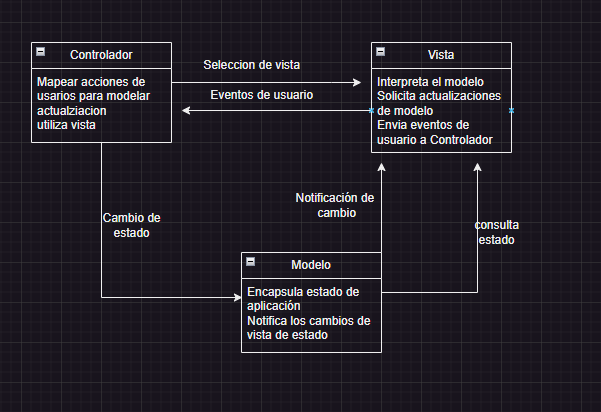
PATRON MVC



Modelo: Son clases que solo tienen propiedades (id\_foto, nom\_foto, etc).

Maneja los datos del sistema y las operaciones con esos datos y el estado de la aplicación

EJ: datos de la aplicación, bd, clases

Vistas: define y gestiona como se presentan los datos al usuario (Razor view engine - aspx)

EJ: modelos, plantillas, html, css, bs5, etc

Controlador: dirige la interacción del usuario (teclas oprimidas). Son clases que definen a los CRUD que solo tienen métodos y constructores. No tienen propiedades (alta baja modificación búsqueda filtros)

EJ: código back-end, interacción

Las vistas están asociadas a un controlador, el controlador genera una subcarpeta dentro de una vista. Las vistas son devueltas a través de un método de acción asociado a un controlador

View model: contiene los datos para mostrar en el view.

Asp.net core (mvc)

.net core

* Multiplataforma
* Crear micro servicios
* Docker containers
* Sistema escalable
* Empezar nuevo proyecto

.net framework

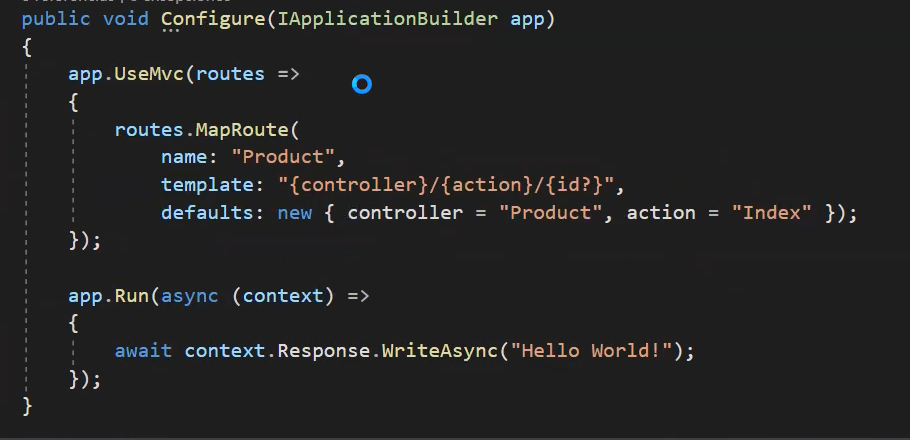
* Paquetes nuGet, librerías .net
* Usar tecnologías no soportadas por core
* Usar una plataforma que no soporta core
* Usar una plataforma que no soporta core
* Modificar un proyecto

Clase start up

* Inicia la aplicación
* Constructor
* Contiene los métodos ConfigureServices y configure donde se configuran los servicios y los middlewares

ConfigureServices -> inyecta en -> cofigure

**Startup.cs**



Class ConfigureServices(IserviceCollection service){

Service.addRazorPages();

}

Class Configure {get}

Middleware es una clase que toma la acción de un usuario y realiza una acción.

¿está dentro del configure?

Middleware1 (){

App.use();

Await nextInvoke();

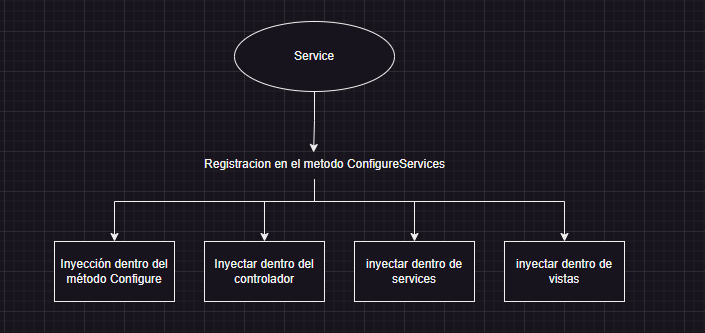
}

Middleware1 (){

App.run();

}

wwwroot es una carpeta que contiene los archivos estáticos (fotos, etc)



Las **clases** definidas como servicios **registran** y luego **inyectan**, se definen dentro del configureServices y luego se inyectan

ViewData o ViewBag son utilizados para pasar información del controlador a la

**ViewData**

* ViewData es un objeto de diccionario fuertemente tipado en el que se introducen los datos a través de la conocida sintaxis "key/value".
* ViewData deriva de la clase ViewDataDictionary.
* Uso y asignación al objetos: ViewData["Dato"] = 1
* Lectura del objeto en la vista: @ViewData["Dato"]

**ViewBag**

* El objeto ViewBag es un objeto de tipo dinámico que le permite crear propiedades *al vuelo*.
* Uso y asignación al objetos ViewBag.Dato = 1
* Lectura del objeto en la vista: @ViewBag.Dato
* El valor se recupera desde la vista

Filtros (en consola – debug.writeline)

OnActionExecuting -> cuando se inicia

OnActionExecuted -> cuando finaliza la ejecución

OnResultExecuting -> Index before result

OnResultExecuted -> Index result is…

---

Un método dentro de una subcarpeta dentro del controlador devuelve una vista

--

Vistas parciales (reusabilidad de HTML)

Cuando inyectar un servicio dentro de una vista

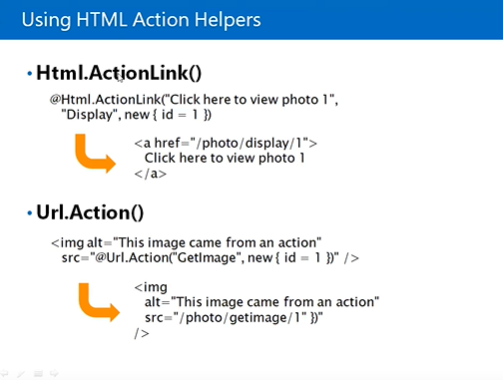
* Servicios que transforman o dan formato a un texto
* Servicios que tienen un impacto visual
* Servicios que requieren múltiples iteraciones dentro de una vista

Cuando inyectar un servicio dentro de una vista

* Servicios que recuperan información (datos)
* Servicios que dependen de una fuente externa (apis, sitios web)
* Servicios que devuelven resultados consistentes para una vista

Introducción a HTML helpers y Tag helpers

* Html helpers
  + Usa sintaxis razor
  + Mas fácil de idenficar áreas de código
  + No requiere habilitar ninguna característica especial en la vista para usarlos
* Tag helpers
  + Usa una sintaxis parecida a la de HTML
  + Requiere uso explicito de la directiva
  + Crea html más legible



@Html.ActionLink(“Click here to view foto 1”, “Display”, new {id = 1})

Nombre del link, método(dentro del controlador), controlador