GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Unión Europea
Una manera de hacer Europa

Entorno tecnológico y lenguajes de programación

Juan Valer y Javier Ruiz
RIAM Intelearning Lab – GNOSS
juanvaler@gnoss.com y javierruiz@gnoss.com

HĒRCULES

Entorno tecnológico y lenguajes de programación

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Introducción

- Plataforma de desarrollo: .Net Core (Versión 3.1)
- Lenguaje de desarrollo: C#
- Entorno de desarrollo: Visual Studio
- Base de datos SQL: PostgreSQL (En el taller usaremos MicrosoftSQL Server Express LocalDB)
- Servidores de backend: Linux CentOS, Apache y Docker

.Net

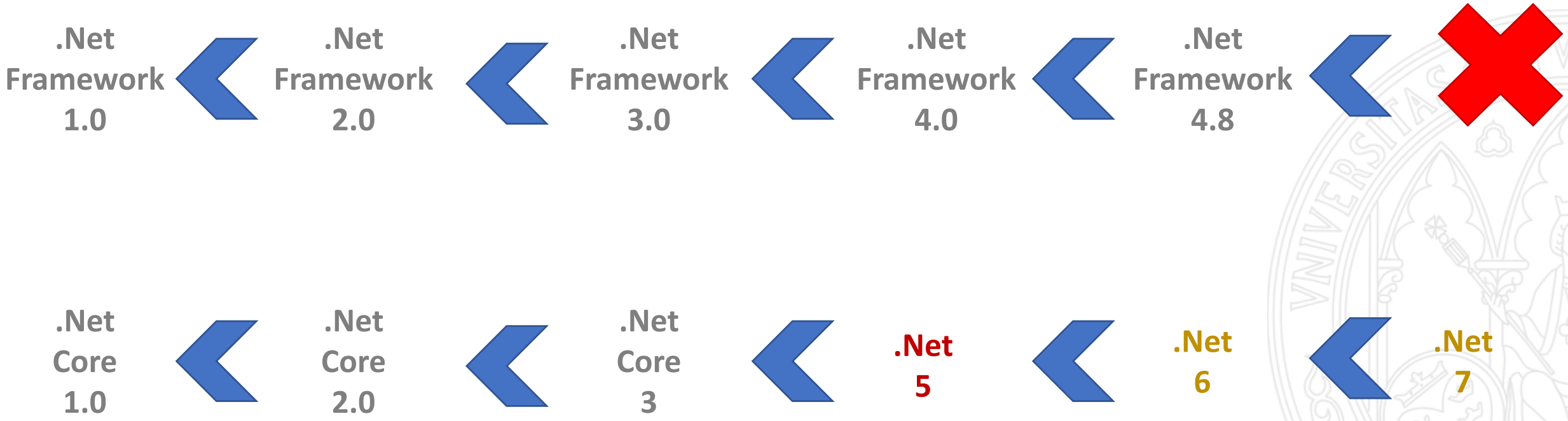
.Net Framework es una tecnología que permite la creación y ejecución de servicios Web y aplicaciones Windows. El diseño de .Net Framework está enfocado a cumplir los objetivos siguientes:

- Entorno de programación orientado a objetos
- Entorno de ejecución que minimiza los conflictos, fomenta la ejecución segura de código y elimina los problemas de rendimiento de los entornos basados en código interpretado
- Ofrece una experiencia coherente entre tipos de aplicaciones muy diferentes (demonios, aplicaciones de escritorio, de consola, web, api…)
- Basa toda la comunicación en estándares

.Net y .Net CORE

- .Net Framework sólo funciona en máquinas Windows. Hay una implementación de .Net Framework para Linux llamada Mono, pero no es oficial
- Microsoft anuncia en Noviembre de 2014 la creación de .Net Core, en un esfuerzo por incluir soporte multiplataforma para .Net
- .Net Core 1.0 se libera en Junio de 2016
- Ahora sí, .Net CORE funciona en máquinas Windows, Linux y MacOS.
- .Net Core 2.0 se libera en Agosto de 2017
- .Net Core 3 se libera en Septiembre de 2019
- .Net 5 se libera en Noviembre de 2020

.Net y .Net CORE



.Net y .Net CORE

- .Net Framework 4.8 es la última versión de .Net Framework que se va a publicar. Sólo se publicarán parches de seguridad.
- A partir de .Net 5, la única plataforma en la que va a trabajar Microsoft es .Net Core (Ahora .Net a secas)

.Net CORE

- Es open source
- Es multiplataforma
- Es modular
- Muestra mejor rendimiento
- El futuro de .NET avanza hacia .NET Core → .NET 5 es la nueva versión de .NET Core
- Orientado a construir aplicaciones basadas en microservicios: Se puede desplegar sobre contenedores Docker en Linux

Por qué usar .Net CORE

- Teníamos más de 10 años de experiencia en .Net
- Es una plataforma sólida y robusta
- Tiene una gran comunidad por detrás
- Código optimizado con alto rendimiento
- Velocidad de desarrollo
- Facilidad de mantenimiento de las aplicaciones

