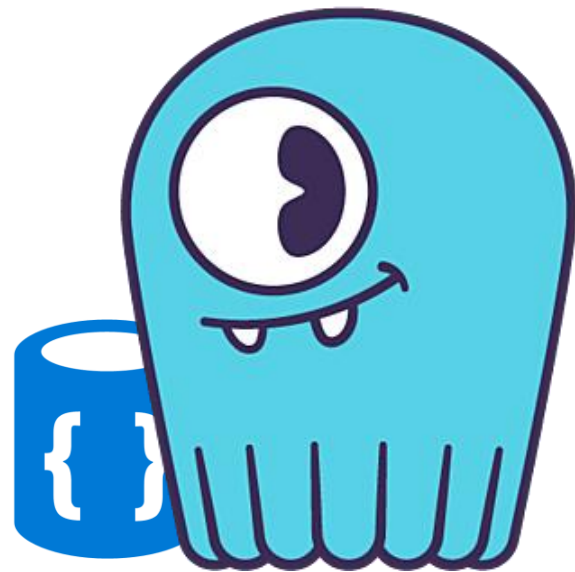


# Primeiros Passos com o *Framework* Django

**Prof. Dr. Diego Bruno**

Education Tech Lead na DIO

Doutor em Robótica e *Machine Learning* pelo ICMC-USP



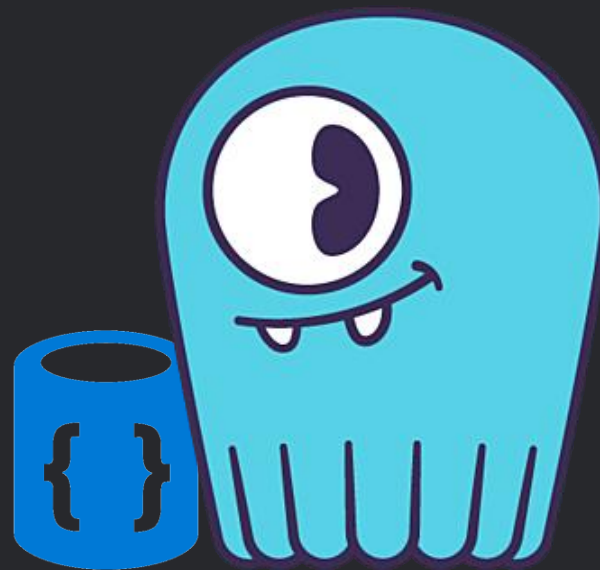
# Roteiro para nossa aula de hoje

- Introdução
- Motivação
- Conclusões



# Django

Prof. Dr. Diego Bruno



# Mas o que é Django?

O que é Django? Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção.



# Mas o que é Django?

***Django*** é um ***framework*** de web server-side extremamente popular e repleto de características, escrito em Python. O módulo mostra por que o Django é um dos frameworks web mais populares, como configurar um ambiente de desenvolvimento e como começar a usa-lo para criar seus próprios aplicativos da Web.



# Mas o que é Django?

Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção. Construído por desenvolvedores experientes, o Django cuida de grande parte do trabalho de desenvolvimento web, para que você possa se concentrar em escrever seu aplicativo sem precisar reinventar a roda.



# Mas o que é Django?

É gratuito e de código aberto, tem uma comunidade próspera e ativa, ótima documentação e muitas opções de suporte gratuito e pago.



# Mas o que é Django?

## Completo

Django segue a filosofia de "baterias incluídas" e fornece quase tudo que desenvolvedores possam querer fazer "fora da caixa". Como tudo o que você precisa é parte de um "produto", tudo funciona perfeitamente junto, seguindo princípios de design consistentes.





# Mas o que é Django?

## Versátil

Django pode ser (e tem sido) utilizado para construir quase todo tipo de website, passando por redes sociais e sites de notícias. Ele pode trabalhar com qualquer framework do lado do cliente, e pode entregar conteúdo em praticamente qualquer formato (incluindo HTML, feeds RSS, JSON, XML, etc).



# Mas o que é Django?

## Versátil

Django pode ser (e tem sido) utilizado para construir quase todo tipo de website, passando por redes sociais e sites de notícias. Ele pode trabalhar com qualquer framework do lado do cliente, e pode entregar conteúdo em praticamente qualquer formato (incluindo HTML, feeds RSS, JSON, XML, etc).



# Mas o que é Django?

## Seguro

Django ajuda os desenvolvedores a evitar os erros de segurança mais comuns, fornecendo um framework que foi desenhado para "fazer as coisas certas", de modo a proteger o website automaticamente.



# Mas o que é Django?

## Escalável

Django usa uma arquitetura baseada em componentes “shared-nothing” (“nada-compartilhado”) (cada parte da arquitetura é independente das outras, e conseqüentemente podem ser substituídas ou mudadas caso necessário).



# Mas o que é Django?

## Sustentável

O código do Django é escrito usando princípios de design e padrões que encorajam a criação de código sustentável (que facilita a manutenção) e reutilizável.



# Mas o que é Django?

## Portável

Django é escrito em Python, que executa em muitas plataformas. Isso significa que você não está preso em nenhuma plataforma de servidor em particular, e pode executar seus aplicativos em muitas distribuições do Linux, Windows e Mac OS X. Além disso, o Django tem um bom suporte em muitos provedores de servidores de web.



