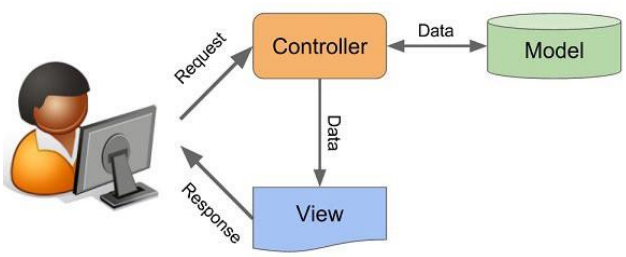
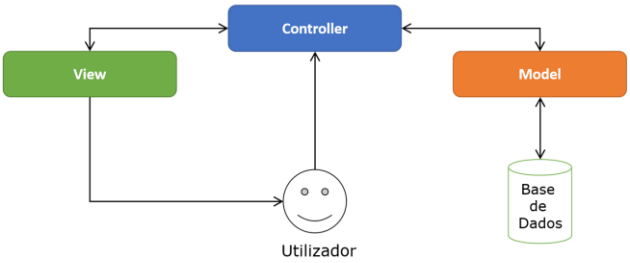
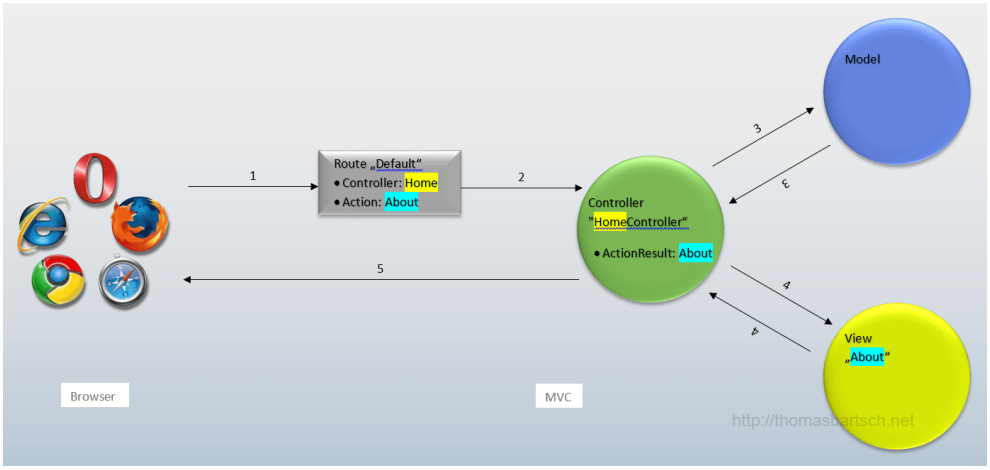
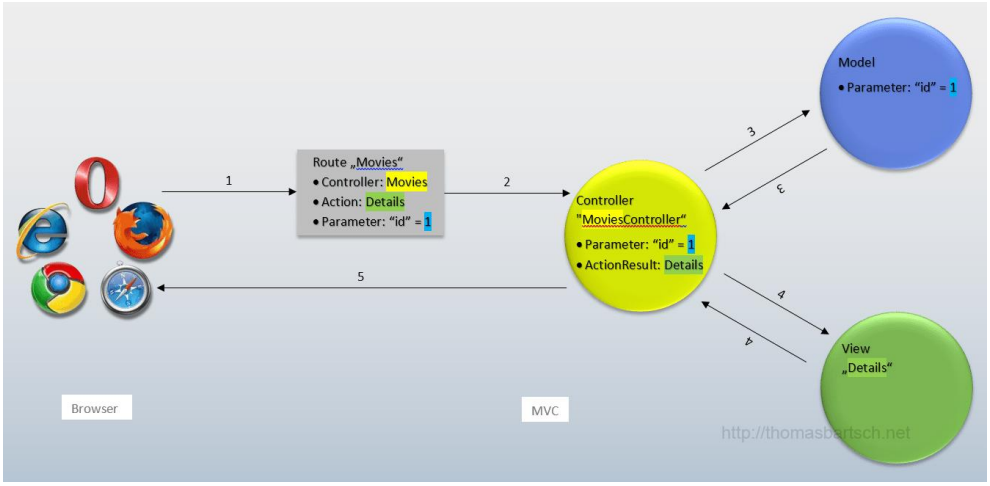
Objetivo:

- Criar um projeto Web ASP.NET Core MVC
- Configurar a aplicação
- Adicionar Controller
- Adicionar View
- Adicionar Model