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

C#: Lenguaje nativo de la plataforma .Net

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

C#

- Es un lenguaje de programación multiparadigma orientado a objetos
- Su sintáxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .Net, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes

C#

Características principales:

- Programación multi-hilo y asíncrona sencilla
- Propiedades con métodos get/set
- Expresiones Lambda y LINQ
- Soporte de delegados y funciones anónimas
- Métodos de extensión
- Garbage collector
- Compatibildad con versiones anteriores

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Unión Europea
Una manera de hacer Europa

C#: Programación multihilo

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        Task forTask = Task.Factory.StartNew(Contar);

        forTask.Wait();
    }

    1 referencia
    static void Contar()
    {
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine($"Item {i}");
        }
    }
}
```

C#: Programación multihilo (con funciones anónimas)

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        Task forTask = Task.Factory.StartNew(() => { for (int i = 0; i < 10; i++) { Console.WriteLine($"Item {i}"); } });

        forTask.Wait();
    }
}
```

C#: Programación multihilo avanzada

```
class Program
{
    static CancellationTokenSource cancellationToken = new CancellationTokenSource();
    0 referencias
    static void Main(string[] args)
    {
        Task forTask = Task.Factory.StartNew(Contar, cancellationToken.Token).ContinueWith(PintarFin);

        Console.WriteLine("Pulsa cualquier tecla para salir...");
        Console.ReadKey();

        if (!forTask.IsCompleted)
        {
            cancellationToken.Cancel();
        }
    }

    1 referencia
    static void Contar()
    {
        for (int i = 0; i < 10; i++)
        {
            if (cancellationToken.IsCancellationRequested)
                break;
            Console.WriteLine($"Item {i}");
        }
    }

    1 referencia
    static void PintarFin(Task tareaAnterior)
    {
        if (tareaAnterior.IsFaulted)
        {
            Console.WriteLine($"Error: {tareaAnterior.Exception.Message}");
        }

        Console.WriteLine($"Fin");
    }
}
```

C#: Programación asíncrona con async y await

```
static void Main(string[] args)
{
    Contar10Async();

    Console.WriteLine("Pulsa cualquier tecla para salir...");
    Console.ReadKey();

    cancellationToken.Cancel();
}
1 referencia
static async void Contar10Async()
{
    await Contar10();
    Console.WriteLine("Fin Contar10Async");
}
1 referencia
static Task Contar10()
{
    Task forTask = Task.Factory.StartNew(Contar, cancellationToken.Token).ContinueWith(PintarFin);
    return forTask;
}

1 referencia
static void Contar()
{
    for (int i = 0; i < 10; i++)
    {
        if (cancellationToken.IsCancellationRequested)
            break;
        Thread.Sleep(500);
        Console.WriteLine($"Item {i}");
    }
}
```

C#: Se sustituyen los habituales Getters y Setters por propiedades

```
public class Usuario
{
    private string _nombre;
    0 referencias
    public string getNombre()
    {
        return _nombre;
    }

    0 referencias
    public void setNombre(string nombre)
    {
        _nombre = nombre;
    }
}
```

```
public class Usuario
{
    0 referencias
    public string Nombre { get; set; }
}
```

C#: Se sustituyen los habituales Getters y Setters por propiedades

```
public class Usuario
{
    private string mNombre;

    0 referencias
    public string Nombre
    {
        get
        {
            return mNombre;
        }
        set
        {
            if (string.IsNullOrEmpty(mNombre))
            {
                throw new ArgumentNullException("Nombre", "El campo Nombre debe tener valor obligatoriamente");
            }
            mNombre = value;
        }
    }

}
```

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

C#: Expresiones lambda

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        List<Usuario> listaUsuarios = CargarUsuarios();

        var resultado = listaUsuarios.Where(usuario => usuario.Email.Equals("usuario1@mail.com") && usuario.Password.Equals("1234")).Select(usuario => new { usuario.ID, usuario.Nombre });

        Console.WriteLine($"Encontrado: {resultado.FirstOrDefault()?.Nombre}");
    }

    1 referencia
    private static List<Usuario> CargarUsuarios()
    {
        List<Usuario> listaUsuarios = new List<Usuario>();
        for (int i = 0; i < 10; i++)
        {
            listaUsuarios.Add(new Usuario() { Nombre = $"Usuario {i}", Email = $"usuario{i}@mail.com", Password = "1234" });
        }
        return listaUsuarios;
    }
}

4 referencias
public class Usuario
{
    2 referencias
    public string Nombre { get; set; }
    1 referencia
    public int ID { get; set; }

    2 referencias
    public string Password { get; set; }
    2 referencias
    public string Email { get; set; }
}
```

C#: Métodos de extensión

```
public static class StringExtensions
{
    1 referencia
    public static bool IsValidMail(this string possibleMail)
    {
        return possibleMail.Contains("@");
    }
}

