

Integrando Python com BDs relacional e NoSQL

Formação Python Developer

Juliana Mascarenhas

Tech Education Specialist DIO / Owner @Simplificandoredes e @SimplificandoProgramação

Mestre em modelagem computacional | Cientista de dados

@in/juliana-mascarenhas-ds/





<https://github.com/julianazanelatto>

Juliana Mascarenhas

Tech Education Specialist

@SimplificandoRedes

@SimplificandoProgramação

Cientista de dados

Desenvolvedora Java/Python

Me Modelagem Computacional - LNCC

Objetivo Geral

Trabalhar com a principal biblioteca ORM de integração de SGBDs com Python – SQLAlchemy. Além disso, não podemos deixar de fora os bancos de dados NoSQL. Sendo assim, utilizaremos Pymongo.

Etapa 1

Integrando Python com SQLite usando SQLAlchemy

// Integração com Python

Conhecendo a Biblioteca SQLAlchemy

// Integração com Python

SQLAlchemy



- Framework – open source
- Licença MIT – 2019
- Mapeamento Objeto Relacional



SQLAlchemy



Database	Fully tested in CI	Normal support	Best effort
Microsoft SQL Server	2017	2012+	2005+
MySQL / MariaDB	5.6, 5.7, 8.0 / 10.4, 10.5	5.6+ / 10+	5.0.2+ / 5.0.2+
Oracle	11.2, 18c	11+	8+
PostgreSQL	9.6, 10, 11, 12, 13, 14	9.6+	8+
SQLite	3.21, 3.28+	3.12+	3.7.16+

[Documentação](#)

Database	Dialect
Action Avalanche, Vector, Actian X, and Ingres	sqlalchemy-ingres
Amazon Redshift (via psycopg2)	sqlalchemy-redshift
Apache Drill	sqlalchemy-drill
Apache Druid	pydruid
Apache Hive and Presto	PyHive
Apache Solr	sqlalchemy-solr
CockroachDB	sqlalchemy-cockroachdb
CrateDB ^[1]	crate-python
EXASolution	sqlalchemy_exasol
Elasticsearch (readonly)	elasticsearch-dbapi

SQLAlchemy

- Dialectos externos

SQLAlchemy

SQLAlchemy



- Vastamente utilizado
- Framework completo
- Flexibilização do SQL
- Segurança nas instruções



SQLAlchemy

Recursos:

- ORM e CORE
- Suporte a dialetos
- Manipulação do BD por meio de Transações
- Suporte a Queries complexas via ORM
- Config: relações e relacionamentos
- Sessões, eventos ...

SQLAlchemy

Extensões:

- I/O assíncrono
- Associação com proxy
- Indexação
- APIs especiais
- ...

```
from sqlalchemy import Column, JSON, Integer
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.ext.indexable import index_property

Base = declarative_base()

class Person(Base):
    __tablename__ = 'person'

    id = Column(Integer, primary_key=True)
    data = Column(JSON)

    name = index_property('data', 'name')
```

[Documentação](#)

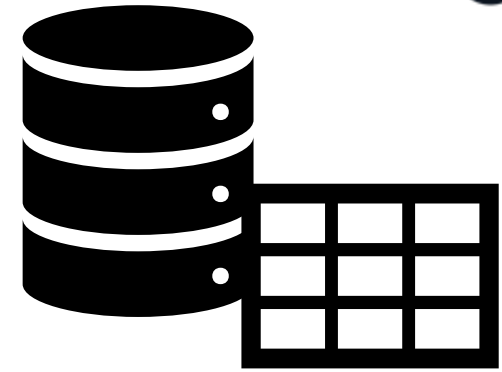
ORM

- ORM - Object Relational Mapping
- Objeto -> Modelo Relacional
- Mais fácil para o programador

CRUD

QUERY

CONEXÃO



ORM

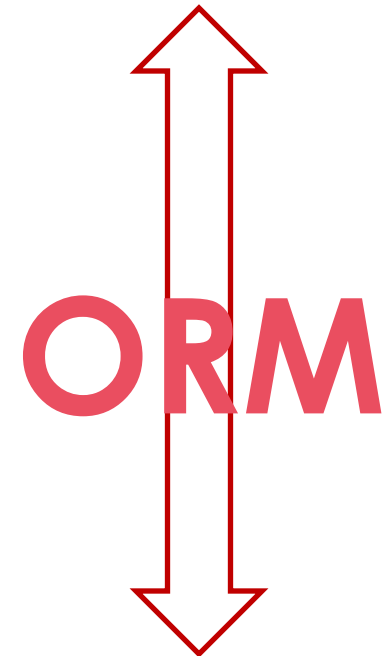
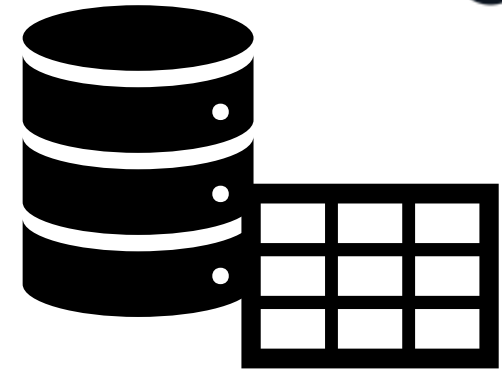
{OOP}

ORM

Vantagens

- Menos código
- Melhor manutenção
- Utilização de conectores
- Indicado para CRUDs

SQLAlchemy



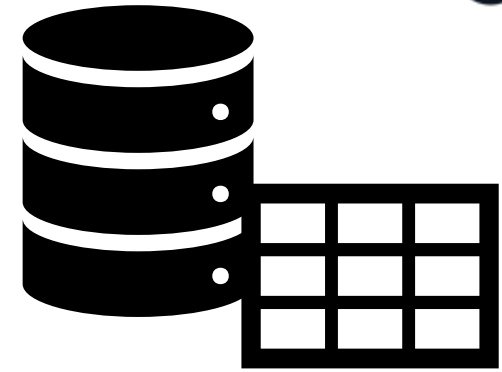
{ OOP }

Entidade

```
>>> from sqlalchemy import Column, Integer, String
>>> class User(Base):
...     __tablename__ = 'users'
...
...     id = Column(Integer, primary_key=True)
...     name = Column(String)
...     fullname = Column(String)
...     nickname = Column(String)
...
...     def __repr__(self):
...         return "<User(name='%s', fullname='%s', nickname='%s')>" % (
...             self.name, self.fullname, self.nickname)
```

ORM

Se pergunte como está
seu modelo de dados



Desvantagens

- Complexidade X ORM
- Dependência do ORM
- Depende do projeto
- Retorno das consultas sem necessidade de programar na "mão"

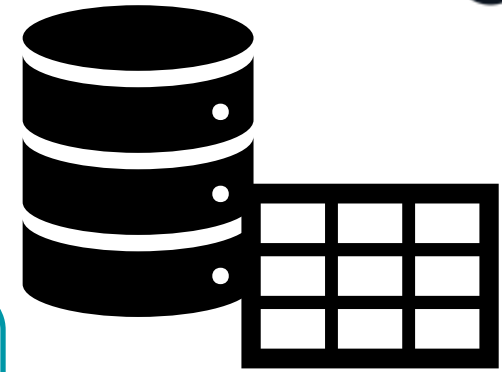
SQLAlchemy

ORM

{ OOP }

ORM

SQLAlchemy



Nem sempre será a query mais otimizada

- Perda de performance
- Deixa de estudar SQL e perde a eficiência na construção
- Número de instâncias x velocidade

ORM

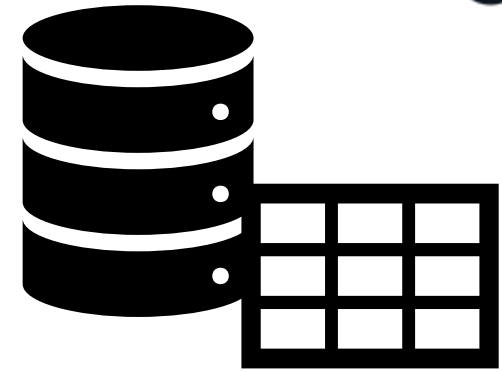
{OOP}

ORM

Por que usar?

- Troca de SGBD mais facilitada
- Modelo MVC
- Diminuição do DRY
- Evita problemas de segurança

SQLAlchemy

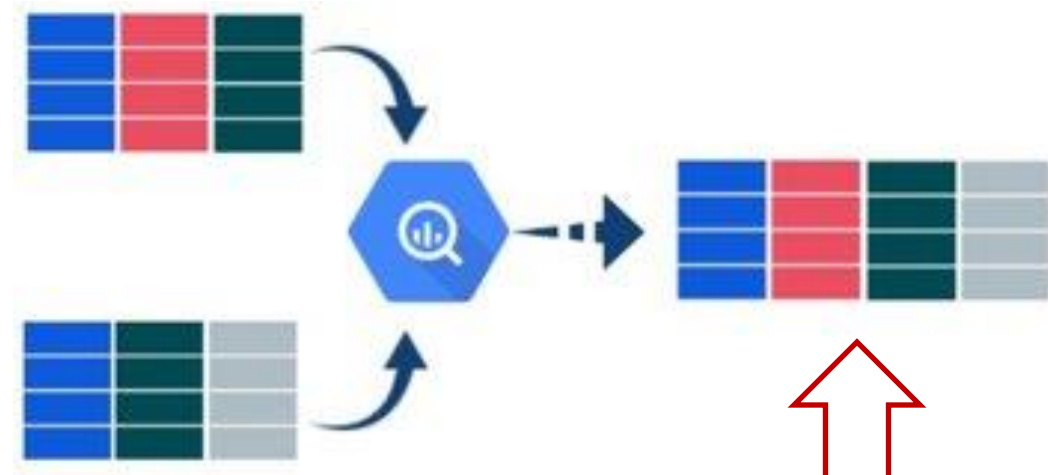


ORM

{OOB}

ORM & SQL

Uso de views



- Melhor dos dois mundos
- Qual a melhor ferramenta para o seu problema?

