# Mapeando banco de dados legados, com Django.

# Parte 1

Já pensou em usar um banco de dados em que não precise cria-lo pelo Django?

Não?

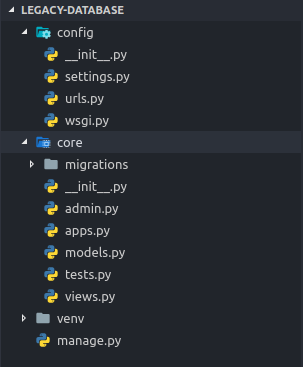
Então, imagine as seguintes situações:

1. Tu estás trabalhando em uma empresa e, nessa empresa, forma-se uma equipe para desenvolver um projeto qualquer, a equipe é composta por um desenvolvedor back-end, outro front-end e um DBA (Database administrator). Então, o DBA cria um banco de dados com todos os relacionamentos entre tabelas, tudo certinho, pronto para uso;
2. Outro caso, talvez o mais comum em nosso país, seria o uso de um banco de dados mais antigo, talvez de um sistema que esta sendo atualizado e a empresa queira usar o mesmo banco de dados.

Dado as duas situações, como faria para usar os bancos de dados igualmente da forma em que está? Iria fazer uma “cópia” do banco de dados e seus relacionamentos, manualmente?

Obvio que não (pelo menos, eu espero), dessa forma, venho lhes apresentar uma solução simples, sem muito esforço para ser usada, usa-se o comando “inspectdb”, para tal finalidade. Sem mais delongas, vamos ao que interessa!

Primeiramente, tens em mente que, suponho que saiba algo sobre python e o framework django, de preferencia em suas versões mais recentes (3.7 e 2.2, respectivamente). Sabendo disso, ja podemos criar a estrutura base de nosso projeto:



*Como pode ver, organizei a estrutura do meu projeto dessa forma, porém, sinta-se à vontade para organizar da forma que lhe for mais conveniente.*

Após a organização basica de nosso projeto, iremos seguir dois passos essenciais para a composição do nosso projeto. Sendo eles, a instalação da lib ”pymysql” e a configuração de nosso banco de dados (acho que ficou explicito que usaremos o MySQL).

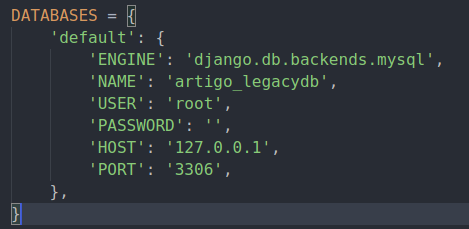
Então, vamos dar continuidade, instalando a lib “pymysql”, da seguinte forma:

legacydatabase_pymysql

*Lembrem-se de estar dentro de seu ambiente virtual, cujo procedimento não cabe nesse artigo, ja que é uma premissa obrigatória, o conhecimento de tal ação.*

Com o componente instalado, vamos dar inicio a nossa configuração do banco de dados, iremos dizer ao Django o nome do banco, o host, o usuario e a senha, para que, o framework saiba onde procurar os dados sempre que for solicitado. Então, vamos lá:

Abra o arquivo “settings.py”, ache o item “DATABASES” e preencha com as informações de seu banco de dados.



Estou informando os dados de um banco local, se você for usar um banco de dados real, use as informações que lhe serão fornecidas previamente.

Explicamando brevemente:

**ENGINE:** É, basicamente, a referencia de seu banco de dados, no caso desse artigo, MySQL;

**NAME:** Será o nome de seu banco de dados;

**USER:** Usuário “root” de seu banco de dados;

**PASSWORD:** Senha do seu usuário “root”;

**HOST:** É o endereço em que se encontra seu banco de dados;

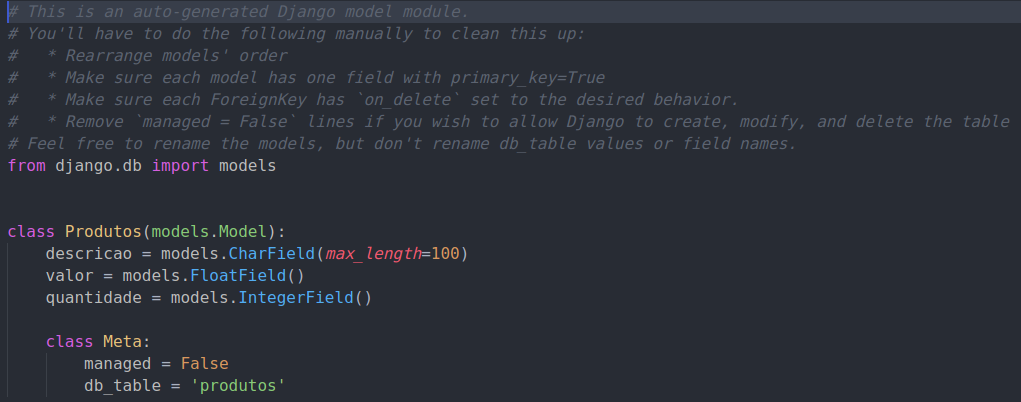
**PORT:** É a porta que o modelo usa, no caso do MySQL, usa-se a porta 3306, por padrão.

Pronto, passada a parte chata, agora iremos realizar o procedimento crucial para que todo esse artigo faça algum sentido (ou não), iremos usar a seguinte linha para *mapear* as tabelas de nosso banco de dados legado:

legacydatabase_inspectdb

Nesse script dizemos ao django mapear o banco de dados que foi fornecido em nossas configurações do projeto e, pedimos que ele pegue a estrutura da tabela “produtos” e jogue para dentro do arquivo “models.py” que, esta dentro do diretório “core”.

Dessa simples forma, obtivemos o seguinte resultado:



Nessa imagem vemos que, a estrutura de nossa tabela foi toda copiada para dentro do nosso arquivo. Além disso, vem com ela as configurações da tabela, dentro da classe “Meta”.

Agora, ja podemos realizar alguns testes com nosso banco de dados, que tal inserirmos e listarmos alguns dados na tela?

Veja a sequência, na parte 2 desse artigo: (URL)