0 referencias
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        List<Usuario> listaUsuarios = CargarUsuarios();

        var mailsInvalidos = listaUsuarios.Where(usuario => usuario.Email.IsValidMail());

        foreach(Usuario usuario in mailsInvalidos)
            Console.WriteLine($"Mail incorrecto: {usuario.Email}");
    }
```

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

Docker

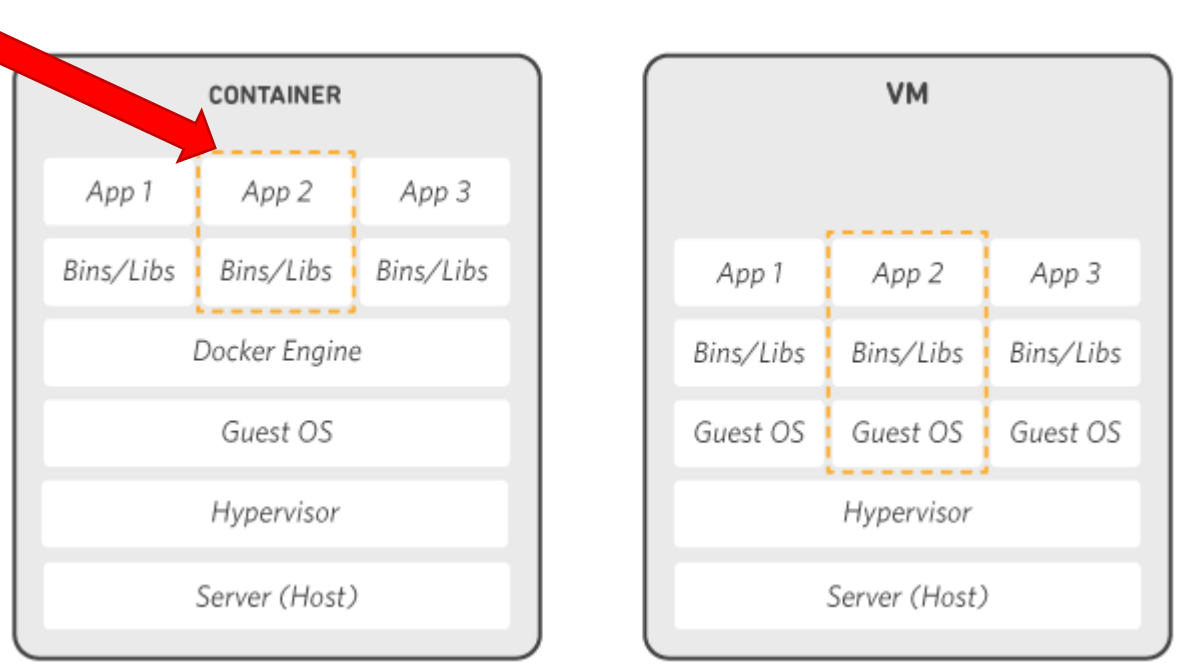
gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Docker

- Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software.
- Proporciona una capa de abstracción y automatización de virtualización de aplicaciones
- Utiliza características de aislamiento de recursos del kernel Linux para permitir que contenedores independientes se ejecuten dentro de una sola instancia Linux.
- Evita la sobrecarga de iniciar y mantener máquinas virtuales

Docker

Cada contenedor contiene una aplicación y todas sus dependencias



Ventajas de Docker

- Si funciona en tu máquina, funciona en producción
- Despliegues más sencillos -> Despliegues más frecuentes