ORM

{OOP}

Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)



Para saber mais

Python e SQLite

- <https://www.sqlalchemy.org/>
- <https://docs.sqlalchemy.org/en/14/tutorial/index.html>
- <https://docs.sqlalchemy.org/en/14/orm/quickstart.html>
- <https://docs.sqlalchemy.org/en/14/orm/tutorial.html>



Para saber mais

Dialetos

- <https://docs.sqlalchemy.org/en/14/dialects/index.html>
- SQLite - <https://docs.sqlalchemy.org/en/14/dialects/sqlite.html>
- MySQL - <https://docs.sqlalchemy.org/en/14/dialects/mysql.html>

Subqueries e demais recursos

- https://docs.sqlalchemy.org/en/14/tutorial/data_select.html#tutorial-select
[data](#)



Para saber mais

Modelo ORM

- <https://docs.sqlalchemy.org/en/14/orm/index.html>
- <https://docs.sqlalchemy.org/en/14/orm/quickstart.html>

Modelo Core

- <https://docs.sqlalchemy.org/en/14/core/index.html>
- <https://docs.sqlalchemy.org/en/14/core/tutorial.html>



Integrando Python MongoDB - Pymongo

Formação Python Developer

Juliana Mascarenhas

Tech Education Specialist DIO / Owner @Simplificandoredes e @SimplificandoProgramação

Mestre em modelagem computacional | Cientista de dados

@in/juliana-mascarenhas-ds/



Etapa 2

Integrando Python com MongoDB usando Pymongo

// Integração com Python

O que são Pymongo e MongoDB?

// Integração com Python

Pymongo

[Documentação](#)

- Módulo para MongoDB
- Autor: MongoDB Python Team
- Apache Software License
- Formato: BSON

```
{  
  _id: ObjectId("5f339953491024badf1138ec"),  
  title: "MongoDB Tutorial",  
  isbn: "978-4-7766-7944-8",  
  published_date: new Date('June 01, 2020'),  
  author: {  
    first_name: "John",  
    last_name: "Doe"  
  }  
}
```

<https://www.mongodbtutorial.org/getting-started/mongodb-basics/>

Pymongo

[Documentação](#)

Installing with pip

We recommend using [pip](#) to install pymongo on all platforms:

```
$ python3 -m pip install pymongo
```

To get a specific version of pymongo:

```
$ python3 -m pip install pymongo==3.5.1
```

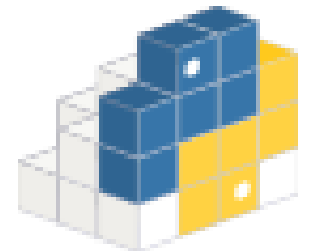
To upgrade using pip:

```
$ python3 -m pip install --upgrade pymongo
```

Pymongo

[Documentação](#)

- Interação com documentos
- Coleções e demais recursos do MongoDB
- Suporte MongoDB 3.6, 4.0, 4.2, 4.4, and 5.0.



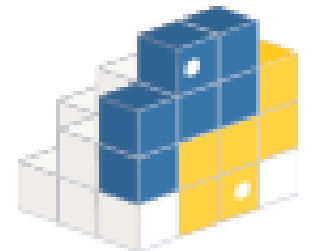
Pymongo

[Documentação](#)

```
>>> import pymongo
>>> client = pymongo.MongoClient("localhost", 27017)
>>> db = client.test
>>> db.name
'test'
>>> db.my_collection
Collection(Database(MongoClient('localhost', 27017), 'test'), 'my_collection')
>>> db.my_collection.insert_one({"x": 10}).inserted_id
ObjectId('4aba15ebe23f6b53b0000000')
>>> db.my_collection.insert_one({"x": 8}).inserted_id
ObjectId('4aba160ee23f6b543e000000')
>>> db.my_collection.insert_one({"x": 11}).inserted_id
ObjectId('4aba160ee23f6b543e000002')
>>> db.my_collection.find_one()
{'x': 10, '_id': ObjectId('4aba15ebe23f6b53b0000000')}
>>> for item in db.my_collection.find():
...     print(item["x"])
...
10
8
11
```

MongoDB

- Banco de Dados NoSQL
- Orientados a documentos
- Flexível (NoSQL) x estruturado e rígido (SQL)
- Schema opcional



MongoDB



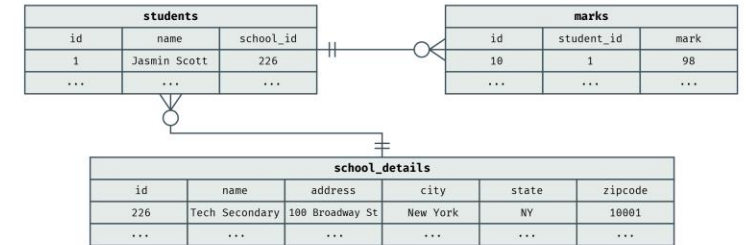
MongoDB

```
{
  "_id": 1,
  "student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
  "marks": [98, 93, 95, 88, 100],
}
```

mongo

```
> db.students.find({"student_name":
  "Jasmin Scott"})
```

SQL



Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...

sql

```
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

<https://www.mongodb.com/docs/>

MongoDB



Coleções

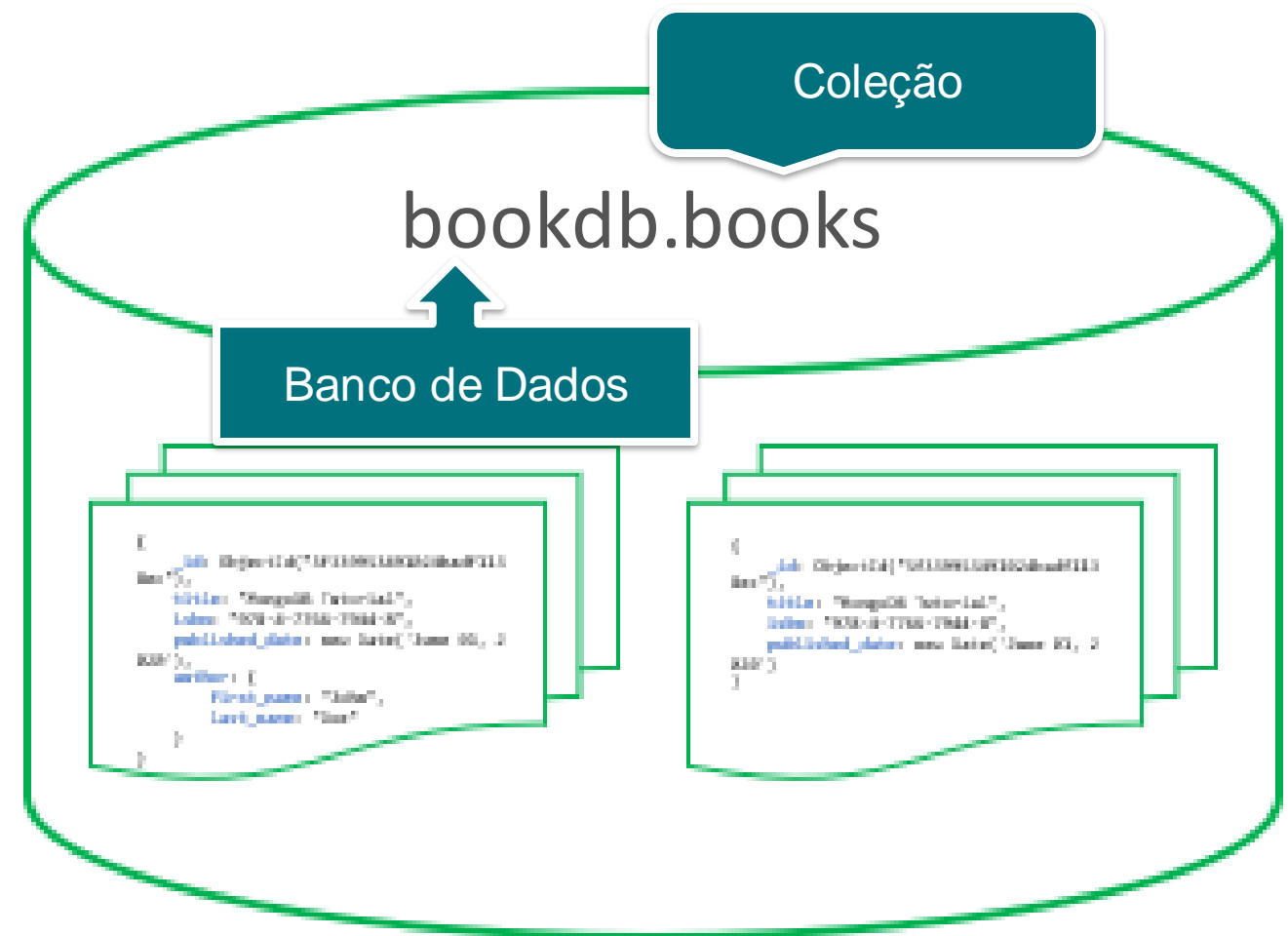
```
{
  _id: ObjectId("5f339953491024badf1138ec"),
  title: "MongoDB Tutorial",
  isbn: "978-4-7766-7944-8",
  published_date: new Date('June 01, 2020'),
  author: {
    first_name: "John",
    last_name: "Doe"
  }
}
```

<https://www.mongodbtutorial.org/getting-started/mongodb-basics/>

MongoDB



Namespace



<https://www.mongodbtutorial.org/getting-started/mongodb-basics/>

Diferenças entre MongoDB e o Modelo Relacional

// Integração com Python

MongoDB x SQL

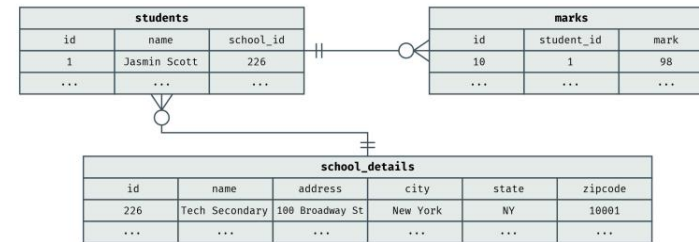
MongoDB