# Mas o que é Django?

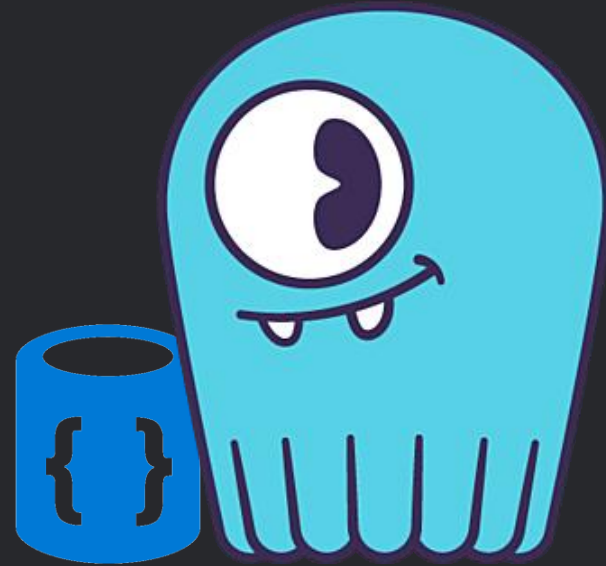
## De onde veio?

Django foi inicialmente desenvolvido entre 2003 e 2005 por um time de web que era responsável por criar e manter sites de jornal. Depois de criar um número de sites, o time começou a fatorar e reutilizar muitos de seus códigos comuns e padrões de design. Esse código comum evoluiu para um framework genérico de desenvolvimento web.



# Obrigado!

Prof. Dr. Diego Bruno



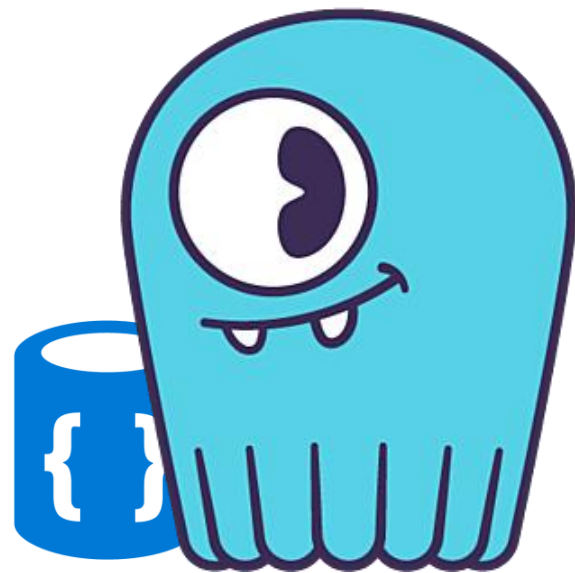


# Trabalhando com o *Framework* Django

**Prof. Dr. Diego Bruno**

Education Tech Lead na DIO

Doutor em Robótica e *Machine Learning* pelo ICMC-USP



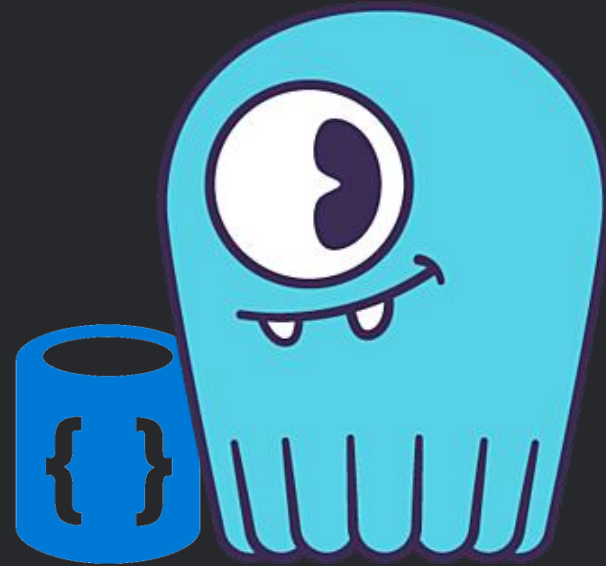
# Roteiro para nossa aula de hoje

- Introdução
- Motivação
- Conhecendo o *Framework*
- Baterias do Django
- CRUD
- Exemplos



# Django

Prof. Dr. Diego Bruno



# Mas o que é Django?

O que é Django? Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção.



# Mas o que é Django?

***Django*** é um ***framework*** de web server-side extremamente popular e repleto de características, escrito em Python. O módulo mostra por que o Django é um dos frameworks web mais populares, como configurar um ambiente de desenvolvimento e como começar a usa-lo para criar seus próprios aplicativos da Web.



# Mas o que é Django?

Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção. Construído por desenvolvedores experientes, o Django cuida de grande parte do trabalho de desenvolvimento web, para que você possa se concentrar em escrever seu aplicativo sem precisar reinventar a roda.



# Mas o que é Django?

É gratuito e de código aberto, tem uma comunidade próspera e ativa, ótima documentação e muitas opções de suporte gratuito e pago.



