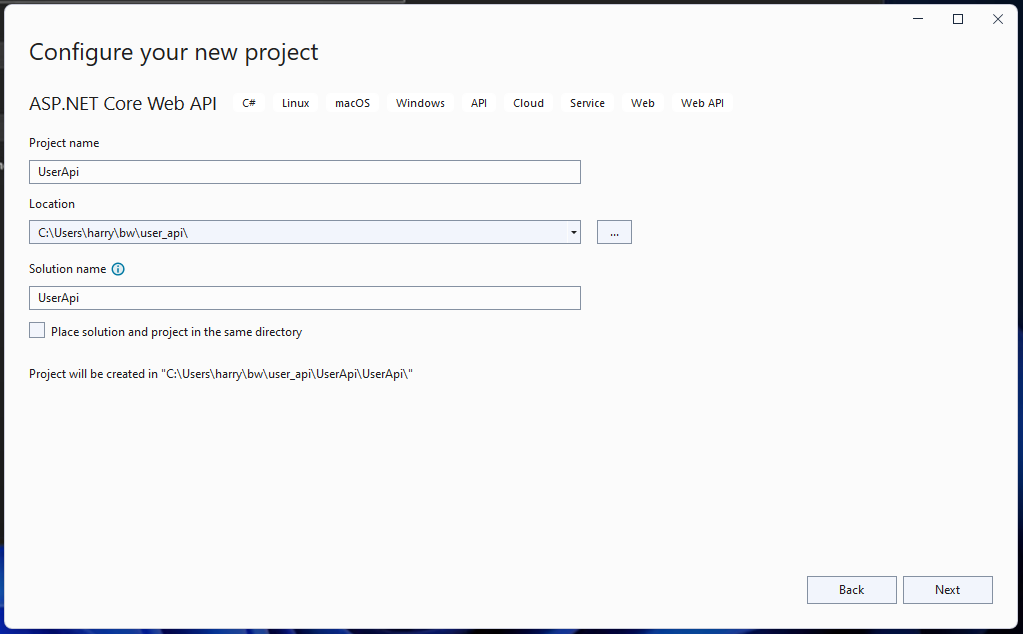
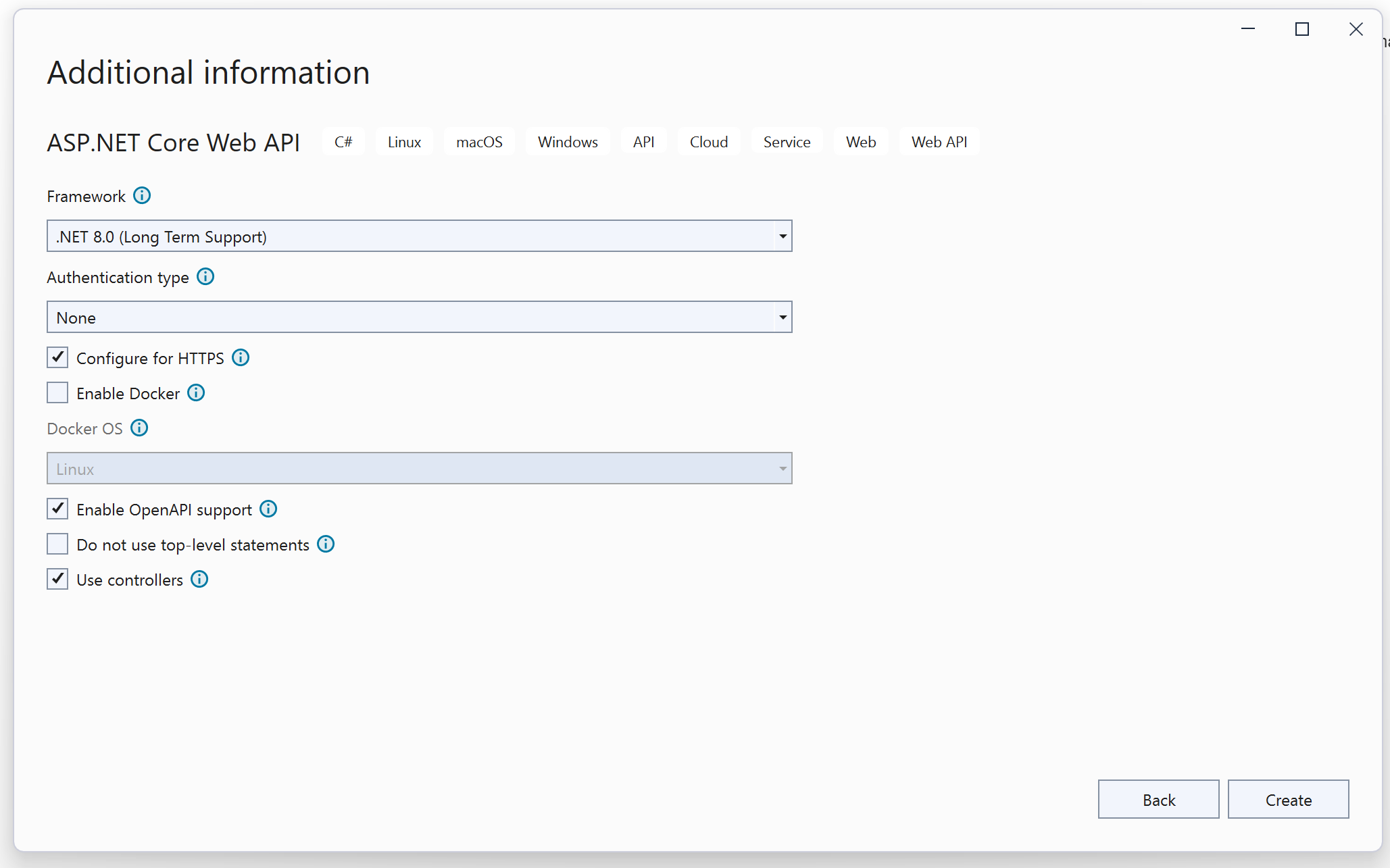