- Estandarización de operaciones: Se despliega igual una tarea en background que una aplicación Web

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
Unión Europea
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa

.Net CORE

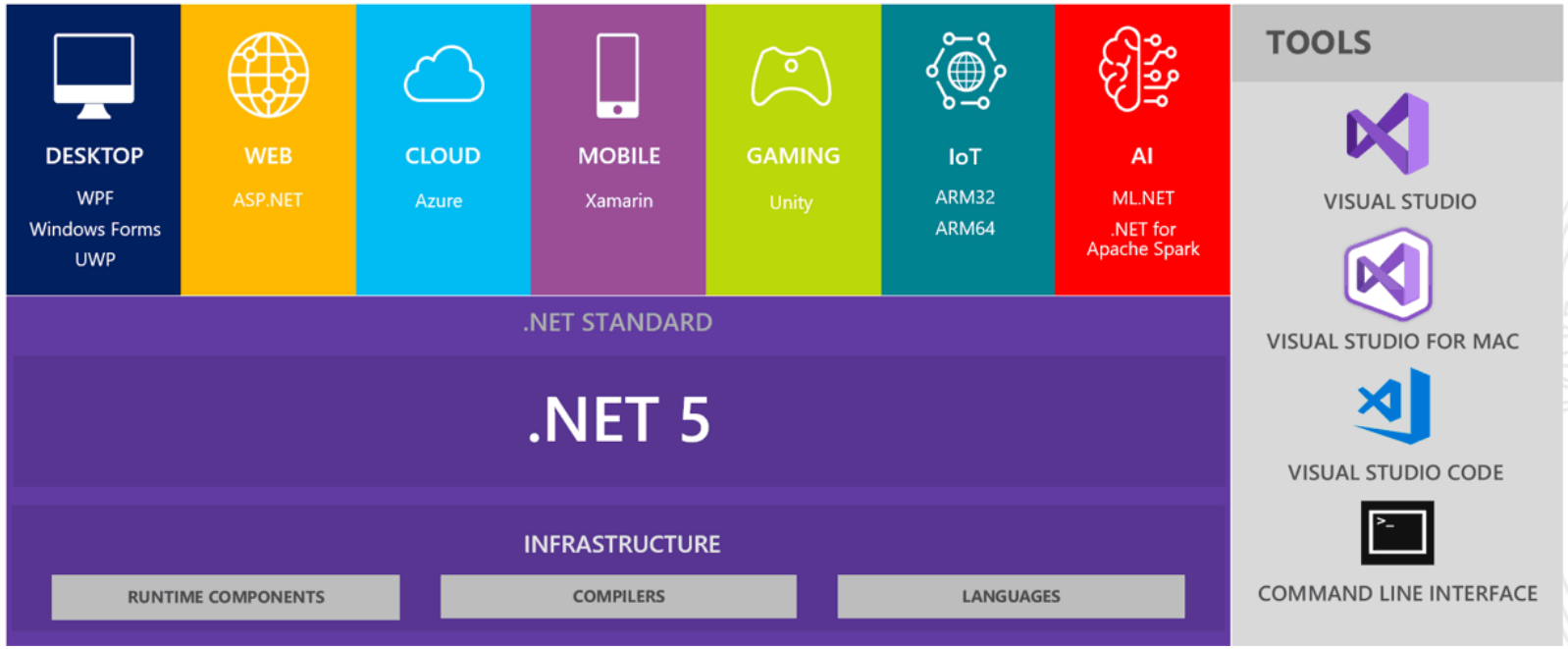
HĒRCULES

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Qué es .Net CORE

- Es la plataforma de desarrollo de Microsoft más moderna, de código abierto, multiplataforma y de alto rendimiento para la creación de todo tipo de aplicaciones.
- Es la unificación de .Net Framework, .Net CORE y Mono.

.Net CORE


DESKTOP
WPF
Windows Forms
UWP
WEB
ASP.NET
CLOUD
Azure
MOBILE
Xamarin
GAMING
Unity
IoT
ARM32
ARM64
AI
ML.NET
.NET for Apache Spark
.NET STANDARD
.NET 5
INFRASTRUCTURE
RUNTIME COMPONENTS
COMPILERS
LANGUAGES
TOOLS
VISUAL STUDIO
VISUAL STUDIO FOR MAC
VISUAL STUDIO CODE
COMMAND LINE INTERFACE

ASP.Net CORE

- Es el módulo de .Net CORE que permite crear aplicaciones Web, IoT, APIs…
- Usa MVC como patrón de desarrollo por defecto