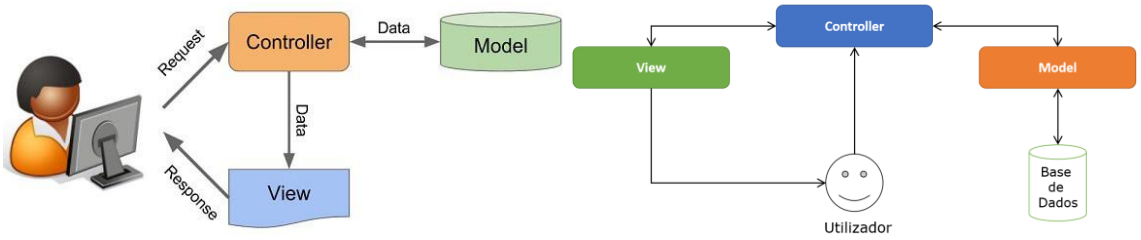
Criar aplicação ASP.NET Core

1. Criar o projeto ASP.NET Core baseado no Template MVC (01-ASP.NET Core-Conceitos fundamentais) ou Empty (02-ASP.NET Core-Projeto de raiz)

2. Adicionar um Controller (03-ASP.NET Core-Controllers)

3. Adicionar uma View (04-ASP.NET Core-Views)

Models



Model (Modelo)

- Consiste nos dados da aplicação, regras de negócios, lógica e funções.
- Contém ou representa os dados da aplicação e expor as operações de manipulação de dados.

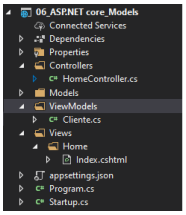
Tipos de Models:
- View Models – Representam dados que são passados diretamente para Views.
- Domain Models – Contêm dados e toda a lógica de operações para transformar, criar regras, guardar e manipular dados. Por exemplo, a lógica de comunicação coma bases de dados.

- A lógica dos dados deve ficar apenas dentro do Model.
- Os models não devem expor detalhes sobre como os dados são obtidos, não conter lógica relacionada com a interação (isso fica para o Controller) e não conter lógicas de apresentação (isso fica para as Views).

Usar Models

Passar informações de uma forma estruturada, com mais potencialidades do ViewBag e ViewData
Podemos usar a Pasta Models, ou então, criar a ViewModels, para agrupar as classes que são usadas exclusivamente para dados para Views, sem incluir acesso a base de dados.

Criar a pasta "ViewModels"
Adicionar a classe Cliente



Cliente.cs

```
    public class Cliente
    {
         public string Nome { get; set; }
         public string Apelido { get; set; }
         public string Telefone { get; set; }
    }
```

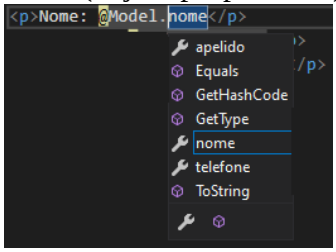
HomeController.cs

```
using _06_ASP.NET core_Models.ViewModels;
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            Cliente cliente = new Cliente()
            {
                Nome = "Martim",
                Apelido = "Moreira",
                Telefone = "123456789"
            };
            // ou ...
            Cliente cliente_1 = new Cliente();
            cliente_1.Nome = "José Pedro";
            cliente_1.Apelido = "Moreira";
            cliente_1.Telefone = "123456789";

            return View(cliente);
        }
    }
```

Home\index.cshtml

```
@model _06_ASP.NET_core_Models.ViewModels.Clinte;
@{
    ViewData["Title"] = "Index";
}

<h1>Cliente</h1>
<p>Nome: @Model.Nome</p>
<p>Apelido: @Model.Apelido</p>
<p>Telefone: @Model.Telefone</p>
```

Para poder ter acesso ao `Intellisense` (objeto.propriedade), por exemplo, `@Model.nome`



Adicionar a diretiva:
`@model _06_ASP.NET_core_Models.ViewModels.Clinte;`

Enviar mais do que um model (ViewModels) para uma View

Uma vez que só podemos usar uma vez `@model` (ou seja apenas uma classe), para usar, por exemplo duas classes Models, podemos criar ViewModel (class) para agregar as duas classes

Cliente.cs

```
public class Cliente
{
     public string Nome { get; set; }
     public string Apelido { get; set; }
     public string Telefone { get; set; }
}
```

Automovel.cs

```
public class Automovel
{
    public string Marca { get; set; }
    public string Modelo { get; set; }
}
```

Classe de agregação das duas classes

ClienteAutomovel.cs

```
public class ClienteAutomovel
{
    public Cliente Cliente { get; set; }
    public Automovel Automovel { get; set; }
}
```

Home\index.cshtml

```
@model _02_ASP.NET_core_ProjetoRaiz.ViewModels.ClienteAutomovel

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

<h1>Cliente e Automovel</h1>
<p>Nome: @Model.Cliente.Nome @Model.Cliente.Apelido</p>
<p>Carro: @Model.Automovel.Marca @Model.Automovel.Modelo</p>
```