```
{
  "_id": 1,
  "student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
  "marks": [98, 93, 95, 88, 100],
}
```

mongo

```
> db.students.find({"student_name":
  "Jasmin Scott"})
```

SQL



Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...

sql

```
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

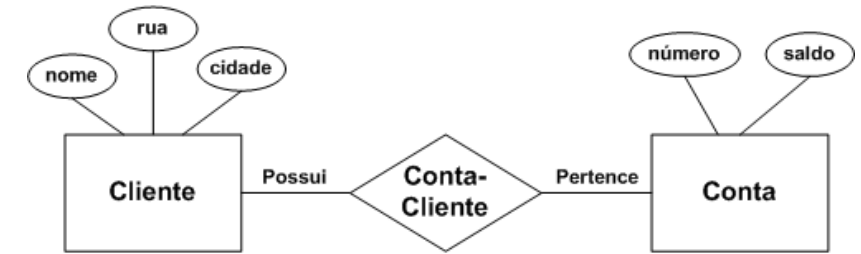
MongoDB x SQL



MONGODB	MODELO RELACIONAL
Documento	Instância (linha)
Campo (field)	Coluna (atributo)
Coleção	Tabelas



MongoDB x SQL



- Documento
- Flexível & Dinâmica
- Escalável
- Melhor do JOINS em "excesso"



- Estrutura rígida
- ACID – Transacional
- Dados dispersos
- Diferentes perspectivas

Relacional



```
create database if not exists first_example;
use first_example;
CREATE TABLE person(
    person_id smallint unsigned,
    fname varchar(20),
    lname varchar(20),
    gender enum('M','F'),
    birth_date DATE,
    street varchar(30),
    city varchar(20),
    state varchar(20),
    country varchar(20),
    postal_code varchar(20),
    constraint pk_person primary key (person_id)
);
```

```
desc person;
```

```
insert into person values      ('5','Roberta','Silva','F', '1979-08-21',
                                'rua tal', 'Cidade J', 'RJ', 'Brasil', '26054-89'),
                                ('6','Luiz','Silva','M', '1979-08-21',
                                'rua tal', 'Cidade J', 'RJ', 'Brasil', '26054-89');
```

```
select * from person;
```

MongoDB

```
MongoDB Shell ▼  
  
db.inventory.insertMany([  
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },  
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },  
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },  
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },  
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }  
]);
```

<https://www.mongodb.com/docs/manual/tutorial/query-documents/>

```
MongoDB Shell ▼  
  
db.inventory.find( {} )
```

```
{  
  "first_name": "John",  
  "last_name": "Doe",  
  "age": 22,  
  "skills": ["Programming", "Databases", "API"]  
}
```

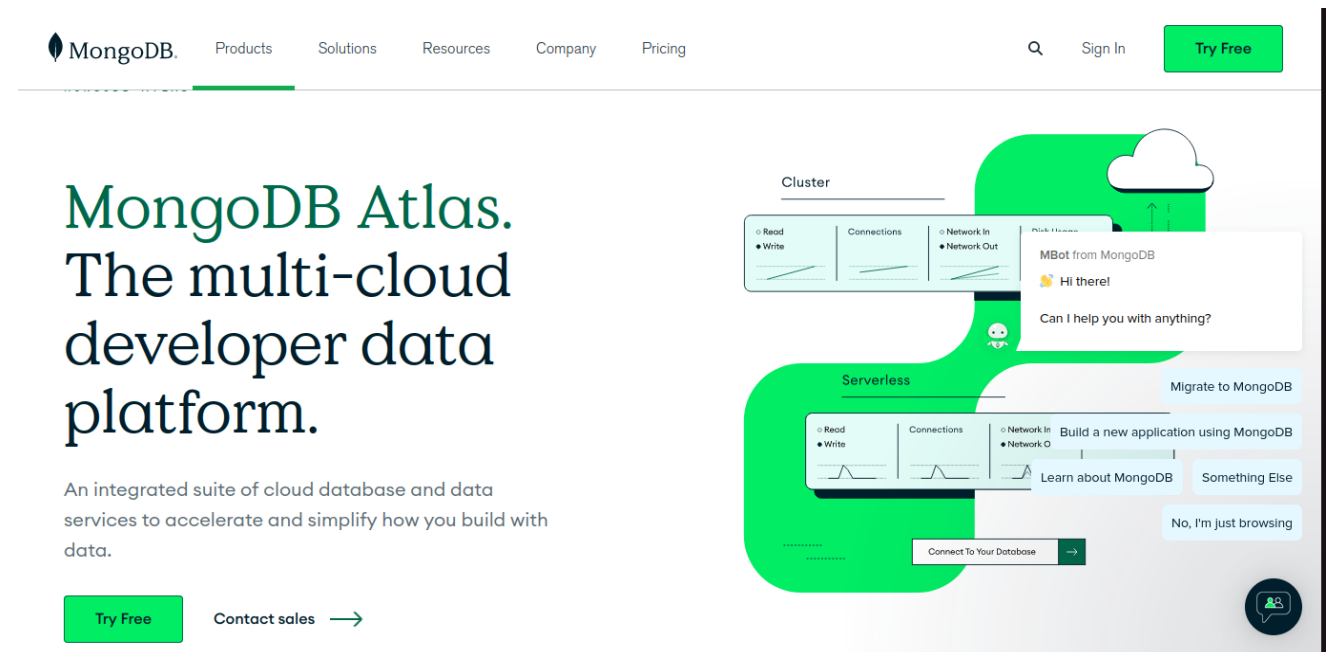


MongoDB Atlas - Instância em nuvem

// Integração com Python

MongoDB Atlas

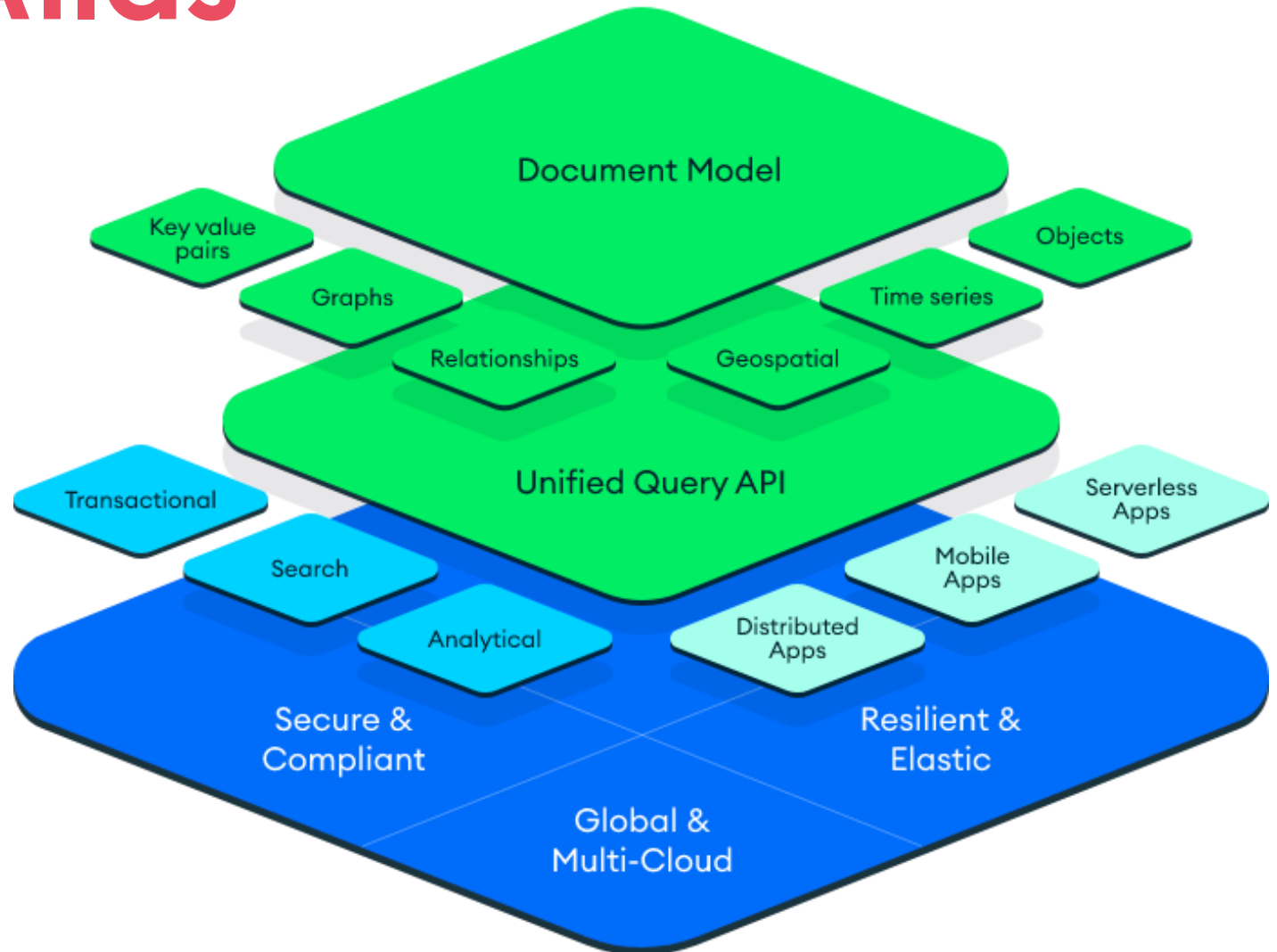
- Plataforma cloud
- Azure, AWS e GCP
- Deploy



[MongoDB Atlas](#)

MongoDB Atlas

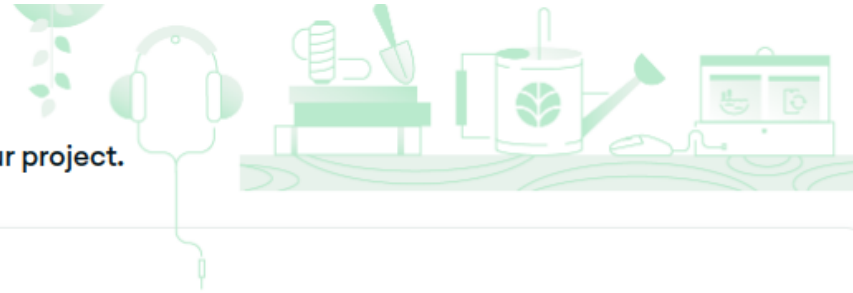
- Documentos
- UQ-API
- Multi-cloud



MongoDB Atlas

Welcome to Atlas! 

