[https://localhost:{PORT}/pizza](https://localhost:%7bPORT%7d/pizza)

|  |  |  |
| --- | --- | --- |
| Get all pizzas  [https://localhost:{PORT}/pizza](https://localhost:%7bPORT%7d/pizza) | | [HttpGet]  public ActionResult<List<Pizza>> GetAll() =>  PizzaService.GetAll(); |
| Get one pizza  [https://localhost:{PORT}/pizza](https://localhost:%7bPORT%7d/pizza)/1 | | [HttpGet("{id}")]  public ActionResult<Pizza> Get(int id)  {  var pizza = PizzaService.Get(id);  if(pizza == null)  return NotFound();  return pizza;  } |
| Add | [HttpPost]  public IActionResult Create(Pizza pizza)  {  PizzaService.Add(pizza);  return CreatedAtAction(nameof(Create), new { id = pizza.Id }, pizza);  } | |
| Update | [HttpPut("{id}")]  public IActionResult Update(int id, Pizza pizza)  {  if (id != pizza.Id)  return BadRequest();    var existingPizza = PizzaService.Get(id);  if(existingPizza is null)  return NotFound();    PizzaService.Update(pizza);    return NoContent();  } | |
| Delete | [HttpDelete("{id}")]  public IActionResult Delete(int id)  {  var pizza = PizzaService.Get(id);    if (pizza is null)  return NotFound();    PizzaService.Delete(id);    return NoContent();  } | |

ApplicationDbContext.cs

**public** **class** ApplicationDbContext : DbContext

{

**public** ApplicationDbContext(DbContextOptions options) : **base**(options)

    {

    }

**public** DbSet<Autor> Autor { **get**; **set**; }

//el nombre de esta propiedad será el que tenga la tabla generada en la BD

}

Startup.cs

**public** **void** ConfigureServices(IServiceCollection services)

{

   services.AddControllers();

   //Configurando la conexión a la BD y el contexto de datos

   services.AddDbContext<ApplicationDbContext>(options =>

     options.UseSqlServer(Configuration.GetConnectionString("defaultConnection")));

   services.AddSwaggerGen(c =>

    {

       c.SwaggerDoc("v1", **new** OpenApiInfo { Title = "WebAPIBiblioteca", Version = "v1" });

     });

}

CONTROLADOR

[HttpPost]

**public** **async** Task<ActionResult> Post(Autor autor)

{

    context.Add(autor);

**await** context.SaveChangesAsync();

**return** Ok();

}

[HttpPut("{id}")]

public IActionResult Update(int id, Pizza pizza)

{

if (id != pizza.Id)

return BadRequest();

var existingPizza = PizzaService.Get(id);

if(existingPizza is null)

return NotFound();

PizzaService.Update(pizza);

return NoContent();

}

[HttpPut("{id:int}")]//especificamos un parámetro de ruta

**public** **async** Task<ActionResult> Put(Autor autor, **int** id)

{

**if** (autor.Id != id)

**return** BadRequest("El id del autor no coincide");

**var** existe = **await** context.Autor.AnyAsync(x => x.Id == id);

**if** (!existe)

**return** NotFound();

    context.Update(autor);

**await** context.SaveChangesAsync();

**return** Ok();

}