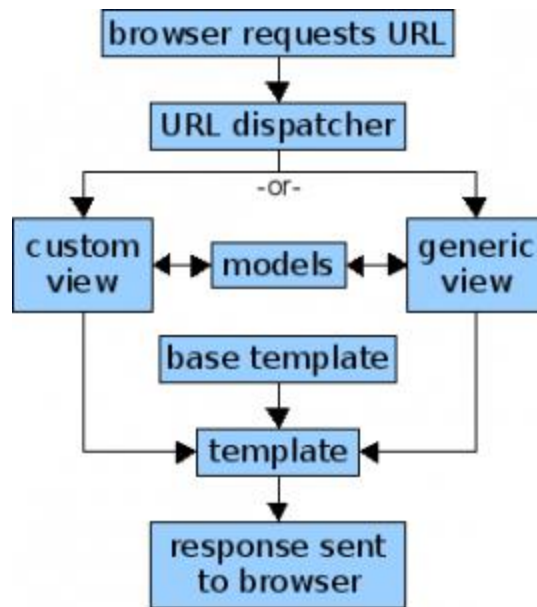


Real Python



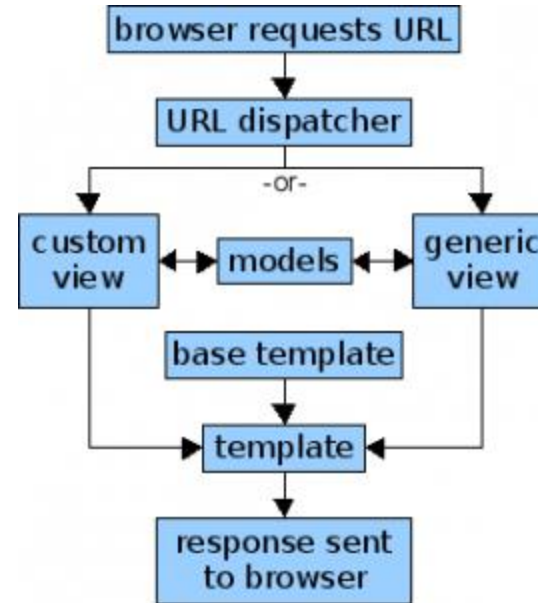
## MTV – Model Template View

Não estamos falando daquele canal de TV... estamos falando de uma variação do MVC (*Model View Controller*).



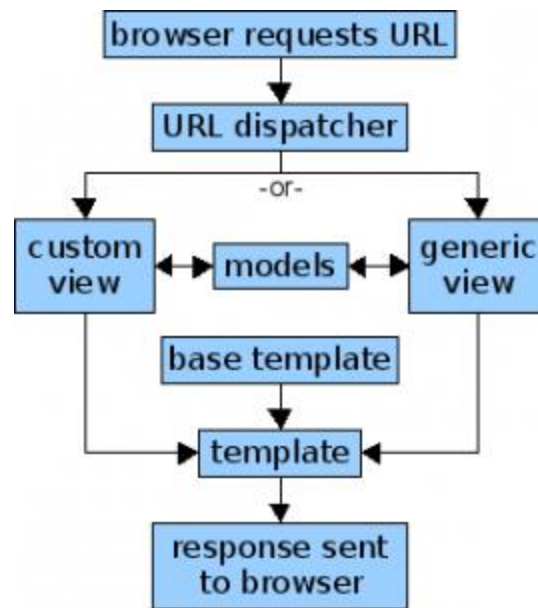
## MTV – Model Template View

Todo o desenvolvedor deve ao menos saber qual o objetivo deste padrão de desenvolvimento. Separa-se as regras de negócios (controlador), os dados e métodos de acessos aos mesmo (modelo) e as regras de apresentação (visualização). Desse modo, caso ocorra alguma alteração na sua camada de visualização (digamos que sua aplicação ganhe uma versão mobile).



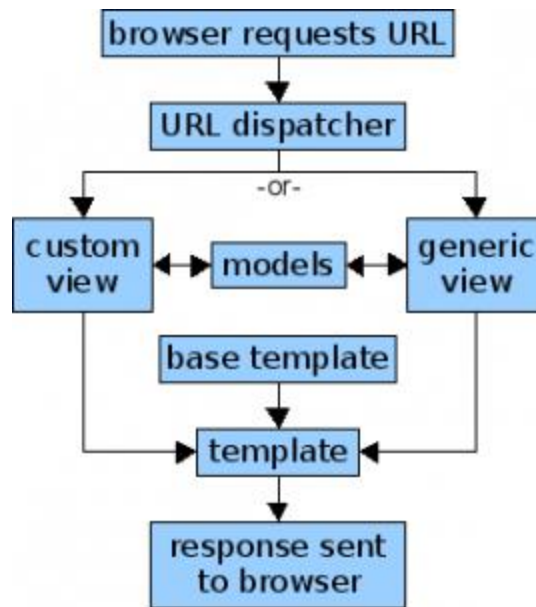
## MTV – Model Template View

No caso do Django, há uma variação deste modelo que é o MTV (Model Template View). Aqui entramos em um assunto que gera bastante discussão entre os iniciantes Django: *aonde raios está o Controller?*



