Soluciones .Net, arquitectura de referencia

Dirección General de Sistemas de Información y Comunicaciones

Áreas de Gobierno Tecnológico de SSII y del Dato

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Contenido

Resumen

- **Versión:** v01r00
- **Fecha publicación:** 1 jun 2017
- **Entrada en vigor desde:** 1 jun 2017

Cumplimiento normativo

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

Contacto Arquitectura: l-arquitectura.stic@juntadeandalucia.es

Histórico de cambios

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabezeras de la tablas dónde se indican las fechas de entrada en vigor y versión.

Versión	v01r00	**Fecha publicación**	1 jun 2017	**Fecha entrada en vigor**	1 jun 2017
Alcance					
• Version inicial, Arquitectura de referencia a seguir para proyectos .Net.					

.NET Framework es un entorno de ejecución administrado que proporciona diversos servicios a las aplicaciones en ejecución. Consta de dos componentes principales: Common Language Runtime (CLR), que es el motor de ejecución que controla las aplicaciones en ejecución, y la biblioteca de clases de .NET Framework, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones. Los servicios que ofrece .NET Framework a las aplicaciones en ejecución son los siguientes:

- Administración de la memoria. En muchos lenguajes de programación, los programadores son responsables de asignar y liberar memoria y de administrar la vida útil de los objetos. En las aplicaciones de .NET Framework, CLR proporciona estos servicios en nombre de la aplicación.
- Sistema de tipos comunes. En los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que complica la interoperabilidad entre lenguajes. En .NET Framework, los tipos básicos los define el sistema de tipos de .NET Framework y son comunes a todos los lenguajes que tienen como destino .NET Framework.

- Biblioteca de clases extensa. En lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores pueden usar una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET Framework.
- Marcos y tecnologías de desarrollo. .NET Framework incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos y Windows Communication Foundation para las aplicaciones orientadas a servicios.
- Interoperabilidad de lenguajes. Los compiladores de lenguajes cuya plataforma de destino es .NET Framework emiten un código intermedio denominado Lenguaje intermedio común (CIL), que, a su vez, se compila en tiempo de ejecución a través de Common Language Runtime. Con esta característica, las rutinas escritas en un lenguaje están accesibles a otros lenguajes, y los programadores pueden centrarse en crear aplicaciones en su lenguaje o lenguajes preferidos.
- Compatibilidad de versiones. Con raras excepciones, las aplicaciones que se desarrollan con una versión determinada de .NET Framework se pueden ejecutar sin modificaciones en una versión posterior.
- Ejecución en paralelo. .NET Framework ayuda a resolver conflictos entre versiones y permite que varias versiones de Common Language Runtime coexistan en el mismo equipo. Esto significa que también pueden coexistir varias versiones de las aplicaciones, y que una aplicación se puede ejecutar en la versión de .NET Framework con la que se compiló.
- Compatibilidad con múltiples versiones (multi-targeting). Al usar la Biblioteca de clases portable de .NET Framework, los desarrolladores pueden crear ensamblados que funcionen en varias plataformas, como Windows 7, Windows 8, Windows 8.1, Windows 10, Windows Phone y Xbox 360.

NET Core es una versión modular de .NET Framework diseñada para que sea portátil entre plataformas, a fin de permitir la reutilización del código al máximo y su uso compartido.
.NET Core es portátil entre plataformas porque, aunque se trata de un subconjunto de la versión completa de .NET Framework, proporciona una funcionalidad clave para implementar las características de la aplicación que necesita y reutilizar este código independientemente del destino de la plataforma. Antes, las distintas versiones de .NET para diferentes plataformas carecían de funcionalidad compartida para las tareas clave, como por ejemplo la lectura de archivos locales.
Las plataformas de Microsoft que podrá establecer como destino con .NET Core incluyen Windows, Linux, MacOs, así como dispositivos y teléfonos usando herramientas como Xamarin,permitiendo el uso en dispositivos Windows, IOS y Android.

1. Plataforma de ejecución del servidor

Arquitectura de ejecución	x86-64bits (AMD64/EM64T)
Sistema Operativo	Windows Server XXXX
Plataforma Objetivo	.NET Framework 4.6.1 .NET Standard 1.4 .NET Core 1.0

2. Entorno Tecnológico

Lenguaje	C# v6
Framework Desarrollo Web	ASP.NET MVC 5 / ASP.NET Core
ORM	Entity Framework 6 / Entity Framework Core
Ioc/DI	DryIoc / LightInject / Simple Injector
Identidad	ASP.NET Identity

3. Diseño y Modularidad

Tener una estructura común y estandarizada permite a desarrolladores trabajar de forma similar en cualquier proyecto, simplificando con ello el entendimiento del mismo.

Modulo de presentación

Se recomienda el uso de ASP.NET Core, aunque se permita el uso de ASP.NET MVC 5.

La estructura de un proyecto .NET Web para el modulo de Presentacion debe adaptarse lo maximo posible ha esta organización:

ASP.NET MVC

- Directorio **App_Data**.

  Este directorio está pensado para ubicar archivos de datos, normalmente bases de datos MSSQL. También es el lugar adecuado para archivos XML o cualquier otra fuente de datos.
- Directorio **App_Start**.

  Este directorio contiene los archivos de código que se ejecutan al inicializar la aplicación..
- Directorio **Content**.

  El directorio Content está pensado para el contenido estático de la aplicación, especialmente útil para archivos css e imágenes asociadas.
- Directorio **Scripts**.

  El directorio scripts está pensado para ubicar los archivos de javascript (*.js). El código javascript es ejecutado en el contexto del navegador, es decir, en la parte cliente, y nos permite ejecutar acciones sin necesidad de enviar los datos al servidor.

ASP.NET Core

- Directorio **wwwroot**

  En este directorio se ubica todo el contenido estatico del projecto
- Directorio **Dependencies**

  Es el directorio usado para gestionar las dependencias de Bower y NPM
- Fichero **Program.cs**

  Es el fichero mas importante del proyecto. Esta clase es usada para inicializar, construir, y ejecutar el servidor(IIS y Kestrel) y alojar la aplicacion. Aqui tambien se define la raiz del proyecto.
- Fichero **Startup.cs**

  Esta clase es la que permite configurar los Middlewares necesarios para las peticiones a la aplicacion.

Comun

- Directorio **Controllers**.

  El directorio controllers en el lugar para los controladores, que son las clases encargadas de recibir y gestionar las peticiones http de la aplicación.
- Directorio **Models**.

  El directorio models es la ubicación que nos propone ASP.NET para las clases que representan el modelo de la aplicación, los datos que gestiona nuestra aplicación.
- Directorio **Views**.

  El directorio Views contiene los archivos de vista. Los controladores devuelven vistas sobre las que inyectamos el modelo de nuestra aplicación. Estas vistas son interpretadas por el motor de renderización. Son archivos similares a aplicaciones de ASP clasico, donde tenemos código HTML estático y determinadas zonas de código que son ejecutadas en el servidor.

Otros Módulos

Para el resto de módulos se debe de seguir una estructura guiada por el patrón BCE segun se describe en la documentación de Arquitectura de Referencia Común.

Se recomienda el uso de .NET Standard para mayor compatibilidad de entornos.