- Integra herramientas de IoC e Inyección de dependencias por defecto, facilitando y fomentando el uso de pruebas unitarias

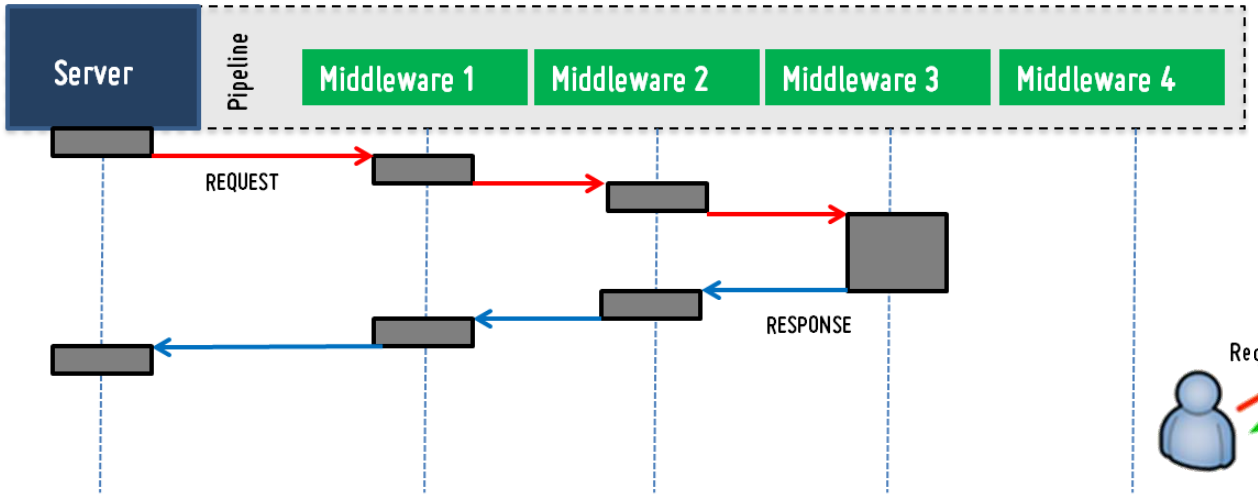
ASP.Net CORE

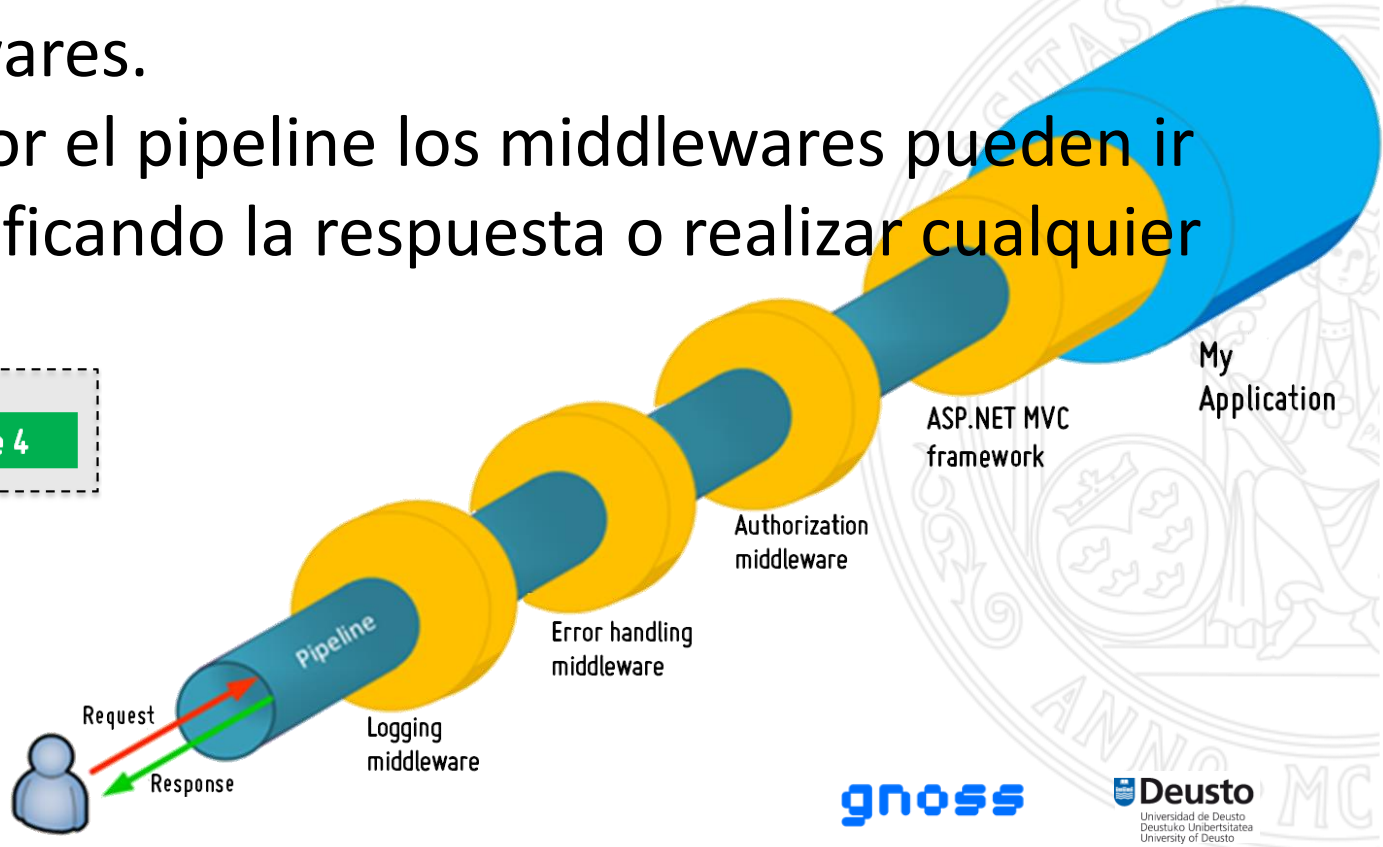
Características:

- Un caso unificado para crear un API y una intefaz de usuario Web
- Razor es el lenguaje de marcado productivo para programar las vistas
- Canalización de solicitudes HTTP ligera, modular y de alto rendimiento.
- Facilidad de integración con frameworks del lado del cliente: Bootstrap, React, Angular…

El pipeline de ASP.Net CORE

- El pipeline es como una tubería, en la que se van colocando anillos, que corresponderían a los middlewares.
- Conforme la petición avanza por el pipeline los middlewares pueden ir consultando sus detalles, modificando la respuesta o realizar cualquier tarea.





ASP.Net CORE: La clase Program

- El punto de inicio de una aplicación ASP.NET es el mismo que el de cualquier otra: el método Main
- El método Main en una aplicación ASP.NET crea y lanza un servidor Web basado en Kestrel
- Se define una clase de Startup para que inicialice todos los componentes necesarios