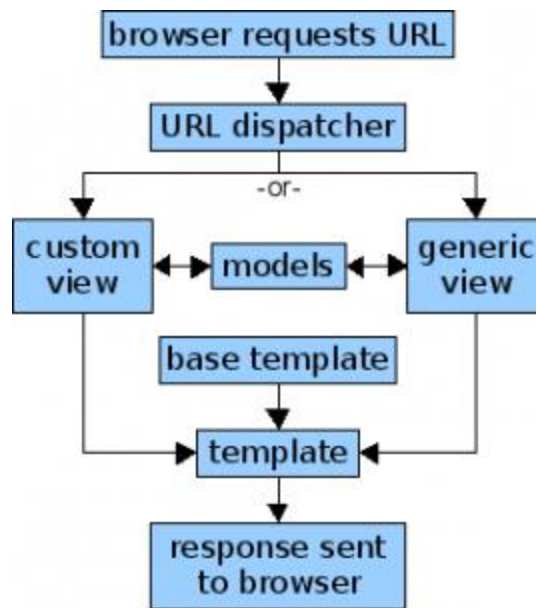
## O Framework é o Controlador

Em Django, o controlador não é responsável pela lógica do negócio e sim pelo funcionamento do seu projeto. Além de **models**, **views** e **templates**, em Django nós temos também **url dispatchers**, **middlewares** e **handlers**. E é este “além” que o Django encara como **Controller**.



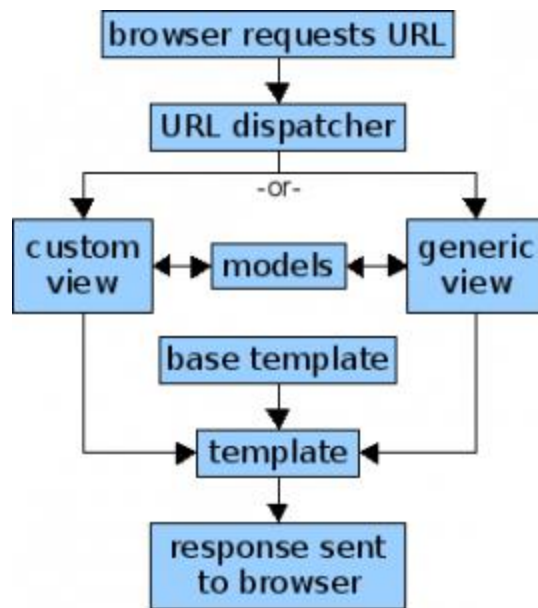
## O Framework é o Controlador

Somos capazes de incrementar o controlador do Django, por exemplo, somos obrigados a criar regras de urls dizendo ao Django que ao receber uma requisição para a url X, ele deverá acionar a view Y.



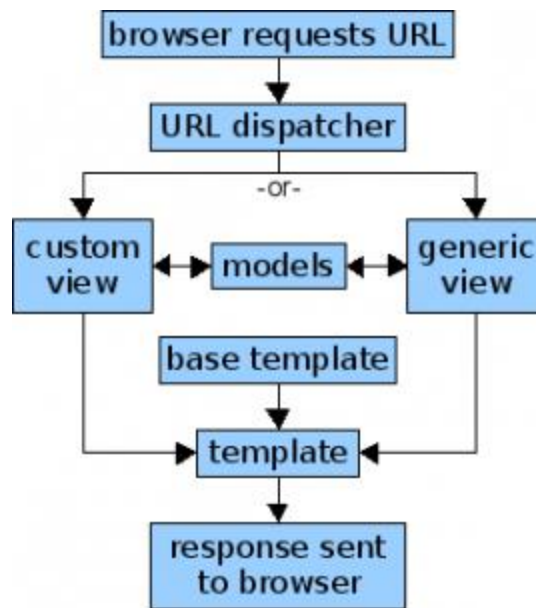
# Models

Em Models, escrevemos classes que designarão nossas tabelas no banco de dados e manipularemos estas através de orientação a objetos (ORM – [Mapeamento Objeto Relacional](#)). Você não precisa escrever absolutamente nada de SQL, a não ser que seja estritamente necessário.



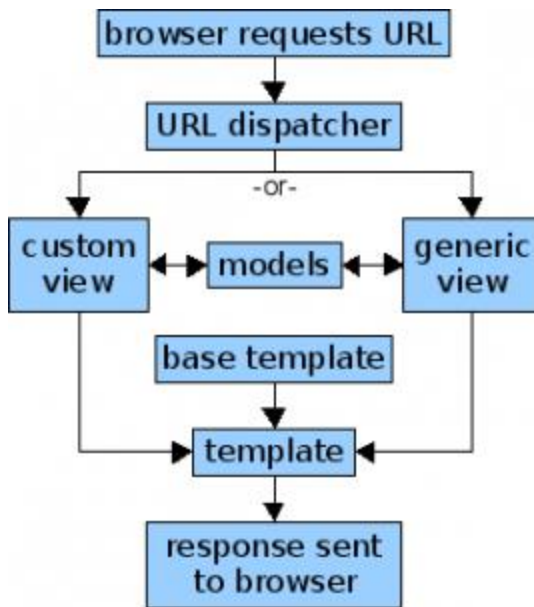
## Models

Você também não precisa se preocupar muito com que banco vai usar – já que o ORM do Django suporta MySQL, PostgreSQL, SQLite e até mesmo Oracle.



