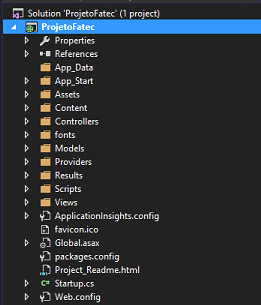
# Estrutura (Pastas que serão utilizadas)



Para a realização desse projeto, utilizamos o conceito de SPA (Single Page Application) oferecido pelo Asp.Net MVC. Não se preocupem caso não conheçam essas coisas, utilizamos assim para facilitar a escrita de código. Isso porque utilizando esse conceito de SPA, tenho uma página genérica que contém o *HEADER, MENU E FOOTER* do meu site (coisas que existem em todas as telas) assim, eu não preciso ficar reescrevendo isso em todos os arquivos, basta escrever o conteúdo que é único de cada página

## App\_Start

* BundleConfig.cs

Arquivo contendo os “imports” dos arquivos .js e .css. Nele é possível criar um “bundle” contendo mais de um arquivo do mesmo tipo, para assim, facilitar na importação dos mesmos.

Obs: Esse arquivo só será utilizado caso tenha a necessidade de criar outro arquivo .css ou .js.

Obs2: Ao adicionar o arquivo aqui é necessário referenciar o Bundle no local desejado.

## Assets

Contém os arquivos do template, isto é, arquivos .css e .js, além de imagens.

Obs: Dificilmente será necessário (e útil) utilizar essa pasta.

## Content

Contém os arquivos .css utilizados no site.

Obs: Ao adicionar um arquivo aqui deve se adicionar no BundleConfig.cs. (Já existe um bundle na linha 69 contendo a “importação” dos arquivos css dessa pasta, se adicionado nesse mesmo bundle, ele já estará vinculado no site).

## Controllers

Pasta contendo as controllers das Views. As controllers servem para carregar a página conforme o link passado. Ou seja, se eu tenho uma tag *<a href=”/Despesas”>Despesas</a>*, quando o usuário clicar nesse link, será chamado o método *Index()* da controller, que retorna a tela correspondente.

Ele sabe qual a tela correspondente conforme a estrutura de pastas da pasta Views. Então, para uma controller funcionar (por exemplo, a controller *Despesa*) é necessário que exista uma pasta, dentro de Views, com o nome *Despesa*, assim, quando ele chama a função *Index()*, ela retorna o conteúdo da *Index.cshtml* dentro da pasta *Despesas*.

Obs: Essa pasta e esses arquivos só serão utilizados caso exista a necessidade de se criar um link inexistente no site até agora. Porém, para cumprir com os requisitos mínimos, não existe essa necessidade.

## Scripts

## Contém os arquivos .js utilizados no site. O arquivo site.js contém algumas coisas em comum no site todo. Dentro da pasta *Scripts/views* contém os arquivos .js referentes a cada tela da pasta *Views*. Apenas esses arquivos podem (e devem) serem alterados.

Obs: Ao adicionar um arquivo aqui deve se adicionar no BundleConfig.cs. Porém, para cumprir com os requisitos mínimos do sistema, não existe a necessidade de ser criado um novo arquivo.

## Viws

Contém os arquivos com html do site. Diferente do que estamos acostumados, os arquivos aqui possuem a extensão .cshtml e não .html, porém, para a realização do projeto, não precisamos nos preocupar com isso, os arquivos apenas possuem essa extensão, para que a estrutura MVC funcione.

Pastas:

* Shared:

Dentro dessa pasta temos os arquivos que são compartilhados entre as telas.

* + Layout.cshtml  
    Arquivo principal, nele possuímos os *htmls* comuns entre as páginas, além dos “imports” de arquivos .css e .js. Aqui possuímos algumas coisas novas, que são:
    - Html.RenderPartial: Utilizado para carregar algum outro arquivo html que é apenas uma parte do html. Utilizamos ele para carregar o HEADER, MENU e FOOTER que estão separados em outras pastas.
    - RenderBody: Quando chamamos uma controller e ela retorna uma “View” o que ela está fazendo é carregar o conteúdo de um arquivo html dentro desse RenderBody.
    - Scripts.Render: Utilizado para importar nossos arquivos ,js e .css adicionados no arquivo BundleConfig.cs (mencionado no tópico 1.1)
  + Includes  
    Contém os arquivos que devem compor um outro, no caso, contém o html de HEADER, MENU e FOOTER.
* Login:

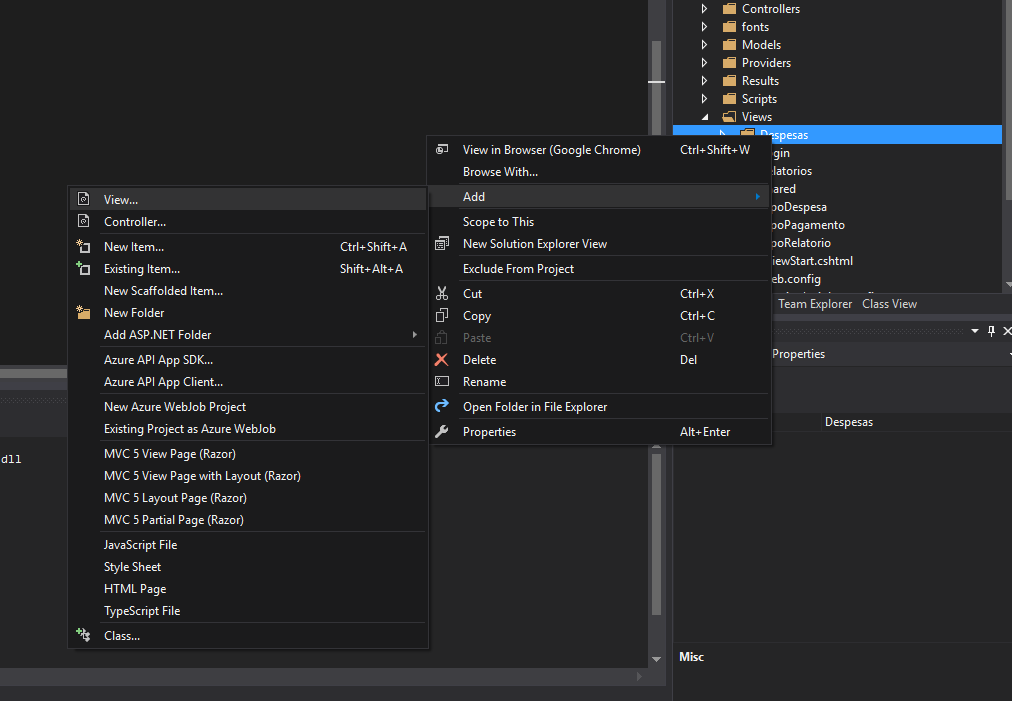
Contém o html da tela de Login, essa tela não possui partes em comum com as outras telas, por isso, ela é um arquivo único com uma estrutura completa do HTML

* Demais Pastas:

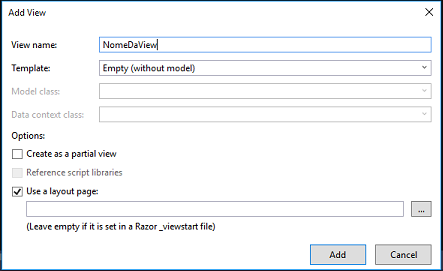
Nas demais pastas contém o HTML de cada link do menu, dentro dessa pasta, existe um arquivo *Index.cshtml* (que é carregado pelas controllers) esse arquivo carrega o HTML desejado e o arquivo.js referente aquela página.

# Adicionando uma view

Nossos arquivos HTML nessa estrutura são chamados de View, para adicionar uma nova View basta clicar com o botão direito do mouse na pasta desejada (dentro da pasta Views) ir em *Add* e clicar em *View...*

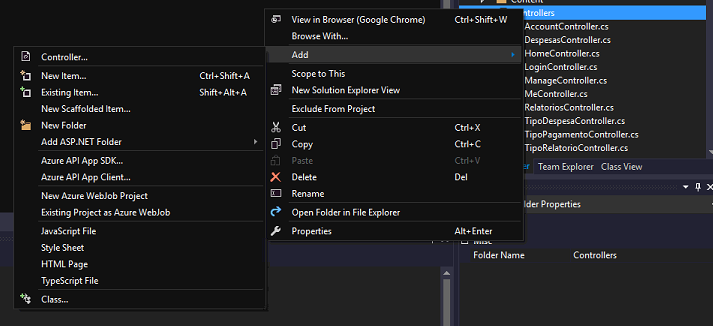


## Depois, bastar dar um nome para a View e clicar em *Add.*

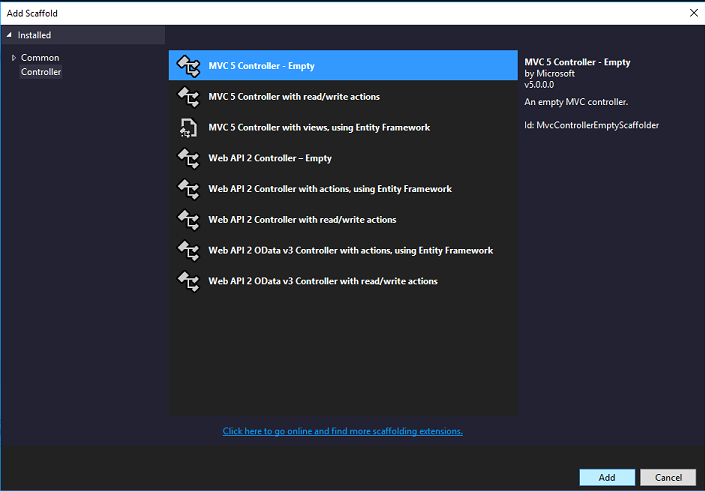


# Criando uma controller

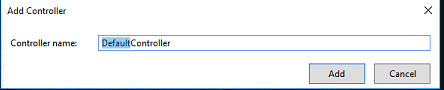
Na pasta *Controllers* clique com o botão direito nela, vá em *Add* e clique em *Controller...*



Escolha a opção *MVC 5 Controller – Empty*



Depois disso basta dar um nome ao a nova Controller



**IMPORTANTE:** O nome da controller deve terminar com a palavra “Controller”.

# VIEWBAG.TITLE

Vocês vão reparar que tanto na *Layout.cshtml*, quanto nas *Index.cshtml* possuem a variável ViewBag.Title.

ViewBag é um conceito do ASP.NET MVC que não entraremos em detalhes no projeto, mas sua utilização aqui serve para mudar o <title> do html dependendo da tela em que estamos.

Além disso, utilizamos essa variável para aplicarmos o efeito de item ativo, no menu.

# Links auxiliares

Template: <http://keenthemes.com/metronic/preview/>

Documentação do Template: <http://keenthemes.com/metronic/documentation.html>

Bootstrap: <https://v4-alpha.getbootstrap.com/>

FontAwesome: <https://fontawesome.com/icons?d=gallery> (foi atualizado recentemente, pode ser que alguns ícones não funcionem, ou não tenham a mesma cara)

LineAwesome: <https://icons8.com/line-awesome>

FlatIcon: <https://www.flaticon.com/>

## 