```
public class Program
{
    0 referencias
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    1 referencia
    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

ASP.Net CORE: La clase Startup

Método Configure:

- Crea la configuración del pipeline.

Método ConfigureServices:

- Configura los servicios de la aplicación que proporciona funcionalidades de la aplicación. Se usan a través de inyección de dependencias.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseMiddleware(typeof(ErrorHandlingMiddleware));
    app.UseMiddleware(typeof(LoadConfigJsonMiddleware));
    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseForwardedHeaders(new ForwardedHeadersOptions
    {
        ForwardedHeaders = ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto
    });

    app.UseAuthorization();

    app.UseSwagger();
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Uris factory");
    });

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

ASP.Net CORE: La clase Startup

Método Configure:

- Crea la configuración del pipeline.

Método ConfigureServices:

- Configura los servicios de la aplicación que proporciona funcionalidades de la aplicación. Se usan a través de inyección de dependencias.

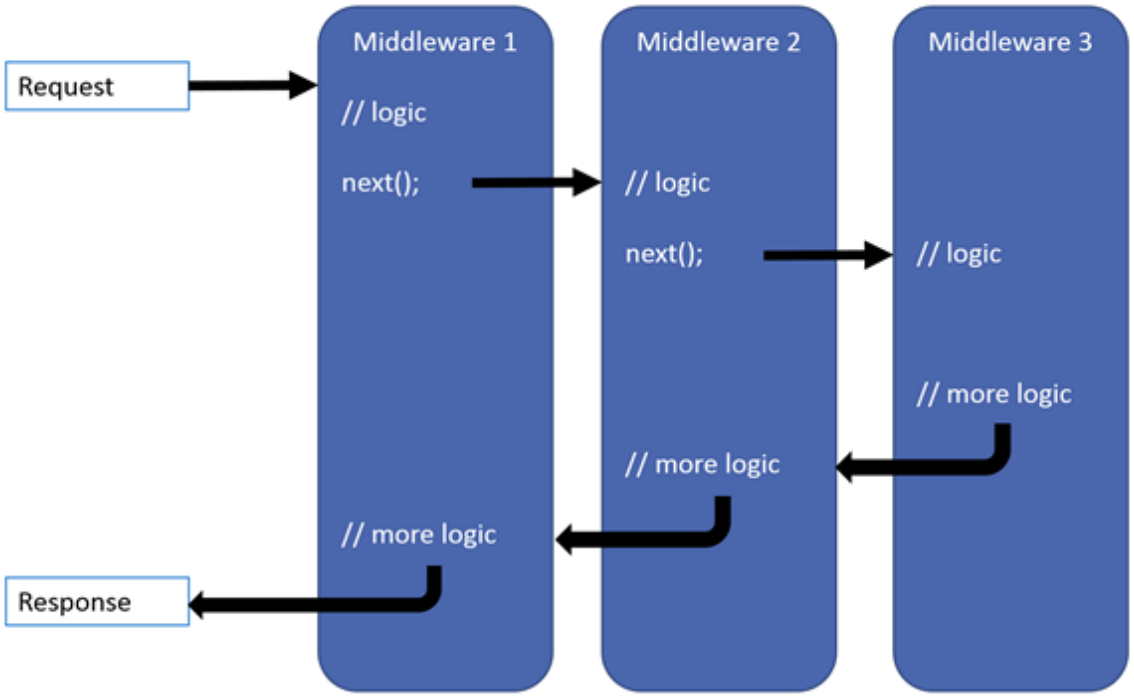
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "Uris factory", Version = "v1" });
        c.OperationFilter<AddParametersFilter>();
    });
    services.Configure<ForwardedHeadersOptions>(options =>
    {
        options.KnownProxies.Add(IPAddress.Parse("127.0.0.1"));
    });

    services.AddSingleton(typeof(ConfigJsonHandler));
    services.AddScoped<ISchemaConfigOperations, SchemaConfigFileOperations>();
}
```

ASP.Net CORE: El middleware

- Es un software que se ensambla en una canalización, para controlar las solicitudes y respuestas. Es el encargado de invocar al siguiente componente.
- Normalmente en una aplicación usaremos varios middlewares componiendo una cadena llamada pipeline
- El orden en el que se configuran en el método Configure de la clase Startup define el orden en el que se invocarán estos middlewares. El orden es importante

ASP.Net CORE: El middleware



GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Unión Europea
Una manera de hacer Europa

ASP.Net CORE MVC: Modelo – Vista – Controlador

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

ASP.Net CORE MVC: Controlador Web

```
public class FilmController : Controller
{
    private readonly ILogger<FilmController> _logger;
    private IFilmService _filmService;

    0 referencias
    public FilmController(ILogger<FilmController> logger, IFilmService filmService)
    {
        _logger = logger;
        _filmService = filmService;
    }