Tell us a few things about yourself and your project.



What is your goal today?

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

- ☐ Build a new application
- ☐ Migrate an existing application
- ☐ Learn MongoDB
- ☐ Explore what I can build

What type of application are you building?

Select...

MongoDB Atlas

What type of application are you building?


Content Management ▼

What is your preferred language?

We'll use this to customize code samples and content we share with you. You can always change this later.

 Python ▼

MongoDB Atlas




MONGODB.ATLAS

Deploy a cloud database

Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.

NEW

 **Serverless**


For application development and testing, or workloads with variable traffic. Minimal configuration required.

- ✓ Pay only for the operations you run
- ✓ Resources scale seamlessly to meet your workload
- ✓ Always-on security and backups

Create

Starting at
\$0.10/1M reads

ADVANCED

 **Dedicated**


For production applications with sophisticated workload requirements. Advanced configuration controls.

- ✓ Network isolation and fine-grained access controls
- ✓ On-demand performance advice
- ✓ Multi-region and multi-cloud options available

Create

Starting at
\$0.08/hr*
*estimated cost \$56.94/month

FREE


 **Shared**

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at
FREE



MongoDB Atlas

[CLUSTERS](#) > [CREATE A SHARED CLUSTER](#)

Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Serverless

Dedicated

FREE Shared


For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.


No credit card required to start. Upgrade to dedicated clusters for full functionality.


Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region

AWS, Sao Paulo (sa-east-1) ▾

aws

Google Cloud


Azure

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)


Create Cluster





MongoDB Atlas


Cloud Provider & Region

GCP, Sao Paulo (southamerica-east1) ▼









★ Recommended region ⓘ  Dedicated tier region ⓘ



NORTH AMERICA / SOUTH AMERICA



EUROPE / MIDDLE EAST / AFRICA



ASIA PACIFIC



 Sao Paulo (southamerica-east1) ★


 Iowa (us-central1) ★



 South Carolina (us-east1) ★ 



 N. Virginia (us-east4) ★ 



 Los Angeles (us-west2) ★ 



 Salt Lake City (us-west3) ★ 


 Belgium (europe-west1) ★


 Warsaw (europe-central2) ★ 


 Finland (europe-north1) ★ 


 London (europe-west2) ★ 


 Frankfurt (europe-west3) ★ 


 Netherlands (europe-west4) ★


 Mumbai (asia-south1) ★

 Singapore (asia-southeast1) ★

 Seoul (asia-northeast3) ★

 Jakarta (asia-southeast2) ★

 Taiwan (asia-east1) ★


 Tokyo (asia-northeast1) ★

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)

Create Cluster



[25]

Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)



Para saber mais

Python e MongoDB

- <https://pymongo.readthedocs.io/en/stable/>
- <https://pymongo.readthedocs.io/en/stable/tutorial.html>
- <https://www.mongodb.com/docs/>
- <https://www.mongodb.com/json-and-bson>
- <https://www.mongodbtutorial.org/getting-started/mongodb-basics>



Para saber mais

Python e MongoDB

- <https://acervolima.com/consultas-aninhadas-em-pymongo/#:~:text=PyMongo%20%C3%A9%20um%20m%C3%B3dulo%20Python,est%C3%A3o%20no%20formato%20JSON%20bin%C3%A1rio>
- <https://www.mongodb.com/docs/manual/tutorial/query-documents/>

pip install pymongo

pip install build-essential python3-dev

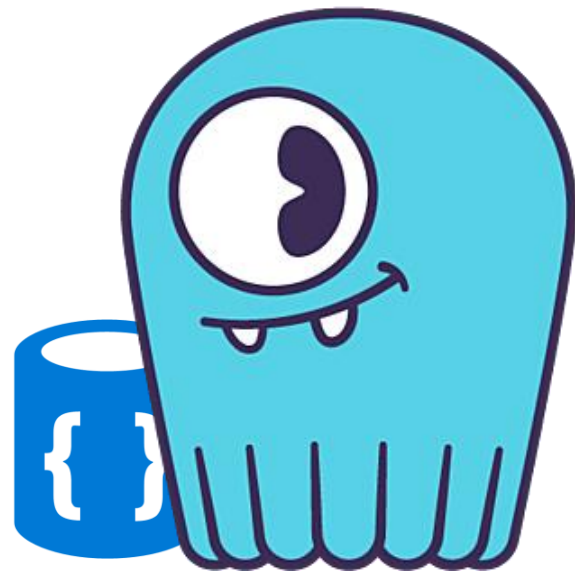


Primeiros Passos com o *Framework* Django

Prof. Dr. Diego Bruno

Education Tech Lead na DIO

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



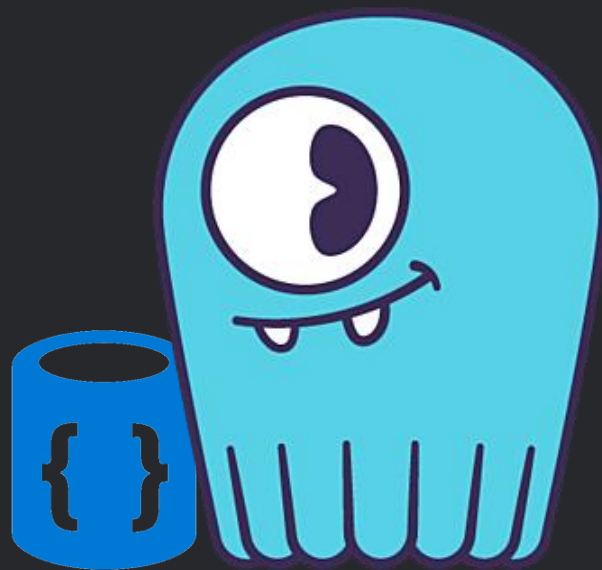
Roteiro para nossa aula de hoje

- 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?

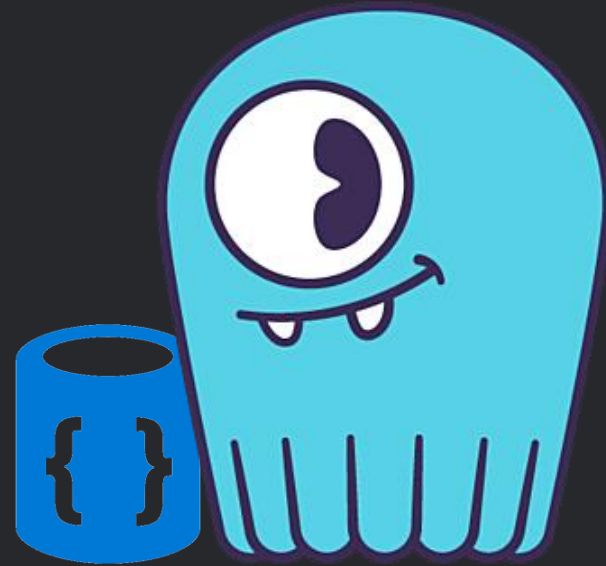
De onde veio?

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

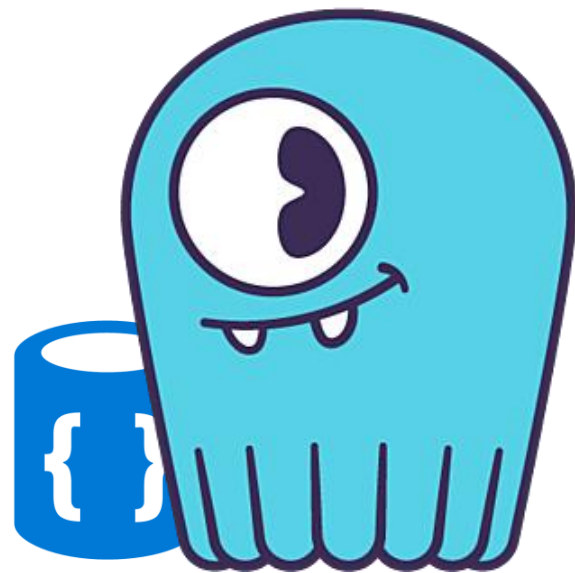


Trabalhando com o *Framework* Django

Prof. Dr. Diego Bruno

Education Tech Lead na DIO

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



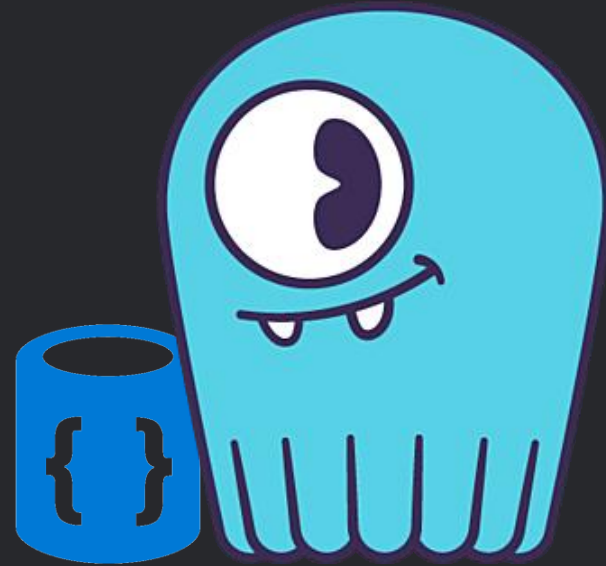
Roteiro para nossa aula de hoje

- 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.

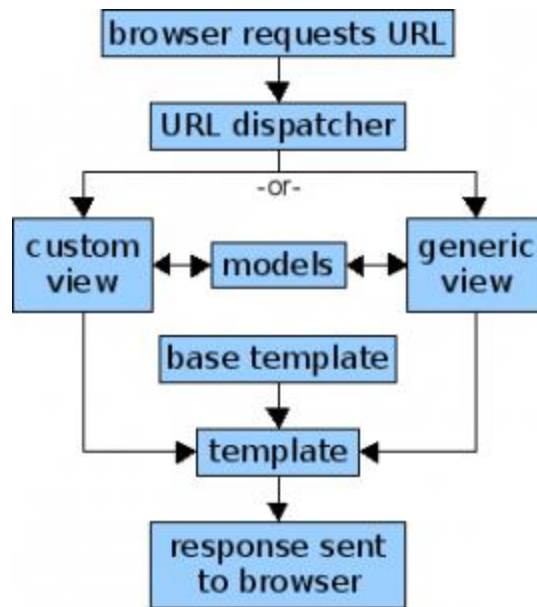




Real Python

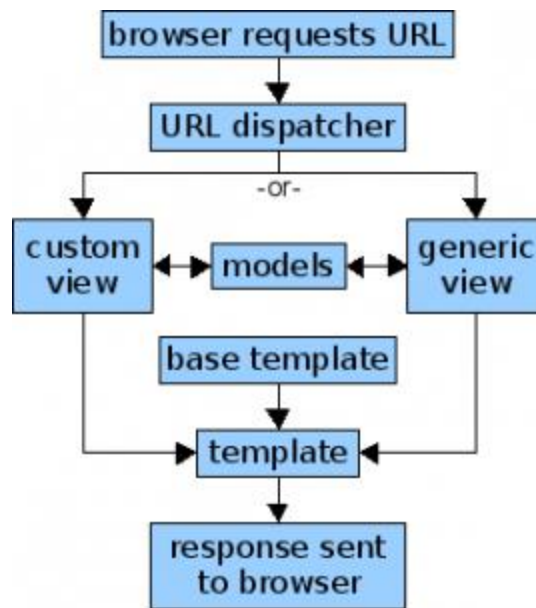
MTV – Model Template View

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



MTV – Model Template View

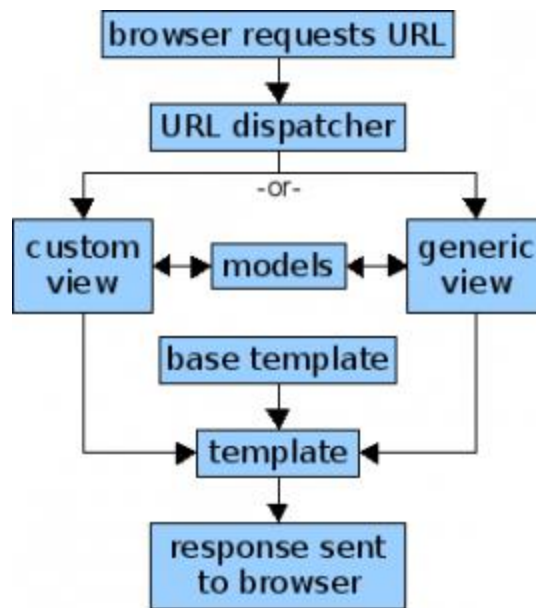
Todo o desenvolvedor deve ao menos saber qual o objetivo deste padrão de desenvolvimento. Separa-se as regras de negócios (controlador), os dados e métodos de acessos aos mesmo (modelo) e as regras de apresentação (visualização). Desse modo, caso ocorra alguma alteração na sua camada de visualização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



MTV – Model Template View

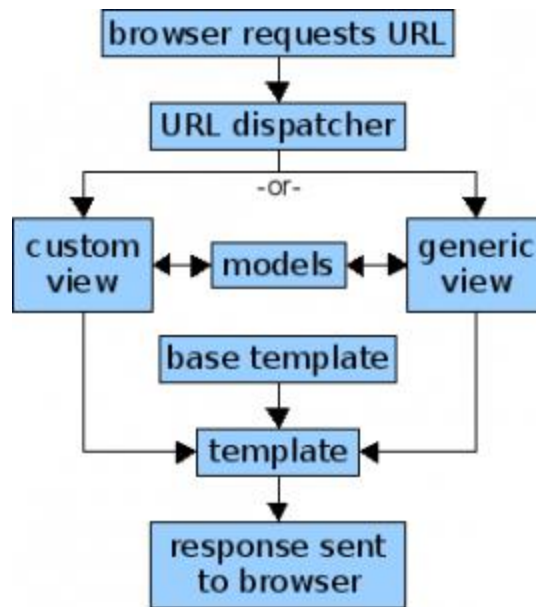
No caso do Django, há uma variação deste modelo que é o MTV (Model Template View).

Aqui entramos em um assunto que gera bastante discussão entre os iniciantes Django: *aonde raios está o Controller?*



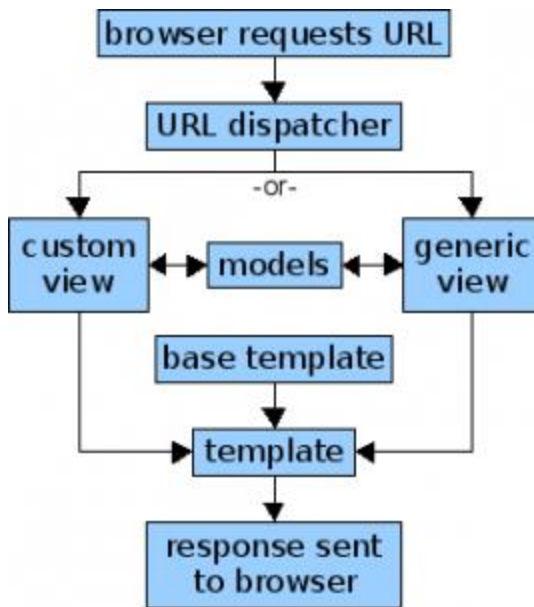
O Framework é o Controlador

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**.



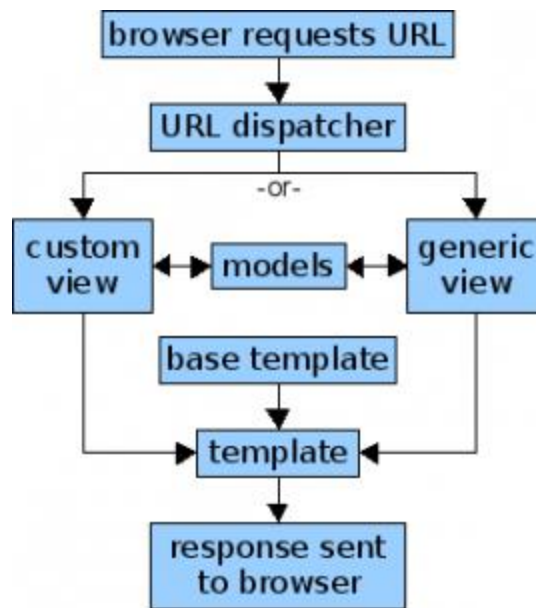
O Framework é o Controlador

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.



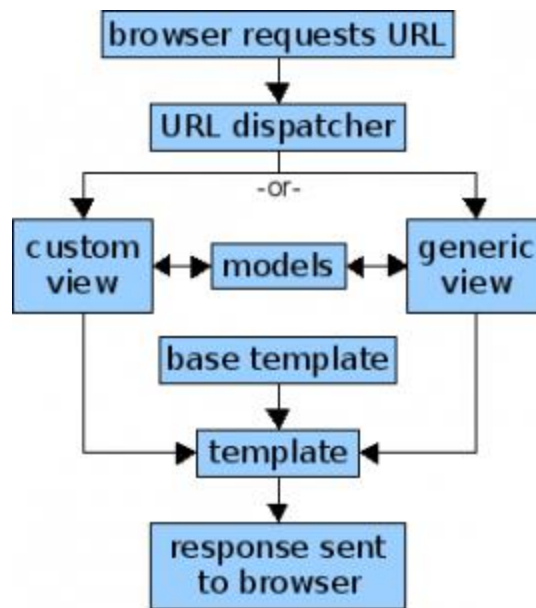
Models

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.



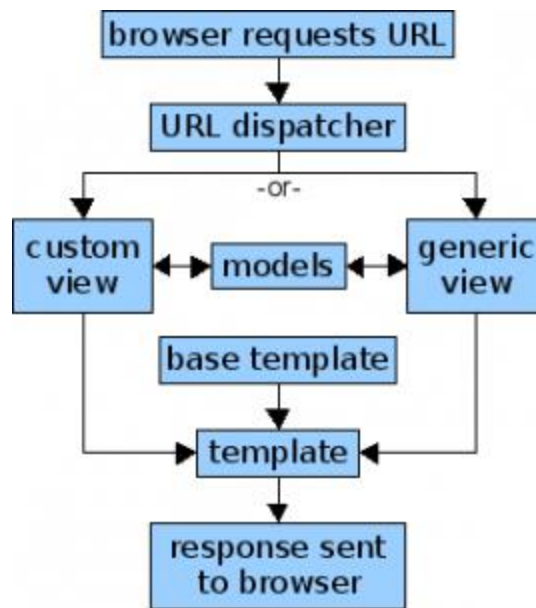
Models

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.



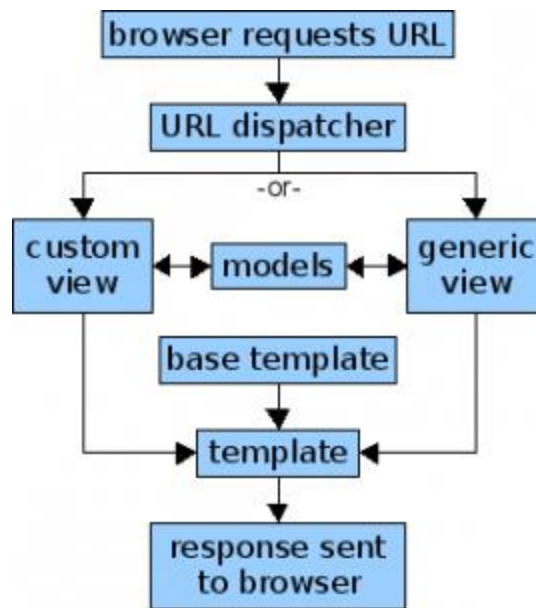
Views

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..



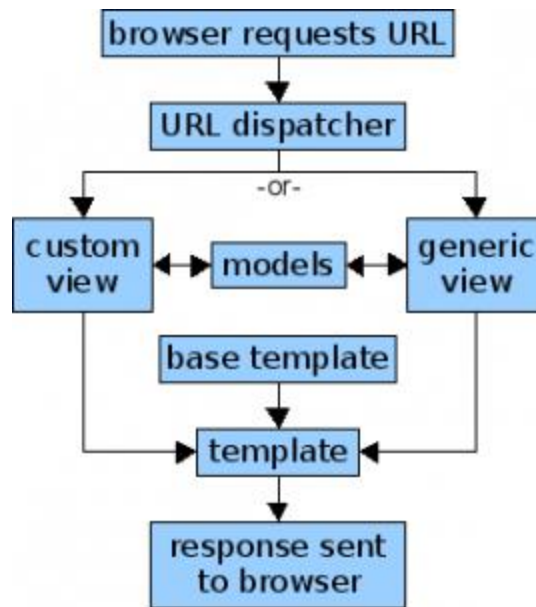
Views

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

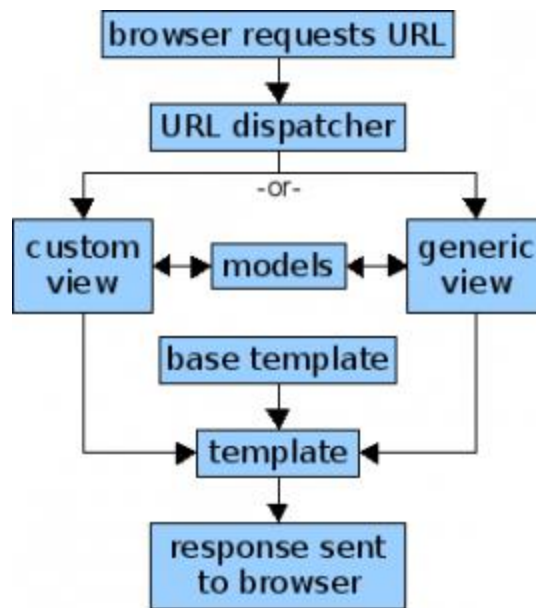
Em Django, escrevemos formulários em classes geralmente situadas no arquivo `forms.py`. Logo, podemos escrever as regras de comportamento de um formulário dentro de sua classe, tirando esta responsabilidade da View. Isto é interessante pois, se usarmos um formulário em mais de uma View não é necessário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



Templates

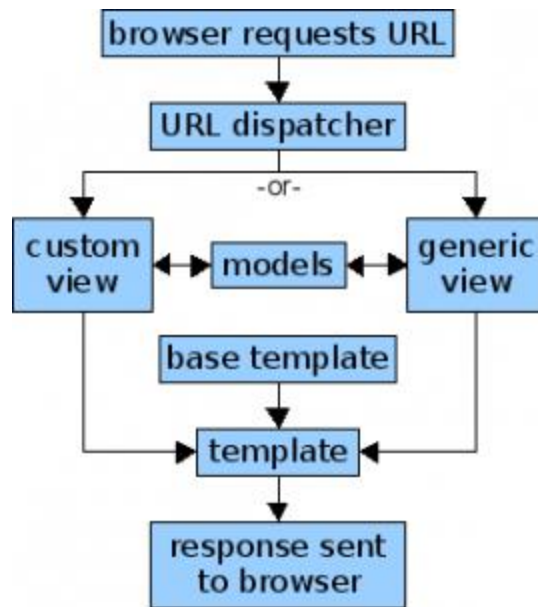
Não engane-se... template não refere-se apenas a HTML!

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”.



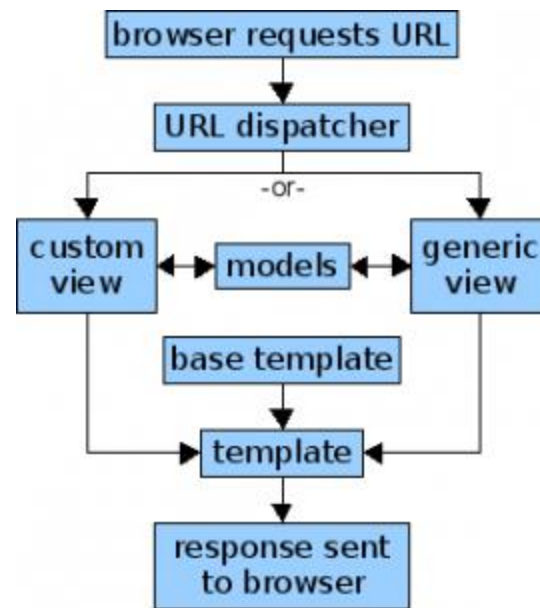
Templates

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.



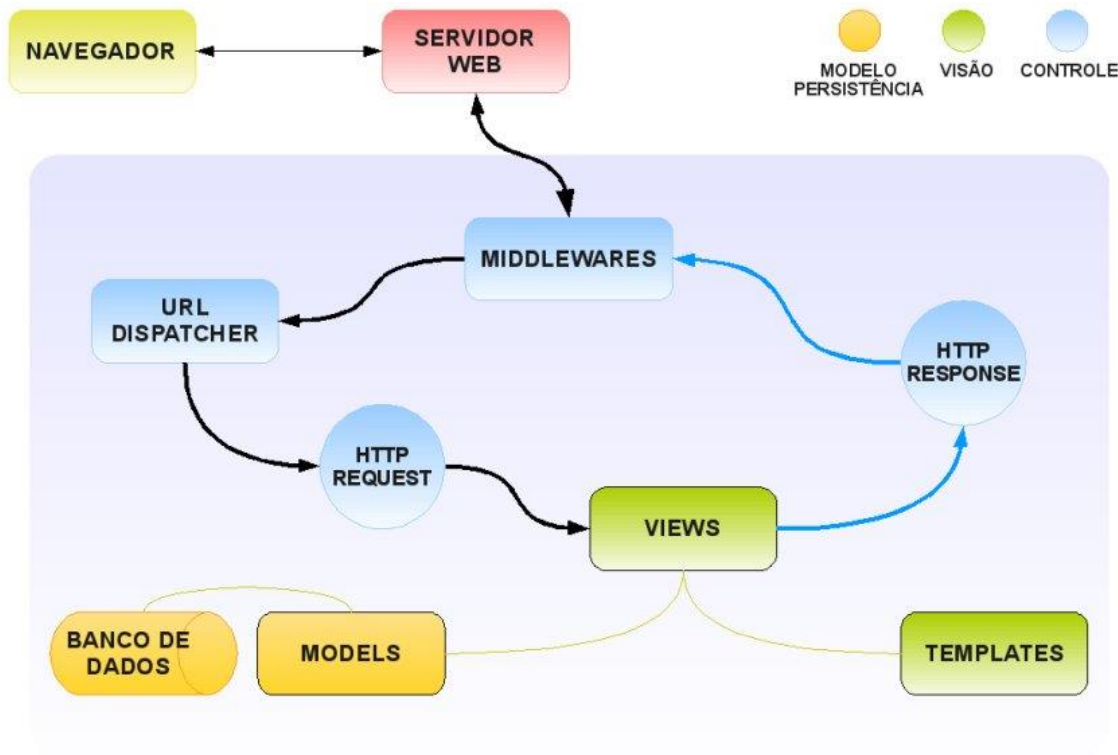
Templates

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.

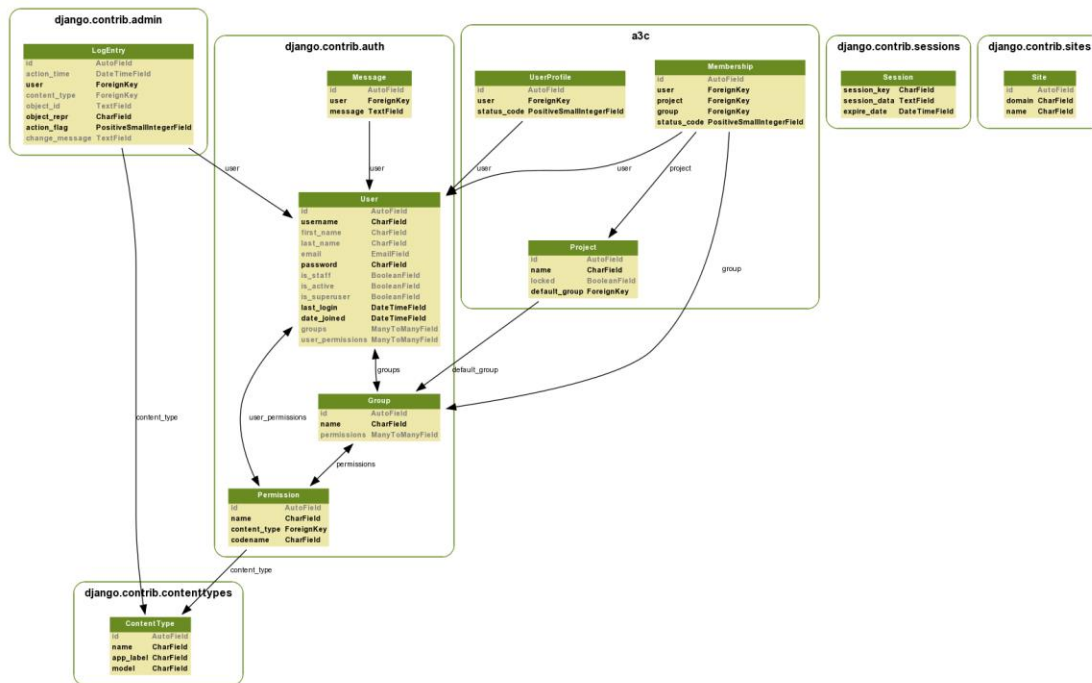
Projetos, Aplicações e “Plugabilidade”

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”

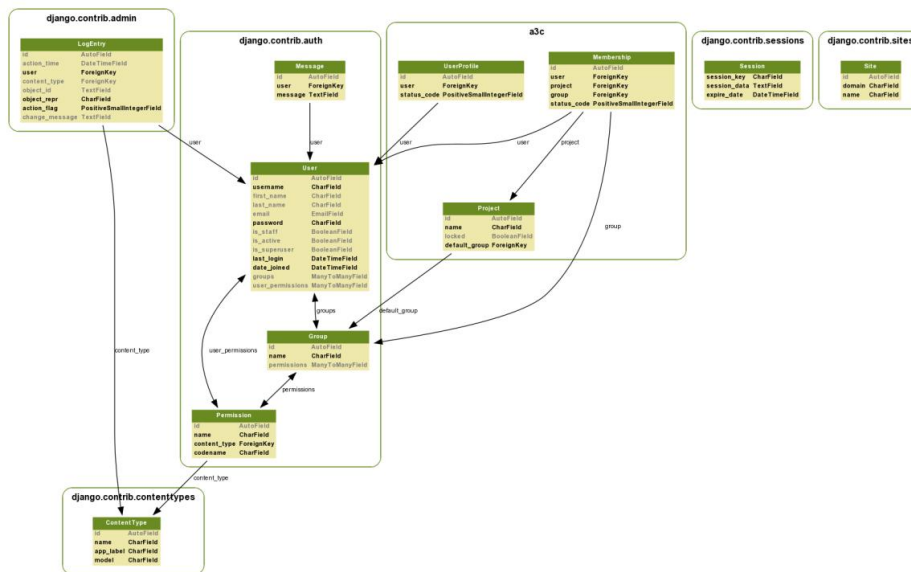
Além do seu “core”, o Django disponibiliza uma série de recursos que os Pythonistas costumam chamar de “baterias inclusas”.



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.

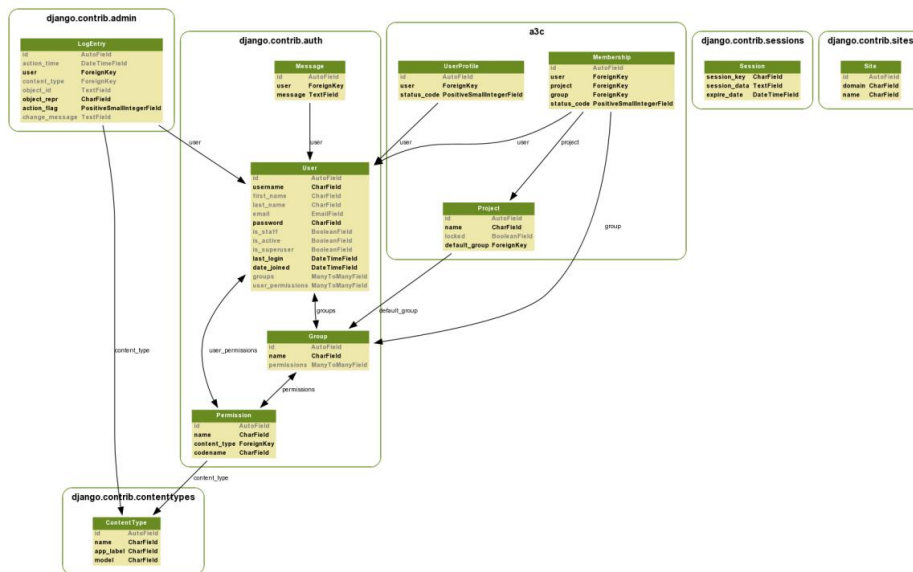
<https://www.profissionaisti.com.br/entendendo-o-django/>



Projetos, Aplicações e “Plugabilidade”

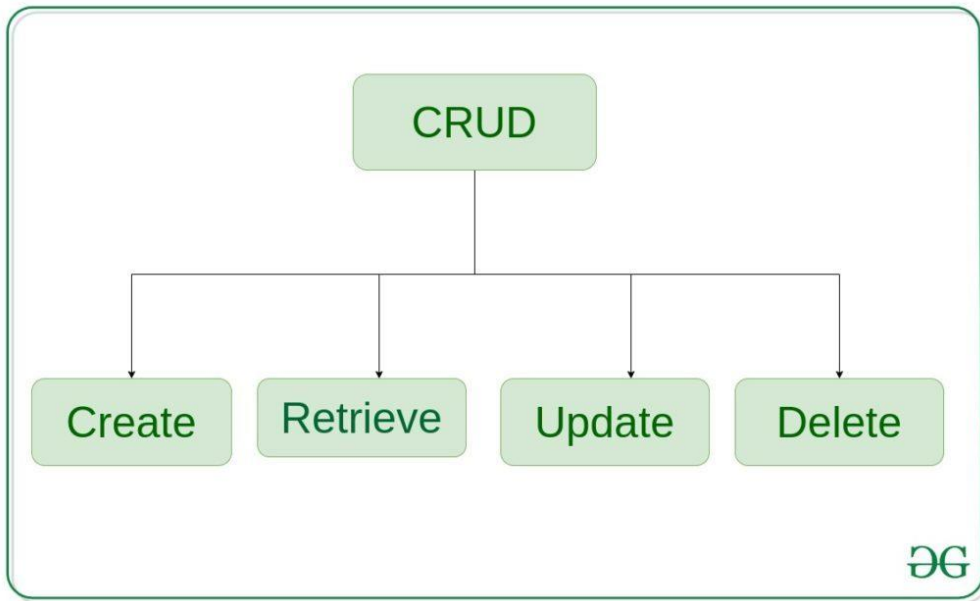
Dentre os recursos mais comuns, temos à disposição o auth, sites, admin, contenttypes, etc. Sem eles, o seu website funciona. Com eles, você economiza quilômetros de LOC e terá aplicações testadas à exaustão.

<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:

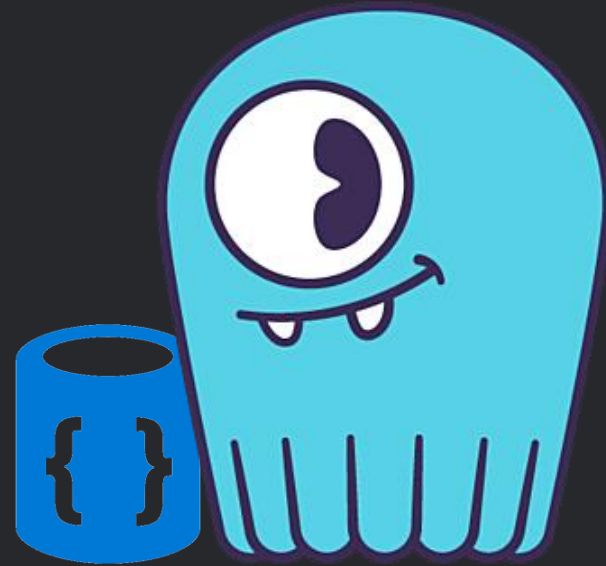
[UpdateView](#) - atualizar ou editar entradas existentes em uma tabela no banco de dados