# API (.NET)

## Create API

* Create a new ASP.NET Core Web API project in Visual Studio



# Controller – .NET for API

## What make the controller to be a controller

* Has a suffix of Controller in the name of the class and file
  + Example: UserApiController
* Derive from the ControllerBase class under namespace of Microsoft.AspNetCore. Mvc
  + Example: public class UserApiController:ControllerBase
  + A using statement is needed below:
    - Example: using Microsoft.AspNetCore.Mvc;
* Need a controller attribute placed above the class
  + Example: [ApiController]
* Need a route attribute placed above the class
  + Example: [Route("api/UserApi")] or [Route("api/[Controller]")]

## What make an end point to be an end point

* Need a verb attribute place above the function of the end point
  + Example: [HttpGet] or [HttpGet(“{id:int}”)]

## Handling return of status codes

* Return type at the function of the end point should be ActionResult<return type of object>, e.g. ActionResult<UserDto> and ActionResult<IEnumerable<UserDto>>
* Verification of the return value can specify the return status, e.g.:
  + If(id == 0){return BadRequest()};
  + If(user == null){return NotFound()};
  + return Ok();

## Document the return status of end point in Swagger

* Attributes [ProducesResponseType(status code)] are needed for documenting the status above the function of end point, e.g.
  + [ProducesResponseType(200)] or [ProducesResponseType(StatusCodes.Status200OK)]
  + [ProducesResponseType(400)] or [ProducesResponseType(StatusCodes.Status400BadRequest)]
  + [ProducesResponseType(404)] or [ProducesResponseType(StatusCodes.Status404NotFound)]

# Visual Studio Code

## Suggested Setting on VS Code

* File -> Auto Save
* File -> Preferences -> Settings -> Text Editor -> Formatting -> Format On Save
* File -> Preferences -> Settings -> Search(place open brace on new line)-> JavaScript …. Place Open Brace on New Line For Control Blocks
* File -> Preferences -> Settings -> Search(place open brace on new line)-> JavaScript …. Place Open Brace on New Line For Functions
* File -> Preferences -> Settings -> Search(place open brace on new line)-> TypeScript …. Place Open Brace on New Line For Control Blocks
* File -> Preferences -> Settings -> Search(place open brace on new line)-> TypeScript …. Place Open Brace on New Line For Functions

## If not .vscode folder generated

* Generate the .vscode folder manually by opening your Command Palette (CTRL + SHIFT + P) and then select .NET: Generate Assets For Build And Debug.
* The .vscode folder has launch.json and tasks.json

# Dotnet CLI commands

* dotnet new –list
  + list all the template projects which can be built
* dotnet new webapi -n <<project name>> -controllers
  + create an webapi project with the given project name
* dotnet new console –name <<project name>>
  + this creates an project with the name and also create a new folder holding the project
  + if –name <<project name>> is not provided, the project will be created in current folder and the folder name will be the project name
  + --use-program-main for not using top level statement (i.e. there is static void Main(string[] args) in program.cs)

# Items is being Looked for solutions