    [HttpGet]
    0 referencias
    public IActionResult List()
    {
        List<Film> listaPeliculas = _filmService.CargarPeliculas();
        return View(listaPeliculas);
    }

    [HttpGet]
    0 referencias
    public IActionResult Index(Guid id)
    {
        List<Film> listaPeliculas = _filmService.CargarPeliculas();
        Film film =  listaPeliculas.FirstOrDefault(film => film.Film_ID.Equals(id));
        return View(film);
    }
```

ASP.Net CORE MVC: Controlador API

```
[Route("api/[controller]")]
[ApiController]
1 referencia
public class FilmApiController : ControllerBase
{
    private IFilmService _filmService;

    0 referencias
    public FilmApiController(IFilmService service)
    {
        _filmService = service;
    }

    // GET: api/<FilmController>
    [HttpGet]
    0 referencias
    public IEnumerable<Film> Get()
    {
        List<Film> listaPeliculas = _filmService.CargarPeliculas();
        return listaPeliculas;
    }

    // GET api/<FilmApiController>/D85BE4C7-A042-4CDE-9722-BABC70D6B03B
    [HttpGet("{id}")]
    0 referencias
    public Film Get(Guid id)
    {
        List<Film> listaPeliculas = _filmService.CargarPeliculas();
        return listaPeliculas.FirstOrDefault(film => film.Film_ID.Equals(id));
    }
```

ASP.Net CORE MVC: Modelo

```
public class Film
{

    [Key]
    5 referencias
    public Guid Film_ID { get; set; }
    [Required]
    2 referencias
    public string Title { get; set; }
    [Required]
    2 referencias
    public int Year { get; set; }
    2 referencias
    public string Released { get; set; }
    [Required]
    2 referencias
    public int MinuteRunTime { get; set; }
    [ForeignKey("Director")]
    [Required]
    2 referencias
    public Guid Director_ID { get; set; }
}
```

ASP.Net CORE MVC: Vista

```
@model WebApplication2.Models.Film

@{
    ViewData["Title"] = "View";
}

<h1>View</h1>

<h4>Film</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="View">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Film_ID" class="control-label"></label>
                <input asp-for="Film_ID" class="form-control" />
                <span asp-validation-for="Film_ID" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Year" class="control-label"></label>
                <input asp-for="Year" class="form-control" />
                <span asp-validation-for="Year" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Released" class="control-label"></label>
                <input asp-for="Released" class="form-control" />
                <span asp-validation-for="Released" class="text-danger"></span>
            </div>
```

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

Inyección de dependencias, IoC y Test unitarios

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Inyección de dependencias

- La inyección de dependencias es un patrón de diseño que sirve para disminuir el acoplamiento entre los componentes de la aplicación.
- La inyección de dependencias usa abstracciones (interfaces) en lugar de referencias directas.
- Hace que una clase reciba referencias hacia los componentes que necesite para funcionar, en vez de instanciarlas ella misma.

Sin inyección de dependencias

```
public class InvoiceServices
{
    ...
    public void Remove(int invoiceId)
    {
        using (var invoiceRepository = new InvoiceRepository())
        {
            var removed = invoiceRepository.Remove(invoiceId);
            if (removed)
            {
                var notifier = new EmailNotifier();
                notifier.NotifyAdmin($"Invoice {invoiceId} removed");
            }
        }
    }
}
```

Con inyección de dependencias

```
public class InvoiceServices
{
    private readonly IInvoiceRepository _invoiceRepository;
    private readonly INotifier _notifier;

    public InvoiceServices (IInvoiceRepository invoiceRepository, INoti
notifier)
    {
        _invoiceRepository = invoiceRepository;
        _notifier = notifier;
    }
    ...
    public void Remove(int invoiceId)
    {
        var removed = _invoiceRepository.Remove(invoiceId);
        if (removed)
        {
            _notifier.NotifyAdmin($"Invoice {invoiceId} removed");
        }
    }
}
```

Testing e Inyección de dependencias

Realizar pruebas unitarias de las clases de forma aislada, enviando dependencias falsas o controladas.

```
[ApiController]
[Route("[Controller]")]
1 referencia | jruiz, Hace 5 días | 1 autor, 1 cambio
public class SchemaController : Controller
{
    private ConfigJsonHandler _configJsonHandler;
    private ISchemaConfigOperations _schemaConfigOperations;

    0 referencias | jruiz, Hace 5 días | 1 autor, 1 cambio
    public SchemaController(ConfigJsonHandler configJsonHandler, ISchemaConfigOperations
      schemaConfigOperations)
    {
        _configJsonHandler = configJsonHandler;
        _schemaConfigOperations = schemaConfigOperations;
    }

    [HttpGet(Name="getSchema")]
    0 referencias | jruiz, Hace 5 días | 1 autor, 1 cambio
    public FileResult GetSchema()
    {
        string contentType = _schemaConfigOperations.GetContentType();
        return File(_schemaConfigOperations.GetFileSchemaData(), contentType);
    }
```

