**Pesquisa**

* Pesquisar como é o MVC no Django
* Indicar o que cada item no MTV no Django tem de relação com cada item do MVC

O Django segue uma arquitetura Model-View-Template (MTV), que é uma variação do padrão MVC (Model-View-Controller) onde o "Template" (Gabarito) substitui o papel do Controller. O Model lida com a estrutura de dados e a interação com o banco de dados, a View (Visão) lida com a lógica de requisições e a apresentação das respostas, e o Template (Gabarito) é responsável pela apresentação visual dos dados no navegador.

Django usa o padrão de design **MTV (Model-Template-View)**, que é uma variação do padrão **MVC (Model-View-Controller)**. A principal diferença está na nomenclatura e na forma como as responsabilidades são divididas, mas a ideia central é a mesma: separar a lógica de negócio, a apresentação de dados e a forma como a aplicação interage com o usuário.

### **Mapeamento do MTV para o MVC**

Para entender a relação, vamos item por item:

* **Model (no Django) ↔ Model (no MVC):** Este é o ponto mais direto. Em ambos os padrões, o **Model** representa a estrutura de dados e a lógica de negócio da aplicação. No Django, o Model define a estrutura das tabelas do banco de dados e as operações de manipulação desses dados. Ele é a "única fonte de verdade" para seus dados. No MVC, a função é idêntica: gerenciar os dados e a lógica de negócio, sem se preocupar com a interface do usuário.
* **Template (no Django) ↔ View (no MVC):** Esta é a principal diferença de nomenclatura. No Django, o **Template** é responsável pela **apresentação dos dados** ao usuário. Ele é o arquivo HTML que contém a marcação e a lógica de apresentação (usando a linguagem de template do Django) para exibir a informação. Ele não lida com a lógica de negócio. Em contraste, a **View** no MVC tem a mesma responsabilidade: ser a interface do usuário, exibindo os dados fornecidos pelo Model.
* **View (no Django) ↔ Controller (no MVC):** Esta é a outra grande diferença de nome. A **View** no Django atua como um **controlador**. Ela recebe a requisição HTTP do usuário (por exemplo, um clique em um link), interage com o **Model** para obter ou modificar os dados, e então seleciona o **Template** apropriado para renderizar a resposta. Em suma, a **View** do Django orquestra a interação entre o **Model** e o **Template**, o que é a função exata do **Controller** no MVC.

### **Referências**

* **Documentação oficial do Django**
* **Artigos e tutoriais online**