# Verificar la version de .net

* dotnet –version

# Estructura

## CP/Controllers/

Es donde van a estar los controladores

## CP/Program.cs

Es donde van a estar los servicios

# Modelos

Los vamos a ubicar de momento en CP/Modelos

En un principio son solo clases con properties Ejemplo:

namespace PruebaWeb1.Modelos

{

public class Villa

{

public int Id { get; set; }

public string Name { get; set; }

public DateTime Fecha { get; set; }

}

}

### DTO

Generalmente no se trabaja directamente con el modela de la base de datos, sino con uno intermedio

CP/Modelos/Dto/

Debemos poner generalmente NombreModeloDto.cs

Son tambien unas clases que solo tienen properties

namespace PruebaWeb1.Modelos.Dto

{

public class VillaDto

{

public int Id { get; set; }

public string Name { get; set; }

}

}

#### Validaciones

Son por medio de anotaciones

using System.ComponentModel.DataAnnotations;

namespace PruebaWeb1.Modelos.Dto

{

public class VillaDto

{

public int Id { get; set; }

[Required]

[MaxLength(255)]

public string Name { get; set; }

}

}

# Controladores

Siempre deben de terminar con ’ Controller.cs’ ejemplo MiControladorController.cs

Ejemplo uno tipo API vacío

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace PruebaWeb1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class VillaController : ControllerBase

{

}

}

Generalmente heredan de la clase ‘ControllerBase’

Para identificar que es un controlador tiene el (la marca) atributo [ApiController]

Desde el principio se comienza a definir la ruta para ello tiene el (la marca) atributo [Route("api/[controller]")]

## Endpoints

Los endpoint están dentro de los controladores

Van dentro de los controladores

namespace PruebaWeb1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class VillaController : ControllerBase

{

[HttpGet]

public IEnumerable<Villa> GetVillas() {

return new List<Villa> {

new Villa {Id=1, Name="vista1"},

new Villa {Id=2, Name="vista2"},

new Villa {Id=3, Name="vista3"},

};

}

}

}

El endpoint tiene que estar marcado con su verbo [HttpGet]

Si es una lista tiene que retornar un IEnumerable<> y la implementación puede ser un List< >

Se van a diferenciar según sus anotaciones de tipos, las rutas van a ser determinadas por el nombre del controlador, por ejemplo de VillaController va a ser de

* /api/Villa [get]
* /api/Villa [post]
* /api/Villa [put]
* /api/Villa/id:int [get]
* /api/Villa/{id} [delete]

#### Atributos Get

Puedes tener dos endpoint del mismo tipo y nombre (del metodo), pero diferenciados en sus parámetros

Para declarar un parámetro al get, se agrega en la marca [HttpGet("id")]

Y luego hay que decolarlos como atributos del método

[HttpGet("id")]

public VillaDto GetVilla(int id)

{

return VillaStore.villaList.FirstOrDefault(v=> v.Id==id);

}

##### Tipos de datos

Se le puede declarar al lado de su nombre en la marca

[HttpGet("id:int")]

public VillaDto GetVilla(int id)

{

return VillaStore.villaList.FirstOrDefault(v=> v.Id==id);

}

#### ActionResult

Una manera mas correcta de dar las respuestas es mediante ActionResult

Tiene

* Ok(v);
* BadRequest();
* NotFound();
* StatusCode(StatusCodes.Status500InternalServerError);
* NoContent()

[HttpGet("id:int")]

public ActionResult<VillaDto> GetVilla(int id)

{

if (id == 0) {

return BadRequest();

}

var villa = VillaStore.villaList.FirstOrDefault(v => v.Id == id);

if (villa == null)

{

return NotFound();

}

return Ok();

}

##### Redirección

Se puede redirigir a otro endpoint que retorne un mismo tipo de dato que el declarado y retornar a la vez su respuesta

public ActionResult<VillaDto> CrearVilla([FromBody] VillaDto villaDto)

{

if (villaDto == null)

{

return BadRequest(villaDto);

}

if (villaDto.Id>0)

{

return StatusCode(StatusCodes.Status500InternalServerError);

}

villaDto.Id = VillaStore.villaList.OrderByDescending(v => v.Id).FirstOrDefault().Id+1;

VillaStore.villaList.Add(villaDto);

return CreatedAtRoute("GetVilla",new { id=villaDto.Id},villaDto);

}

### Documentar Respuestas

Se le agregan marcas según cada posible tipo de respuesta

[HttpGet("id:int")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<VillaDto> GetVilla(int id){..

### Asignar nombre al endpoint

[HttpGet("id:int",Name = "GetVilla")]

Luego en otro endpoint se puede redirigir a este con

return CreatedAtRoute("GetVilla",new { id=villaDto.Id},villaDto);

### Post

Se marcan con [HttpPost]

Y el paramatro que corresponda al cuerpo del body debe ser una clase de properties y debe estar marcado en el método con [FromBody]

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public ActionResult<VillaDto> CrearVilla([FromBody] VillaDto villaDto)

{

if (villaDto == null)

{

return BadRequest(villaDto);

}

if (villaDto.Id>0)

{

return StatusCode(StatusCodes.Status500InternalServerError);

}

villaDto.Id = VillaStore.villaList.OrderByDescending(v => v.Id).FirstOrDefault().Id+1;

VillaStore.villaList.Add(villaDto);

return CreatedAtRoute("GetVilla",new { id=villaDto.Id},villaDto);

}

### Revisar Validación Modelo

#### Error de modelo

Dentro del endpoint post usar

if (!ModelState.IsValid)

{

return BadRequest(villaDto);

}

Como ves no se especifica el modelo, aparentemente el reconoce si es un post el modelo del cuerpo

#### Error personalizado

Para personalizar la salida de un error

if (VillaStore.villaList.FirstOrDefault(v=>v.Name.ToLower()==villaDto.Name.ToLower())!=null) {

ModelState.AddModelError("NombreExiste", "Este nombre ya existe");

return BadRequest(ModelState);

}

Responde con:

**{**

**"NombreExiste": [**

**"Este nombre ya existe"**

**]**

**}**

Fijarse que el primer parámetro es el key y el otro uno de los contenidos

Además lo que hay que pasarle como argumento a BadRequest es ModelState

### Delete

Pro alguna razon necesita que el parámetro este marcado dentro de {} en "{id:int}"

[HttpDelete("{id:int}")]

[ProducesResponseType(StatusCodes.Status204NoContent)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult DeleteVilla(int id)

{

if (id == 0)

{

return BadRequest();

}

var villa = VillaStore.villaList.FirstOrDefault(v => v.Id == id);

if (villa == null)

{

return NotFound();

}

VillaStore.villaList.Remove(villa);

return NoContent();

}