Pesquisa: Relação **MTV** no **Jango**

Data:17/09/2025
Backend

O'que é **Django**

Na atualidade, o crescente número de pessoas conectadas à internet tornou o desenvolvimento de soluções escaláveis e seguras uma prioridade em diversos setores da indústria de tecnologia. Isso acontece à medida que as pessoas passam a depender cada vez mais de serviços da web, como plataformas de redes sociais, streaming, e-commerce, jogos online, educação, IoT e uma variedade de outros serviços para atender às suas necessidades diárias e momentos de lazer.

Essa tendência tem se tornado o foco no mundo do desenvolvimento de software. Nas maiores empresas de tecnologia, é muito comum precisar de aplicações que atendam a essas demandas, uma vez que a experiência do usuário e a disponibilidade constante são fatores críticos para o sucesso e a satisfação dos clientes.

Para facilitar o processo de criação de soluções web, começaram a surgir frameworks que abstraem a maioria dos processos árduos e repetitivos de configuração do ambiente de desenvolvimento, como gerenciamento de Banco de Dados, configuração de roteamento de URLs, configurações de autenticação e autorização, dentre outras.

Dessa forma, as principais linguagens do mercado passaram a contar com seus respectivos frameworks e bibliotecas para o desenvolvimento web. Java, por exemplo, recebeu o Spring, enquanto o JavaScript adotou o Node.js, entre outros.

Python também trilhou um caminho semelhante e ganhou grande popularidade ao receber um poderoso framework voltado para o desenvolvimento web: o Django.

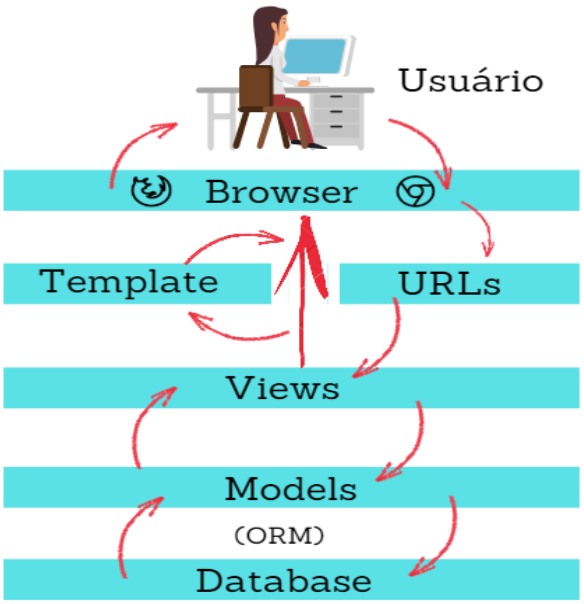
É essencial explorar e compreender essas ferramentas para se destacar no cenário do desenvolvimento de software contemporâneo, assim como entender suas principais características, em que podemos utilizá-las e como podemos trabalhar com elas. Então, vamos conhecer mais sobre Django e como esse framework nos ajuda no dia a dia?

Arquitetura do Django

A arquitetura do Django é relativamente simples. Basicamente, um projeto Django possui como padrão de projeto o MTV (Model, Template, View), que servem para:

- **Model: Mapeamento do banco de dados para o projeto;**
- **Template: Páginas para visualização de dados. Normalmente, é aqui que fica o HTML que será renderizado nos navegadores;**
- **View: Lógica de negócio. É aqui que determinamos o que irá acontecer em nosso projeto.**

Toda esta arquitetura é interligada e conversam entre si. Uma depende da outra para realizar um determinado serviço e, no final, executar a tarefa que o usuário solicitou. A imagem abaixo descreve exatamente como este processo funciona:



Como visto na imagem, quando o usuário faz uma requisição pelo browser, utilizando uma rota, é executado um método das *Views,* que utiliza os *Models* para acessar o banco de dados e retornar as informações. Estas informações são renderizadas pela camada de *Template* e, finalmente, é renderizado para o usuário pelo navegador.

Toda esta arquitetura se assemelha bastante com o padrão MVC (Model, View, Controller), utilizado em diversos outros frameworks, como o Laravel, Zend Framework e muitos outros. Comparado ao MVC, as camadas do padrão MTV podem ser consideradas como podemos ver abaixo:

- **M (MTV) = M (MVC): Estas duas camadas possuem a mesma responsabilidade, mapeamento do banco de dados para o projeto;**
- **T (MTV) = V (MVC): Estas camadas possuem a mesma responsabilidade, exibir informações para o usuário da aplicação, normalmente utilizando páginas HTML;**
- **V (MTV) = C (MVC): Estas duas camadas, apesar de possuírem responsabilidades parecidas, conceitualmente, apresentam algumas diferenças.**

No próprio FAQ do Django, há uma área onde os desenvolvedores explicam as diferenças conceituais e o motivo de nomear a camada de visualização de dados como `templates` e de `views` a camada de lógica de acesso.

Resumidamente, a equipe de desenvolvimento do framework entende que a camada view descreve quais dados serão apresentados ao usuário, não a forma (aparência) que eles serão exibidos. Portanto, no padrão MTV, uma view é uma função que retorna algo para uma solicitação, porque ela define apenas quais dados serão apresentados, não como serão mostrados.

Além disso, é sensato separar o conteúdo da apresentação (por questões de organização e padronização do código). É aí que entra a camada template. Com os dados retornados pela view, a `template` fica responsável por definir a forma que esses dados serão apresentados, normalmente em páginas HTML.

Porém, onde o `controller` se encaixa? No Django, os desenvolvedores do framework entendem que esta camada é a própria estrutura do projeto. É este mecanismo que envia uma solicitação para a `view` adequada de acordo com a configuração de rotas do Django. E isso faz sentido! Se pararmos para pensar, o controlador da aplicação é o próprio conjunto de bibliotecas que compõe o projeto. A `view` é mais adequada a retornar os dados a serem visualizados, que serão exibidos ao usuário por meio de `templates`.

https://www.alura.com.br/artigos/django-framework?