# Views

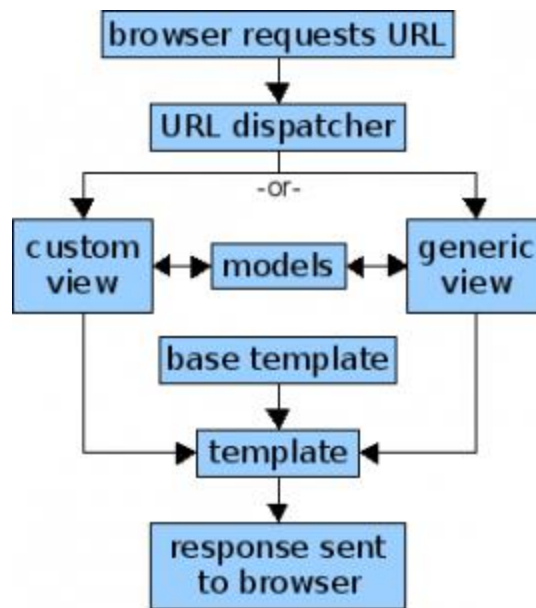
Em views escrevemos as regras de apresentação. Calma lá... não chegamos nos templates ainda. Estamos falando de criar funções que têm por parâmetro um objeto de requisição (request) e por retorno um objeto de resposta (response). O “meio de campo” entre estes extremos é justamente a responsabilidade da sua view..





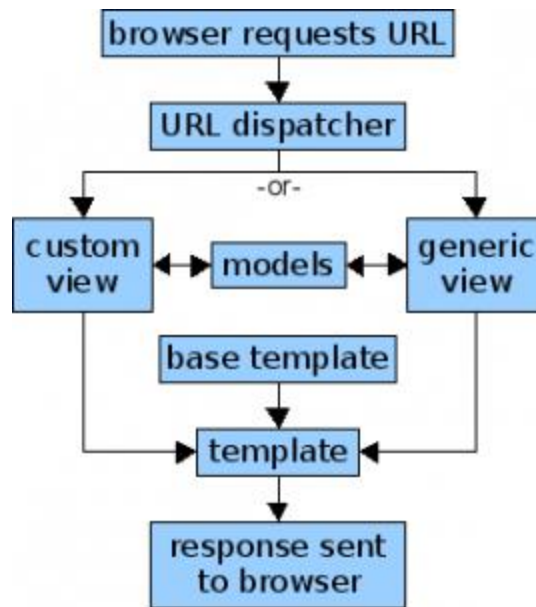
# Views

O que muita gente entende por “regras de negócios do sistema” será escrito na View. É nela que dizemos qual modelo deve ser instanciado, o que ele deve fazer, qual template deve ser importado, como o valor deve ser exibido nele e qual resposta deve ser enviada para o internauta (um HTML, um XML, um SVG, um redirecionamento, um erro 404, um erro 500, etc.).



## Views

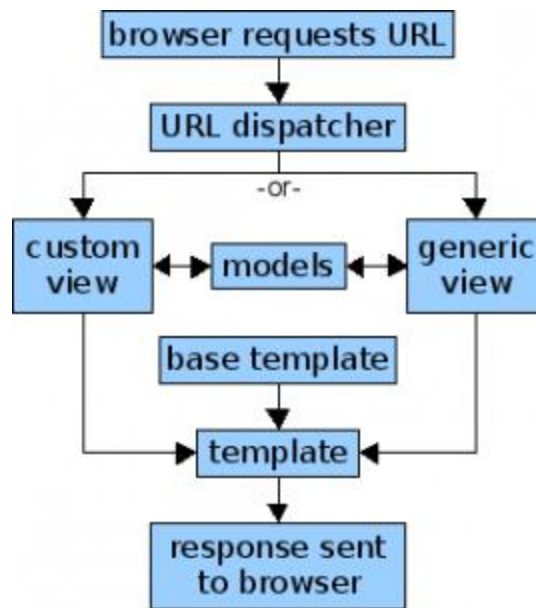
Em Django, escrevemos formulários em classes geralmente situadas no arquivo `forms.py`. Logo, podemos escrever as regras de comportamento de um formulário dentro de sua classe, tirando esta responsabilidade da View. Isto é interessante pois, se usarmos um formulário em mais de uma View não é necessário duplicar código (DRY).



# Templates

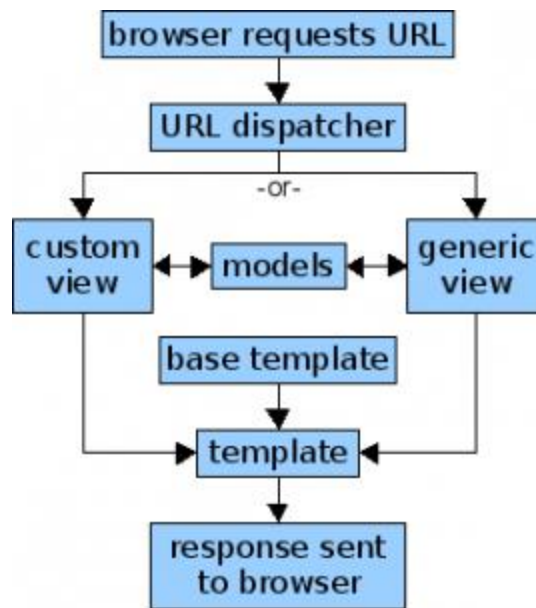
Não engane-se... template não refere-se apenas a HTML!

Podemos escrever templates para HTML, Javascript, CSS, XML, YAML, JSON, SVG, qualquer coisa. Na View você indica qual será o tipo de resposta, o template é só a forma de apresentar o que a View “preparou”.