```
public class UnitTestConfigOperations
{
    [Fact]
    0 referencias | jruiz, Hace 12 días | 1 autor, 2 cambios
    public void TestGetSchemaFileData()
    {
        ConfigJsonHandler configJsonHandler = new ConfigJsonHandler();
        ISchemaConfigOperations schemaConfigOperations = new SchemaConfigMemoryOperations
          (configJsonHandler);
        var bytesSchema = schemaConfigOperations.GetFileSchemaData();
        var bytesAsString = Encoding.UTF8.GetString(bytesSchema);
        UriStructureGeneral uriSchema = JsonConvert.DeserializeObject<UriStructureGeneral>
          (bytesAsString);
        bool correctGenerated = uriSchema != null && uriSchema.Base != null && uriSchema.Characters !
          = null && uriSchema.ResourcesClasses != null && uriSchema.ResourcesClasses.Count > 0 &&
          uriSchema.UriStructures != null && uriSchema.UriStructures.Count > 0;
        Assert.True(correctGenerated);
    }
```

GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

Acceso a datos: Entity Framework CORE

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Entity Framework CORE

- ¿Qué es?: Es el ORM para la plataforma de desarrollo para .Net Core, Un ORM es un tipo de biblioteca de acceso a datos que intenta hacer que esta tarea sea más natural para los desarrolladores. Así, a la hora de acceder a datos, en lugar de utilizar otro lenguaje (generalmente SQL), un ORM permite que puedas utilizar los paradigmas habituales de la programación orientada a objetos: clases y objetos. En lugar de pensar en tablas y relaciones, piensas en objetos y propiedades. (El homologo de Hibernate en Java).
- Para realizar las consultas se usa LINQ, con lo que se puede hacer todas las operaciones presentes en los motores de base de datos como join, select, from, where, etc.

Entity Framework CORE

- Ejemplo de consulta:

```
context.Film.Where(film => film.Released > new DateTime(2020, 1, 1)).OrderByDescending(film => film.Released);
```

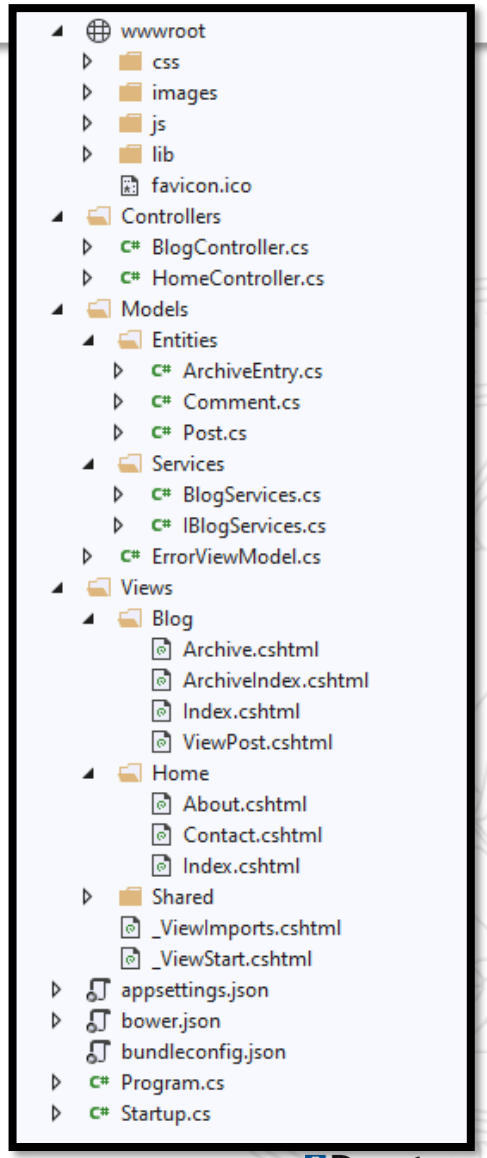
GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa
Unión Europea

Estructura de directorios en ASP.Net CORE

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Estructura de directorios en ASP.Net CORE

- wwwroot: Carpeta con los componentes estáticos de la aplicación.
- Controllers: Contiene los Controladores de la aplicación, estos controladores se deben seguir el siguiente patrón: [nombreControlador]Controller
- Models: Donde se encuentra lo correspondiente al modelo de nuestra aplicación.
- Entities: Donde se encuentran las entidades de la aplicación
- Services: Donde se encuentran los componentes de la lógica de negocio de nuestro modelo.
- ViewModels: Contiene las clases creadas específicamente para intercambiar información con las vistas



GOBIERNO DE ESPAÑA MINISTERIO DE CIENCIA E INNOVACIÓN
UNIVERSIDAD DE MURCIA
Unión Europea
FONDO EUROPEO DE DESARROLLO REGIONAL (FEDER)
Una manera de hacer Europa

¡Gracias!

HĒRCULES

gnoss
Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto