# PASOS QUE VAMOS A SEGUIR.

En las siguientes instrucciones no están todos los pasos contados sino que he añadido aquellos trozos de código que voy a ir indicando que tenéis que copiar y pegar en cada punto.

Primero de todo vamos a facilitarnos la vida con la opción de Realizar seguimiento del elemento activo en el Explorador de soluciones. Para ello:

Sobre solución -→ Herramientas → Opciones →Proyectos y soluciones → General
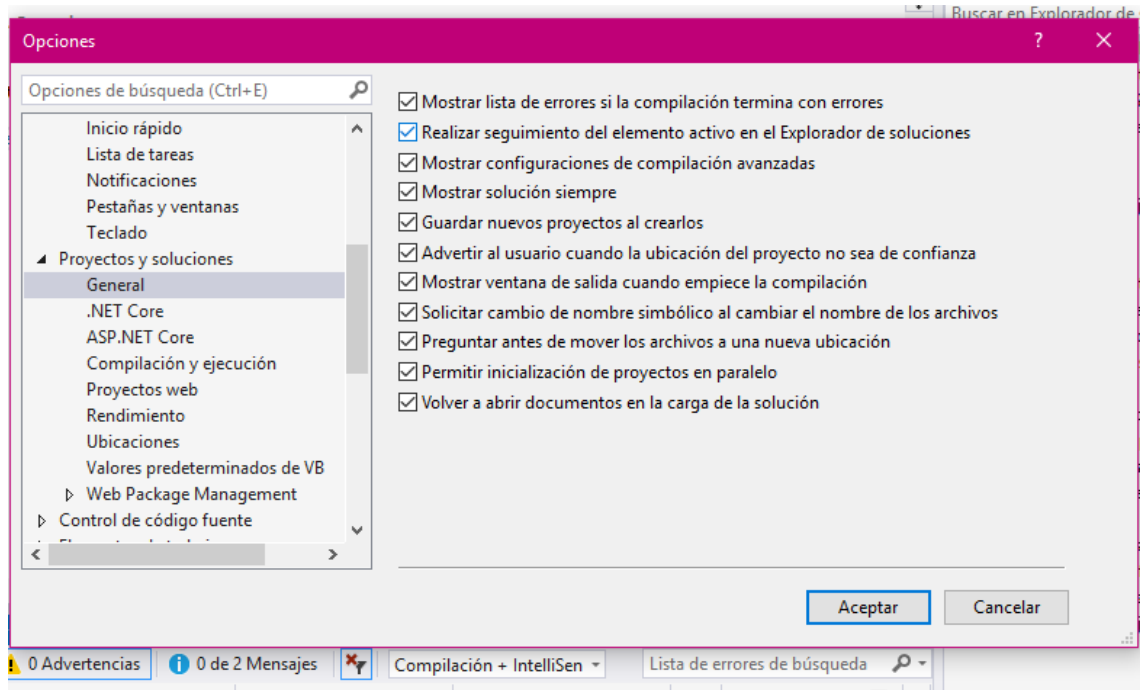
Paquetes a bajar desde Nuget (Ya añadidos a proyecto base y proyecto con modelo)

Pomelo.EntityFrameworkCore.MySql

Microsoft.EntityFrameworkCore.Design

Microsoft.Extensions.Configuration.Json

Conexión a bbdd <u>tras establecer como proyecto de inicio y seleccionarlo (Ya realizado en proyecto con modelo)</u>

Scaffold-DbContext "Server=localhost;User Id=root;Password=Afaya_2010;Database=animalesfantasticos" Pomelo.EntityFrameworkCore.MySql -OutputDir Models

A partir de aquí ya comenzamos todos:

Appsettings.json

```json
{
 "ConnectionStrings": {
   "conexionDatabase": "Server=localhost;User Id=root;Password=Afaya_2010;Database=animalesfantasticos"
 }
}
```

DbContext

```csharp
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
       if (!optionsBuilder.IsConfigured)
       {
          IConfigurationRoot configuration = new ConfigurationBuilder()
             .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)
             .AddJsonFile("appsettings.json")
             .Build();
          optionsBuilder.UseMySql(configuration.GetConnectionString("conexionDatabase"));
       }
    }
```

Repositorios

```csharp
public interface IAnimalesRepository: IDisposable
    {
        List<Animales> GetAnimales();
        Animales GetAnimalById(int animalId);
        void InsertAnimal(Animales animalToAdd);
        void DeleteAnimal(int animalId);
        void UpdateAnimal(Animales animalToUpdate);
        void saveChanges();
    }
```

```csharp
public class AnimalesRepository : IAnimalesRepository
{
    private animalesfantasticosContext context;
    private bool disposed = false;

    public AnimalesRepository(animalesfantasticosContext context)
    {
        this.context = context;
    }

    public void DeleteAnimal(int animalId)
    {
        Animales animalToDelete = context.Animales.Find(animalId);
        context.Animales.Remove(animalToDelete);
        saveChanges();
    }


    public Animales GetAnimalById(int animalId)
    {
        Animales animalObtained = context.Animales.Find(animalId);

        return animalObtained;
    }


    public List<Animales> GetAnimales()
    {
        List<Animales> allAnimales = context.Animales.ToList();

        return allAnimales;
    }

    public void InsertAnimal(Animales animalToAdd)
    {
        context.Animales.Add(animalToAdd);
        saveChanges();
    }

    public void saveChanges()
    {
        context.SaveChanges();
    }

    public void UpdateAnimal(Animales animalToUpdate)
    {
        context.Animales.Update(animalToUpdate);
        saveChanges();
    }
}

    protected virtual void Dispose(bool disposing)
    {
        if (!this.disposed)
        {
            if (disposing)
            {
                context.Dispose();
            }
        }
        this.disposed = true;
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
```

```
        }
    }
```

DTOs

Atajo prop

AutoMapper

```csharp
public class AnimalesDTO
    {
        public AnimalesDTO()
        {

        }

        public int id { get; set; }
        public string nombre { get; set; }
        public string especie { get; set; }
        public int? edad { get; set; }

        public bool isJunior
        {
            get
            {
                return edad < 3;
            }
            set { }
        }

        public bool isSenior
        {
            get
            {
                return edad > 10;
            }
            set { }
        }

        public AnimalesDTO createAnimalesDTO(Animales animalToCreate)
        {
            AnimalesDTO animal = new AnimalesDTO();
            animal.nombre = animalToCreate.Nombre;
            animal.especie = animalToCreate.Especie;
            animal.edad = animalToCreate.Edad;
            animal.id = animalToCreate.Id;

            return animal;
        }

        public Animales createAnimales(AnimalesDTO animalDTOToCreate)
        {
            Animales animal = new Animales();
            animal.Nombre = animalDTOToCreate.nombre;
            animal.Especie = animalDTOToCreate.especie;
            animal.Edad = animalDTOToCreate.edad;
            animal.Id = animalDTOToCreate.id;

            return animal;
        }

    }
```

MANAGERS

```
public interface IAnimalesManager
    {
        List<AnimalesDTO> GetAnimalesList();
        AnimalesDTO GetAnimalByName(string animalName);
        void UpdateAnimal(AnimalesDTO animalToUpdate);
        void DeleteAnimalById(int animalId);
        void CreateAnimal(AnimalesDTO animalToCreate);
    }
```

```csharp
public class AnimalesManager: IAnimalesManager
    {
        public IAnimalesRepository animalesRepository;

        public AnimalesManager()
        {
            animalesRepository = new AnimalesRepository(new
animalesfantasticosContext());
        }

        public AnimalesManager(IAnimalesRepository animalesRepository)
        {
            this.animalesRepository = animalesRepository;
        }

        public void DeleteAnimalById(int animalId)
        {
            animalesRepository.DeleteAnimal(animalId);
        }

        public AnimalesDTO GetAnimalByName(string animalName)
        {
            AnimalesDTO animalResult = null;

            List<Animales> allAnimales = animalesRepository.GetAnimales();
            Animales currentAnimal = allAnimales.Find(x => x.Nombre ==
animalName);
            if (currentAnimal !=null)
            {
                animalResult = new AnimalesDTO()
                        .createAnimalesDTO(currentAnimal);
            }

            return animalResult;
        }

        public List<AnimalesDTO> GetAnimalesList()
        {
            List<AnimalesDTO> animalesList = new List<AnimalesDTO>();

            List<Animales> allAnimales = animalesRepository.GetAnimales();
                foreach(var animal in allAnimales)
                {
                    AnimalesDTO currentAnimal = new AnimalesDTO()
                                .createAnimalesDTO(animal);
                    animalesList.Add(currentAnimal);
                }


            return animalesList;
        }

        public void UpdateAnimal(AnimalesDTO animalToUpdate)
        {
            Animales animal = new AnimalesDTO().createAnimales(animalToUpdate);
            animalesRepository.UpdateAnimal(animal);
        }

        public void CreateAnimal(AnimalesDTO animalToCreate)
        {
            Animales animal = new AnimalesDTO().createAnimales(animalToCreate);
            animalesRepository.InsertAnimal(animal);
```
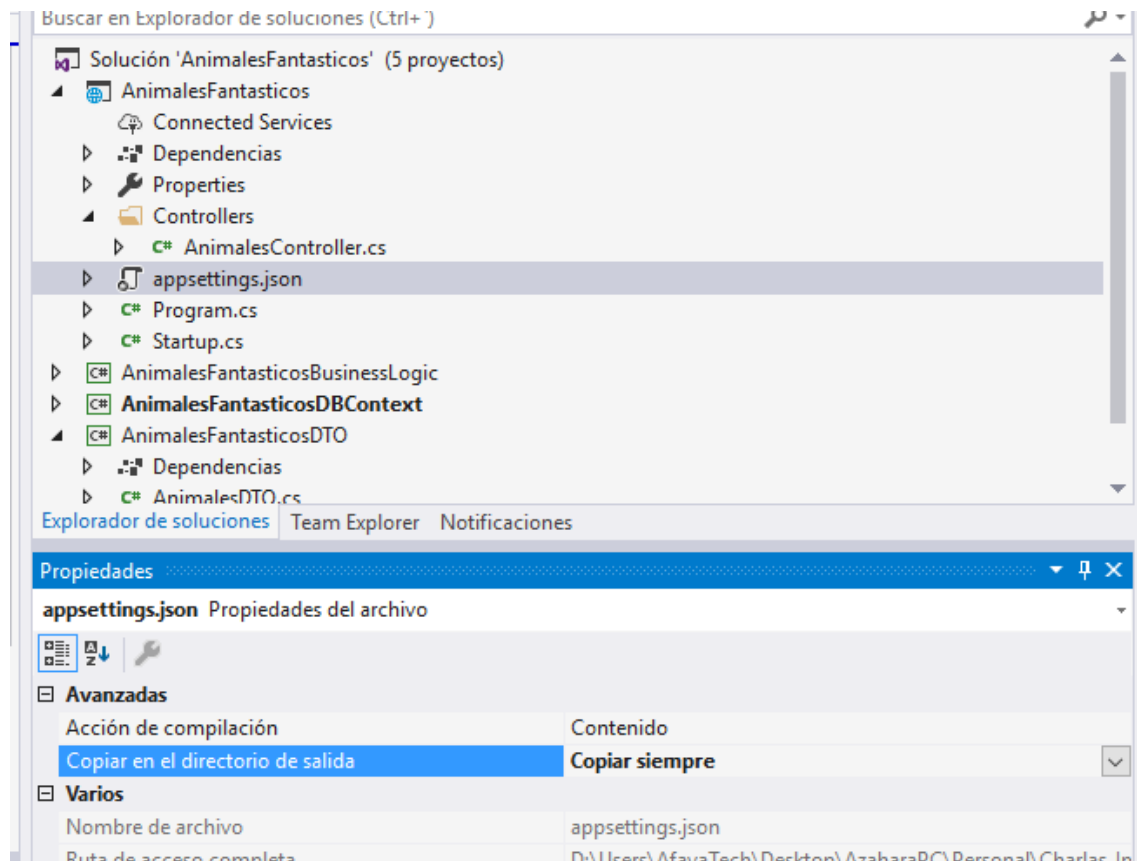
```
        }
}
```

CONTROLLERS


```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "conexionDatabase": "Server=localhost;User
Id=root;Password=Afaya_2010;Database=animalesfantasticos"
  }

}
```

```csharp
[Route("api/[controller]")]
[ApiController]
public class AnimalesController : ControllerBase
{
    public IAnimalesManager animalesManager = new AnimalesManager();

    // GET api/animales
    [HttpGet]
    public ActionResult<List<AnimalesDTO>> Get()
    {
        List<AnimalesDTO> resultList = new List<AnimalesDTO>();
        try
        {
            resultList = animalesManager.GetAnimalesList();
            return resultList;

        }
        catch (Exception ex)
        {
            return BadRequest("Error del servidor");
        }
    }

    // GET api/animales/name
    [HttpGet("{name}")]
    public ActionResult<AnimalesDTO> Get(string name)
    {
        AnimalesDTO result = null;

        try
        {
            result = animalesManager.GetAnimalByName(name);
            return result;
        }
        catch(Exception exception)
        {
            return BadRequest("Error del servidor");
        }
    }

    // POST api/animales
    [HttpPost]
    public ActionResult<bool> Post([FromBody] AnimalesDTO animal)
    {
        bool isOK = true;
        try
        {
            animalesManager.CreateAnimal(animal);
            return isOK;
        }
        catch (Exception exception)
        {
            return BadRequest("Error del servidor");
        }
    }
```

```csharp
        // PUT api/animales/5
        [HttpPut("{id}")]
        public ActionResult<bool> Put(int id, [FromBody] AnimalesDTO animal)
        {
            bool isOK = true;
            try
            {
                animalesManager.UpdateAnimal(animal);
                return isOK;
            }
            catch (Exception exception)
            {
                return BadRequest("Error del servidor");
            }
        }

        // DELETE api/animales/5
        [HttpDelete("{id}")]
        public ActionResult<bool> Delete(int id)
        {
            bool isOK = true;
            try
            {
                animalesManager.DeleteAnimalById(id);
                return isOK;
            }
            catch (Exception exception)
            {
                return BadRequest("Error del servidor");
            }
        }
    }
```
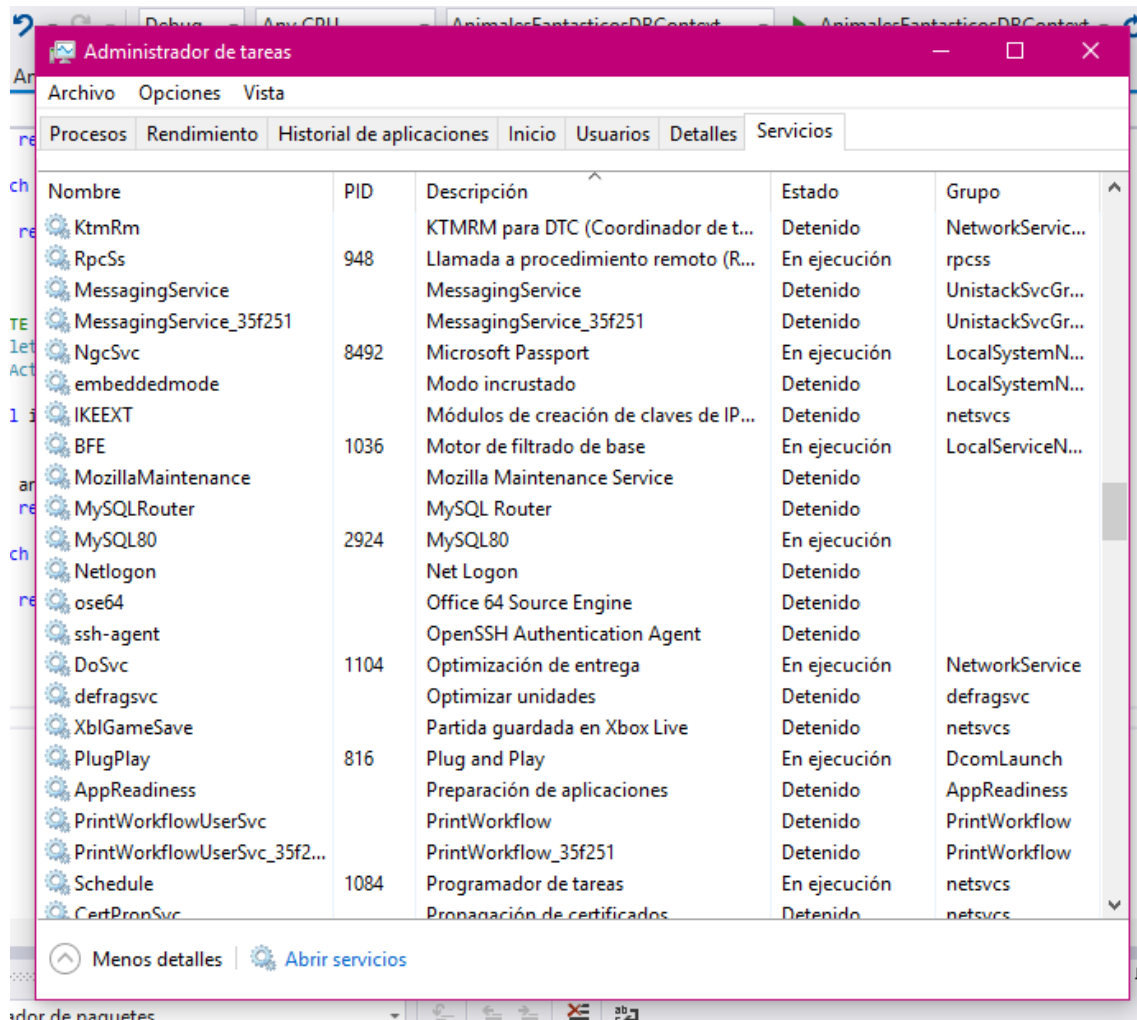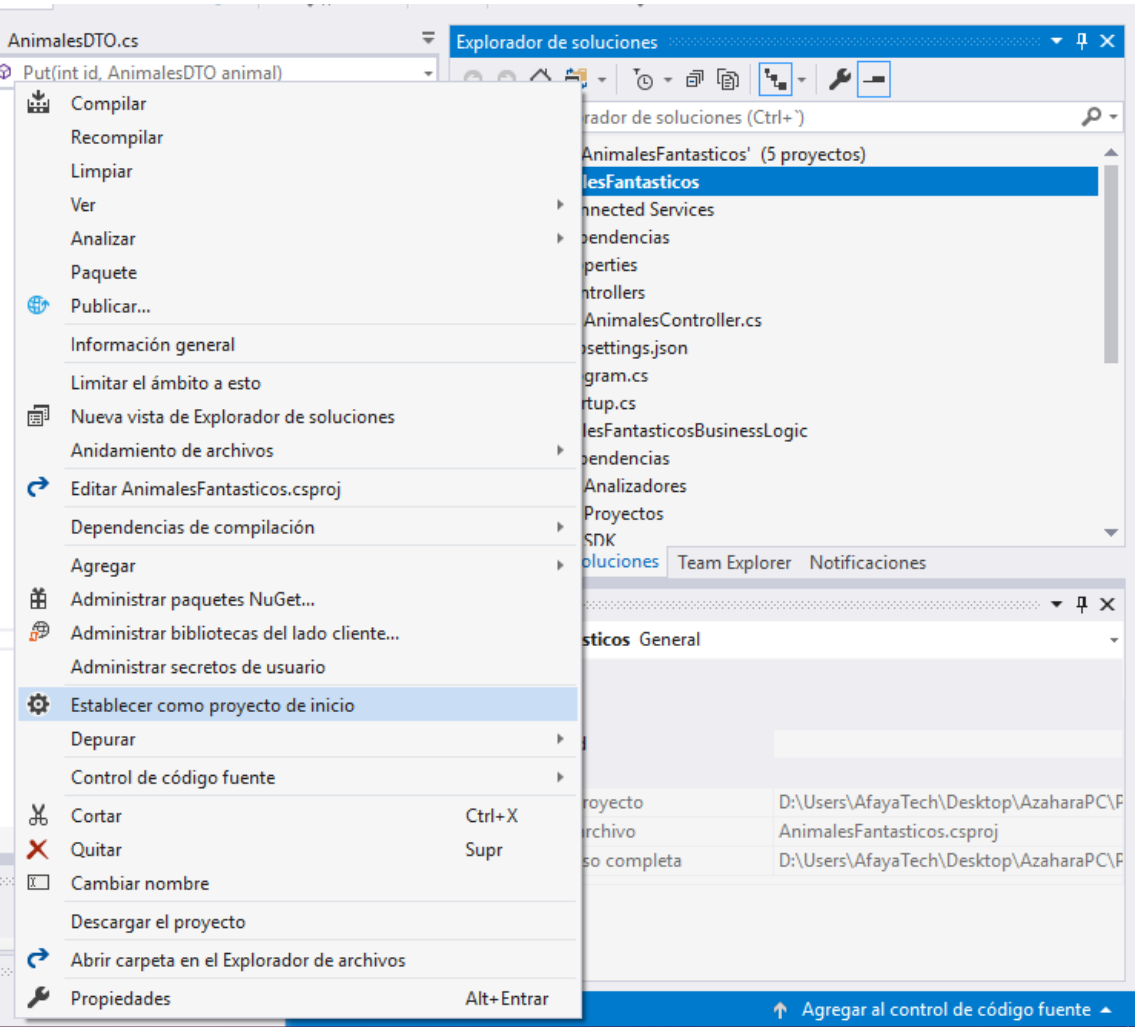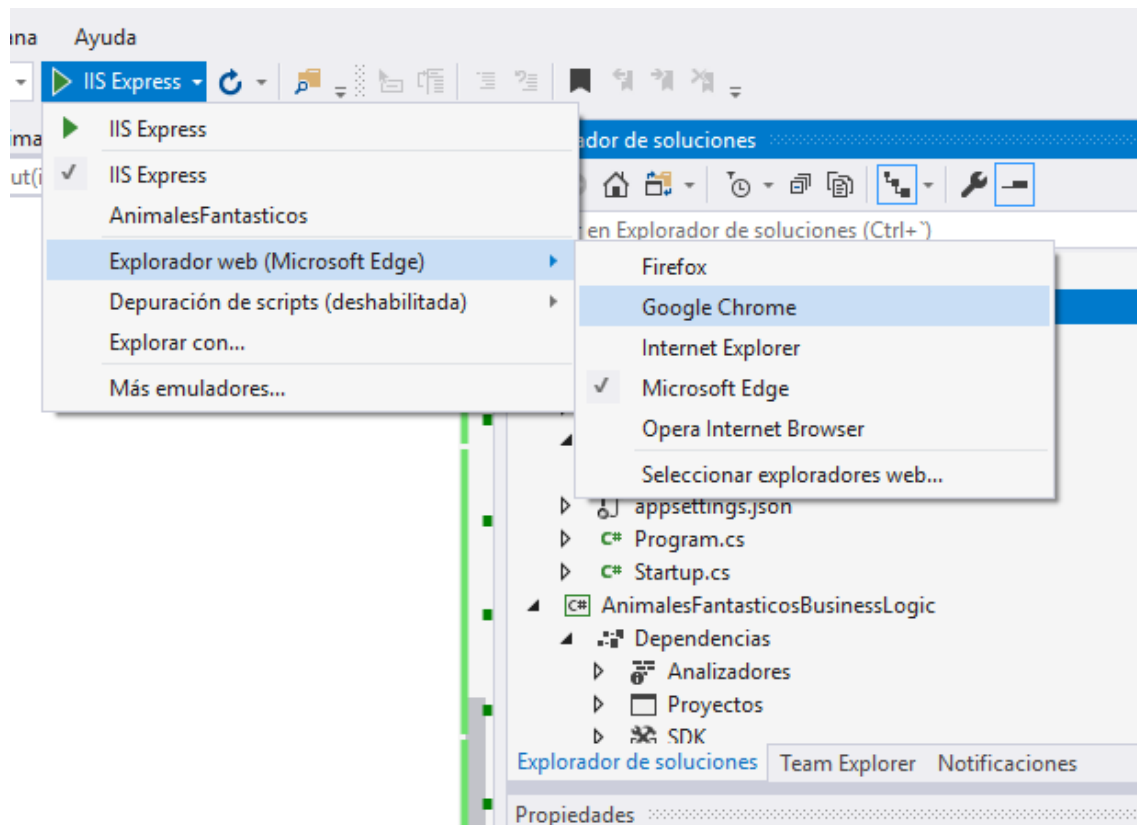
COMPROBAR QUE MYSQL ESTÁ CORRIENDO

ESTABLECER COMO PROYECTO DE INICIO

ELEGIR NAVEGADOR

NAVEGAR

https://localhost:44305/api/animales

https://localhost:44305/api/animales/Occamy

https://localhost:44305/api/animales/Ewok

SWAGGER

Swashbuckle.AspNetCore

```
public class SwaggerConfiguration
    {

        /// <summary>
        /// <para>Foo API v1</para>
        /// </summary>
        public const string EndpointDescription = "WeCodeFest example";

        /// <summary>
        /// <para>/swagger/v1/swagger.json</para>
        /// </summary>
        public const string EndpointUrl = "/swagger/v1/swagger.json";
```

```csharp
        /// <summary>
        /// <para>Jorge Serrano</para>
        /// </summary>
        public const string ContactName = "Azahara Fernandez";

        /// <summary>
        /// <para>http://afaya.es</para>
        /// </summary>
        public const string ContactUrl = "http://afaya.es";

        /// <summary>
        /// <para>v1</para>
        /// </summary>
        public const string DocNameV1 = "v1";

        /// <summary>
        /// <para>Foo API</para>
        /// </summary>
        public const string DocInfoTitle = "Taller .Net Core";

        /// <summary>
        /// <para>v1</para>
        /// </summary>
        public const string DocInfoVersion = "v1";

        /// <summary>
        /// <para>Foo Api - Sample Web API in ASP.NET Core 2</para>
        /// </summary>
        public const string DocInfoDescription = "Descubriendo .Net Core en
un taller de WeCodeFest";

    }
```

Luego en el AppStartup, al inicio del método Configure:

```csharp
// Enable middleware to serve generated Swagger as a JSON endpoint.
        app.UseSwagger();

        // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
specifying the Swagger JSON endpoint.
        app.UseSwaggerUI(c =>
        {
            c.SwaggerEndpoint(SwaggerConfiguration.EndpointUrl,
SwaggerConfiguration.EndpointDescription);
        });
```

Al final del método ConfigureServices:

```csharp
// Register the Swagger generator, defining one or more Swagger documents
        services.AddSwaggerGen(swagger =>
        {
            var contact = new Contact() { Name =
SwaggerConfiguration.ContactName, Url = SwaggerConfiguration.ContactUrl };
            swagger.SwaggerDoc(SwaggerConfiguration.DocNameV1,
                        new Info
                        {
                            Title = SwaggerConfiguration.DocInfoTitle,
                            Version =
SwaggerConfiguration.DocInfoVersion,
```

```
                                    Description =
SwaggerConfiguration.DocInfoDescription,
                                    Contact = contact
                        }
                     );
        });
```

Al ejecutar:

https://localhost:44305/swagger/index.html

TEST UNITARIOS

Crear proyecto biblioteca clases tipo nunit test e instalar Moq

```
public class Tests
    {
        private Mock<IAnimalesRepository> mockAnimalesRepository;
        private List<Animales> mockAnimalesList;
        private Animales animalToAdd;
        private Animales animalToUpdate;

        [SetUp]
        public void Setup()
        {
            mockAnimalesRepository = new Mock<IAnimalesRepository>();
            fillmockAnimalesList();
            fillAnimalToAdd();
            fillAnimalToUpdate();
            mockAnimalesRepository.Setup(ar =>
ar.GetAnimales()).Returns(mockAnimalesList);
            mockAnimalesRepository.Setup(ar =>
ar.InsertAnimal(It.IsAny<Animales>()))
                                    .Callback((Animales item) =>
mockAnimalesList.Add(animalToAdd));
            mockAnimalesRepository.Setup(ar =>
ar.UpdateAnimal(It.IsAny<Animales>()))
                                    .Callback((Animales item) =>
mockAnimalesList.Find(x=>x.Id==4).Nombre= animalToUpdate.Nombre);
            mockAnimalesRepository.Setup(ar => ar.DeleteAnimal(It.IsAny<int>()))
                                    .Callback((int id) =>
mockAnimalesList.Remove(mockAnimalesList.Find(x=>x.Id== id)));
        }


        private void fillmockAnimalesList()
        {
            mockAnimalesList = new List<Animales>();
            for(int i=1; i<6; i++)
            {
                Animales animal = new Animales
                {
                    Nombre = "Animal "+i,
                    Edad = 5,
                    Especie = "Especie "+i,
                    Id = i
                };

                mockAnimalesList.Add(animal);
            }

        }

        private void fillAnimalToAdd()
        {
            animalToAdd = new Animales
            {
                Nombre = "Animal 6",
                Edad = 6,
                Especie = "Especie 6",
                Id = 6
            };
        }
```

```csharp
        private void fillAnimalToUpdate()
        {
            animalToUpdate = new Animales
            {
                Nombre = "Animal Updated",
                Edad = 4,
                Especie = "Especie 4",
                Id = 4
            };
        }

        [Test]
        public void TestGetAnimalesIsOK()
        {
            //Arrange
            AnimalesManager animalesManager = new
AnimalesManager(mockAnimalesRepository.Object);
            //Act
            List<AnimalesDTO> animalesResult = animalesManager.GetAnimalesList();
            //Assert
            Assert.AreEqual(5, animalesResult.Count, "No se ha obtenido el número
esperado de animales");
        }

        [Test]
        public void TestGetAnimalByNameIsOK()
        {
            //Arrange
            AnimalesManager animalesManager = new
AnimalesManager(mockAnimalesRepository.Object);
            string animalTestName = "Animal 5";
            //Act
            AnimalesDTO animalResult =
animalesManager.GetAnimalByName(animalTestName);
            //Assert
            Assert.IsNotNull(animalResult, "El animal no ha sido encontrado");
        }

        [Test]
        public void TestCreateAnimalIsOK()
        {
            //Arrange
            AnimalesManager animalesManager = new
AnimalesManager(mockAnimalesRepository.Object);
            //Act
            animalesManager.CreateAnimal(new AnimalesDTO());
            AnimalesDTO animalResult = animalesManager.GetAnimalByName("Animal
6");

            //Assert
            Assert.IsNotNull(animalResult, "El animal no ha sido encontrado");
        }

        [Test]
        public void TestUpdateAnimalIsOK()
        {
            //Arrange
            AnimalesManager animalesManager = new
AnimalesManager(mockAnimalesRepository.Object);
            //Act
            animalesManager.UpdateAnimal(new AnimalesDTO());
```

```csharp
            AnimalesDTO animalResult = animalesManager.GetAnimalByName("Animal
Updated");

            //Assert
            Assert.IsNotNull(animalResult, "El animal no ha sido encontrado");

        }

        [Test]
        public void TestDeleteAnimalIsOK()
        {
            //Arrange
            AnimalesManager animalesManager = new
AnimalesManager(mockAnimalesRepository.Object);
            int animalToDeleteId = 2;
            //Act
            animalesManager.DeleteAnimalById(animalToDeleteId);
            AnimalesDTO animalResult = animalesManager.GetAnimalByName("Animal
2");

            //Assert
            Assert.IsNull(animalResult, "El animal ha sido encontrado");

        }
    }
```