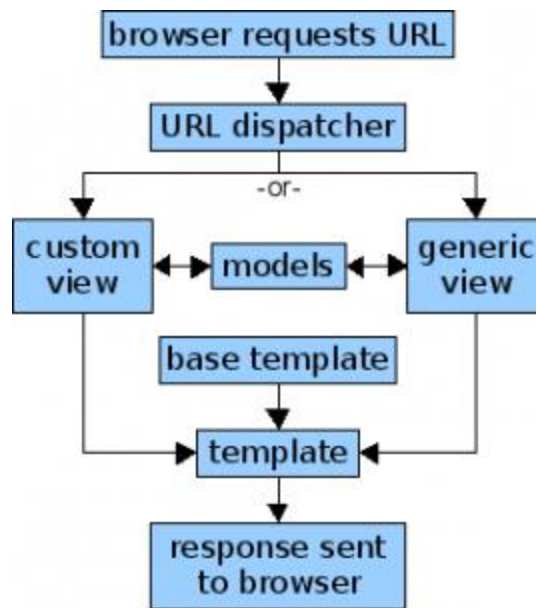
# Templates

O sistema de templates do Django é uma de suas mais notórias funcionalidades. Com ele podemos criar heranças, ou seja, um template base contendo a estrutura básica do seu website e templates específicos que herdam as características deste template base e atribuem/criam suas próprias características. Acredite, controlar as meta-tags do seu website nunca foi tão fácil... e nem é necessário uma aplicação para isso, basta saber um pouco de HTML.



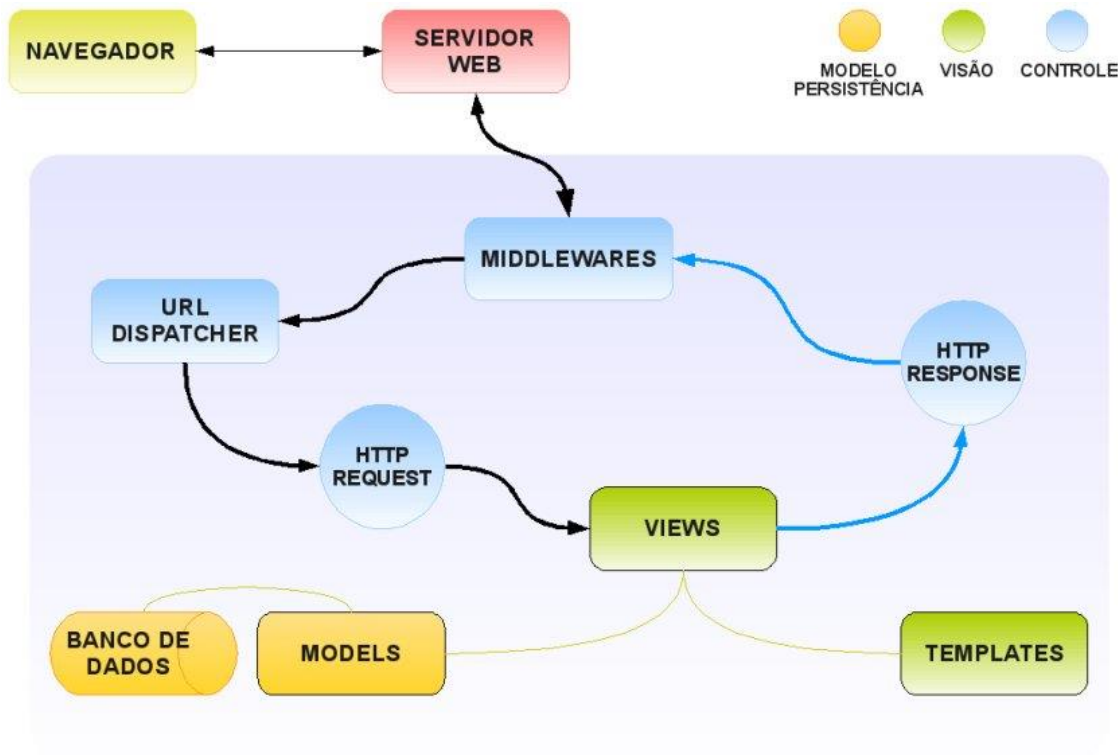
# Templates

Dentro desta estrutura de desenvolvimento, é muito fácil separar as funções do Webprogrammer e do Webdesigner. O sistema de templates do Django possui sintaxe própria e simples. O programador só preocupa-se com os dados que ele deve enviar para o template, o designer só preocupa-se com que dados ele irá receber..



# Modelagem de Sistemas

Dentro desta estrutura de desenvolvimento, é muito fácil separar as funções do Webprogrammer e do Webdesigner.



## Projetos, Aplicações e “Plugabilidade”

Em Django, iniciamos um projeto com a simpática sintaxe em um terminal do seu SO:

```
python manage.py startapp <nome-da-sua-aplicacao>
```

Ele criará uma pasta com o nome do seu projeto que conterá os arquivos:

- **\_\_init\_\_.py**: É o arquivo que determina que aquela pasta é um pacote Python;
- **settings.py**: Arquivos de configurações em XML? Esqueça! Em Django temos a configuração do nosso projeto em um único arquivo .py;

## Projetos, Aplicações e “Plugabilidade”

Em Django, iniciamos um projeto com a simpática sintaxe em um terminal do seu SO:

→ **urls.py**: Este é o “temido” url dispatcher. Você passará bons (e maus) momentos com ele;

→ **manage.py**: O regente da orquestra. É através dele que você executará ações como criar uma aplicação, sincronizar o banco de dados, iniciar o servidor web embutido (somente recomendado para ambiente de desenvolvimento), etc. Esse é um dos caras que fará o seu dia bem melhor com Django.



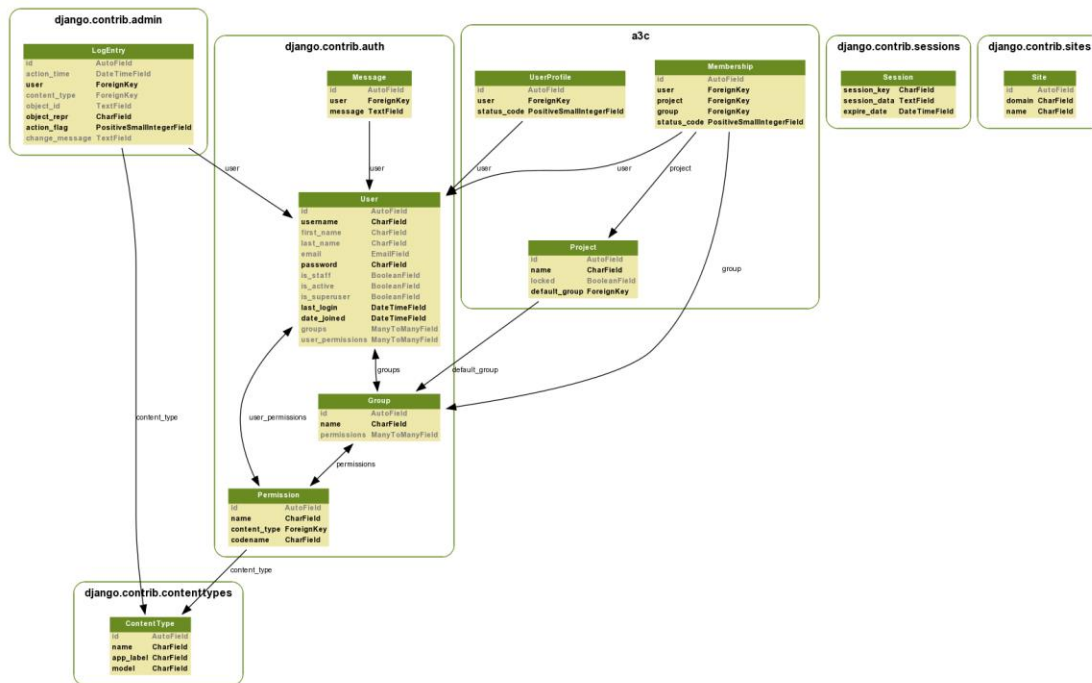
## Projetos, Aplicações e “Plugabilidade”

Novamente, você não precisa executar o **manage.py startapp** para iniciar uma aplicação, é apenas praticidade.

Para o Django, um Projeto é um conjunto de aplicações. Na prática: digamos que você irá desenvolver um website para o cliente Bikes do Diego. Então poderia chamar seu projeto de `bikes_do_diego` e a partir daí desenvolver as aplicações necessárias para o funcionamento do website.

# Projetos, Aplicações e “Plugabilidade”

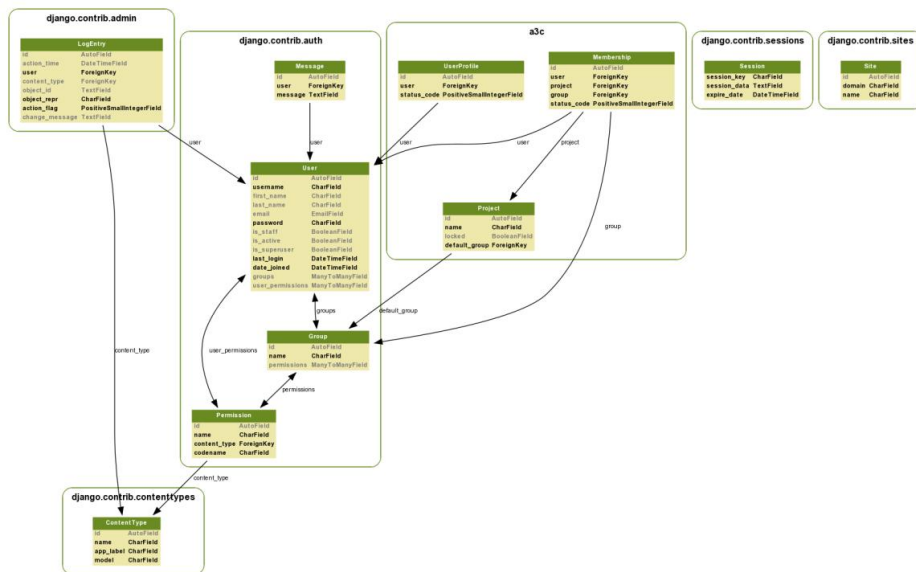
Além do seu “core”, o Django disponibiliza uma série de recursos que os Pythonistas costumam chamar de “baterias inclusas”.



# Projetos, Aplicações e “Plugabilidade”

Dentre os recursos mais comuns, temos à disposição o auth, sites, admin, contenttypes, etc. Sem eles, o seu website funciona. Com eles, você economiza quilômetros de LOC e terá aplicações testadas à exaustão.

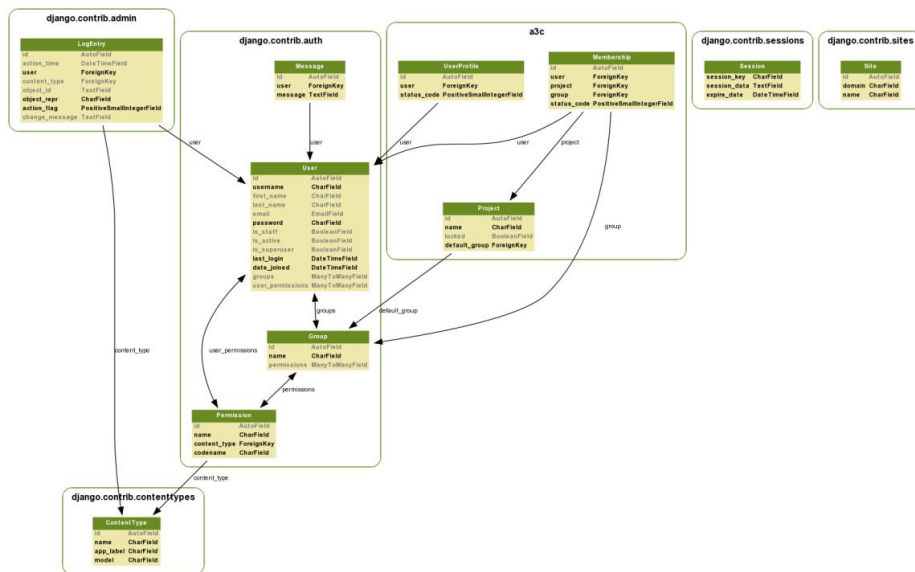
<https://www.profissionaisti.com.br/entendendo-o-django/>



# Projetos, Aplicações e “Plugabilidade”

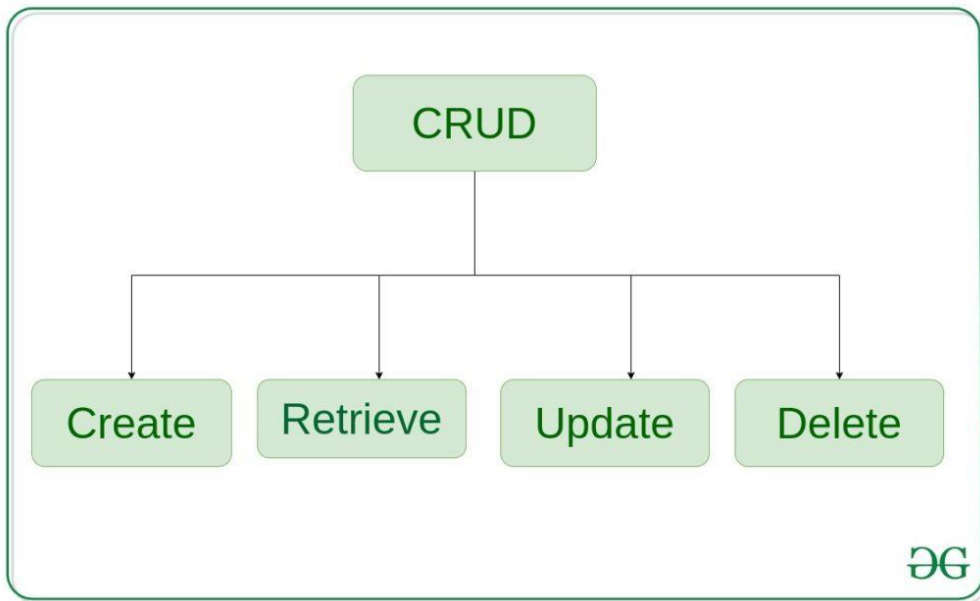
Dentre os recursos mais comuns, temos à disposição o auth, sites, admin, contenttypes, etc. Sem eles, o seu website funciona. Com eles, você economiza quilômetros de LOC e terá aplicações testadas à exaustão.

<https://www.profissionaisti.com.br/entendendo-o-django/>



# CRUD

Este exemplo gira em torno da implementação completa de **Visualizações Baseadas em Classes no Django** (Criar, Recuperar, Atualizar, Excluir). Vamos discutir o que realmente significa CRUD,



# CRUD

Este exemplo gira em torno da implementação completa de **Visualizações Baseadas em Classes no Django** (Criar, Recuperar, Atualizar, Excluir). Vamos discutir o que realmente significa CRUD:

CreateView - cria ou adiciona novas entradas em uma tabela no banco de dados.

Recuperar visualizações - ler, recuperar, pesquisar ou visualizar entradas existentes como uma lista ( ListView ) ou recuperar uma entrada específica em detalhes( DetailView )

UpdateView - atualizar ou editar entradas existentes em uma tabela no banco de dados

DeleteView - excluir, desativar ou remover entradas existentes em uma tabela no banco de dados

# CRUD

Este exemplo gira em torno da implementação completa de **Visualizações Baseadas em Classes no Django** (Criar, Recuperar, Atualizar, Excluir). Vamos discutir o que realmente significa CRUD:

[UpdateView](#) - atualizar ou editar entradas existentes em uma tabela no banco de dados

[DeleteView](#) - excluir, desativar ou remover entradas existentes em uma tabela no banco de dados

[FormView](#) - renderizar um formulário para modelo e manipular os dados inseridos pelo usuário

# Obrigado!

Prof. Dr. Diego Bruno