[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

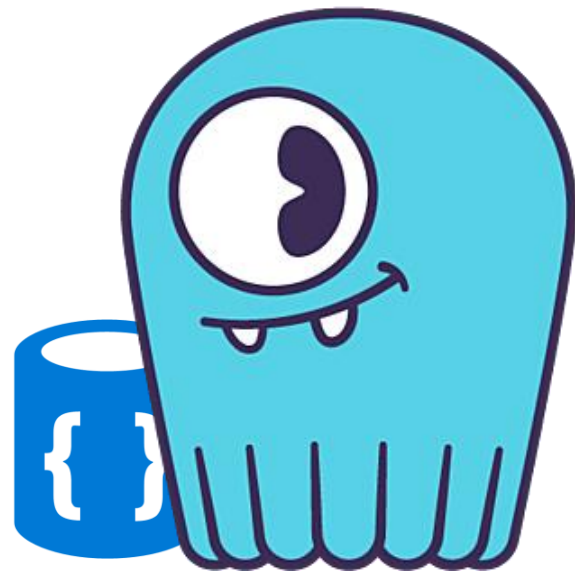


Primeiros Passos com o *Framework Flask*

Prof. Dr. Diego Bruno

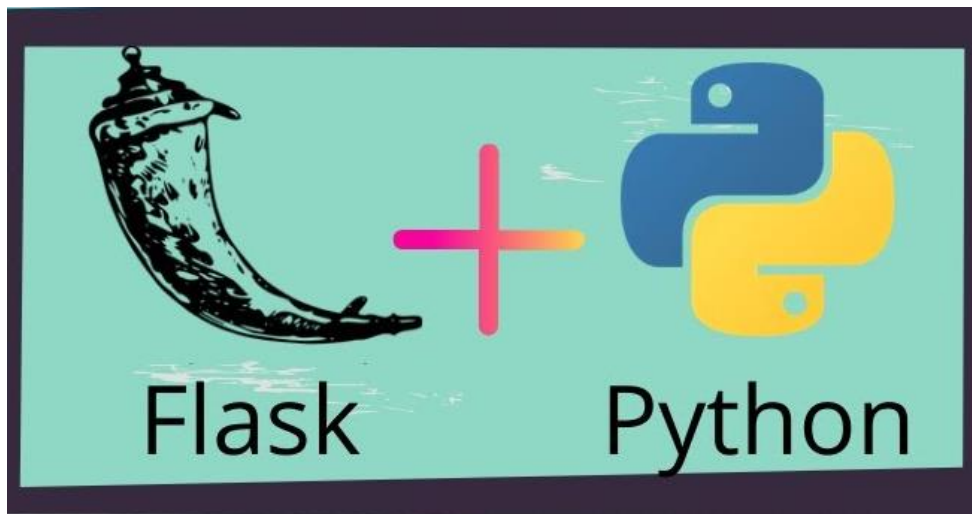
Education Tech Lead na DIO

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



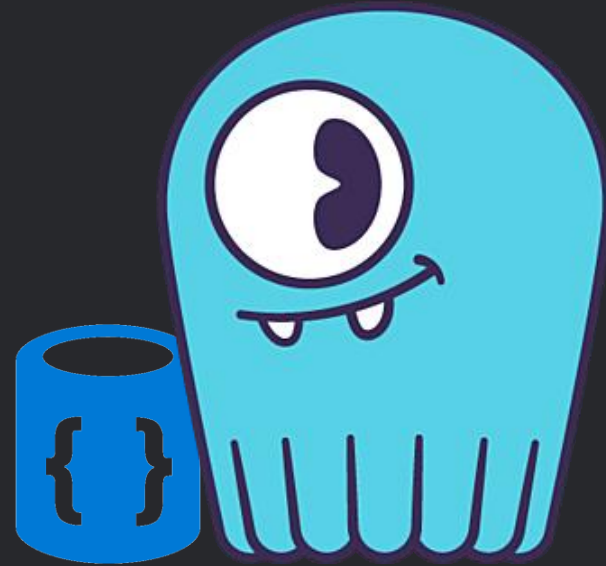
Roteiro para nossa aula de hoje

- Introdução
- Objetivos
- Base teórica
- Motivação
- Conclusões



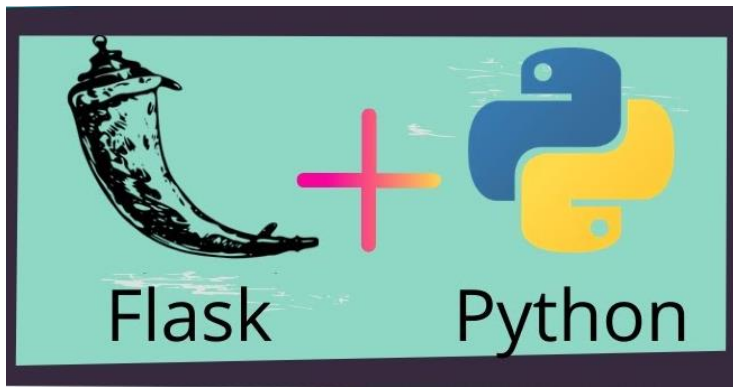
Flask

Prof. Dr. Diego Bruno



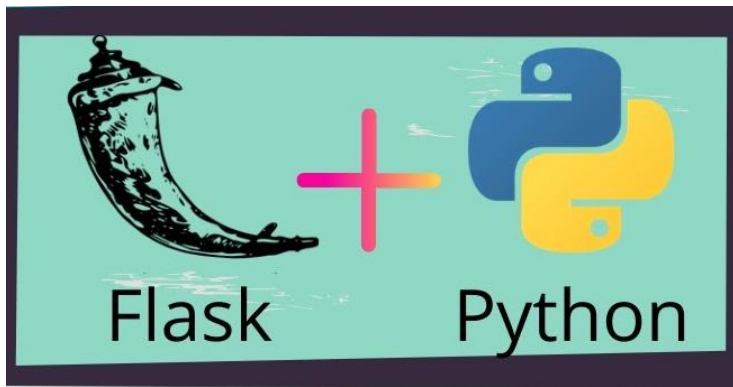
Mas o que é Flask?

Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível.



Mas o que é Flask?

O **Flask Python** é basicamente um framework do tipo “faça você mesmo”. Isso significa que não tem nenhuma interação interna com banco de dados, mas o pacote flask-sqlalchemy irá conectar o banco de dados SQL a um aplicativo Flask. O pacote flask-sqlalchemy precisa somente de uma coisa para se conectar o banco de dados SQL: o banco de dados URL.



Mas o que é Flask?

O Flask precisa que o banco de dados URL seja parte da sua configuração central através do atributo **SQLALCHEMY_DATABASE_URI**. Uma solução rápida e suja pra isso é fazer um hardcode do banco de dados dentro do aplicativo.

```
1 # top of app.py
2 from flask import Flask
3 from flask_sqlalchemy import SQLAlchemy
4
5 app = Flask(__name__)
6 app.config['SQLALCHEMY_DATABASE_URI'] = 'postgres://localhost:5432/flask_todo'
7 db = SQLAlchemy(app)
```


Simplicidade

Você pode tornar as coisas mais simples utilizando variáveis de ambiente. Elas irão garantir que independente da máquina que você rode o código, ele irá sempre apontar na coisa certa, se essa coisa estiver configurada no ambiente que está sendo rodado. Ele também garante que, mesmo que você precise da informação para rodar o aplicativo, nunca irá aparecer como um valor ***hardcoded*** no ***source control***.

Simplicidade

No mesmo lugar que você declarou o `FLASK_APP`, declare um `DATABASE_URL` apontando para o lugar do seu banco de dados Postgres. O desenvolvimento tende a funcionar localmente, então aponte para o seu banco de dados local.

```
1 # Também no seu script active
2
3 export DATABASE_URL='postgres://localhost:5432/flask_todo'
```

Agora em **app.py** inclua o banco de dados URL no seu aplicativo web.

Python

```
1 app.config['SQLALCHEMY_DATABASE_URI'] =
2 os.environ.get('DATABASE_URL', '')
db = SQLAlchemy(app)
```

Simplicidade

No mesmo lugar que você declarou o `FLASK_APP`, declare um `DATABASE_URL` apontando para o lugar do seu banco de dados Postgres. O desenvolvimento tende a funcionar localmente, então aponte para o seu banco de dados local.

```
1 # Também no seu script active
2
3 export DATABASE_URL='postgres://localhost:5432/flask_todo'
```

Agora em **app.py** inclua o banco de dados URL no seu aplicativo web.

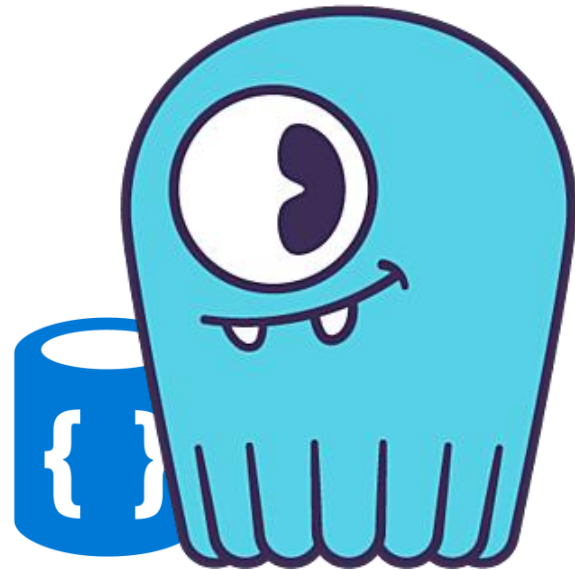
Python

```
1 app.config['SQLALCHEMY_DATABASE_URI'] =
2 os.environ.get('DATABASE_URL', '')
db = SQLAlchemy(app)
```

Simplicidade

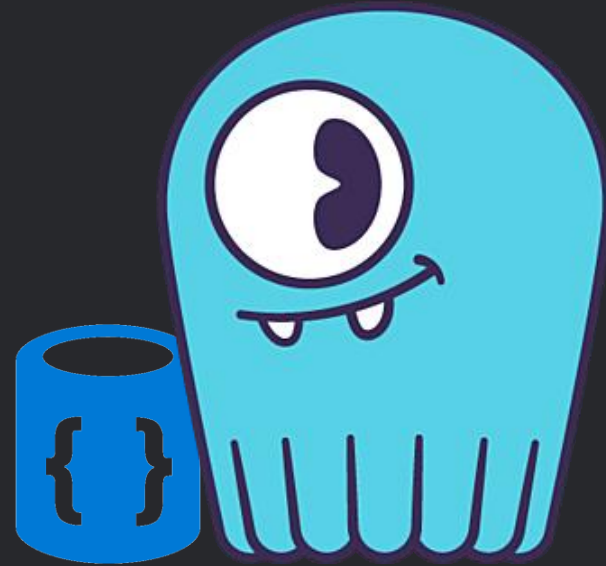
E simplesmente dessa forma, o seu aplicativo agora tem uma conexão com um banco de dados.

```
1 app.config['SQLALCHEMY_DATABASE_URI'] =  
2 os.environ.get('DATABASE_URL', "")  
db = SQLAlchemy(app)
```



Obrigado!

Prof. Dr. Diego Bruno

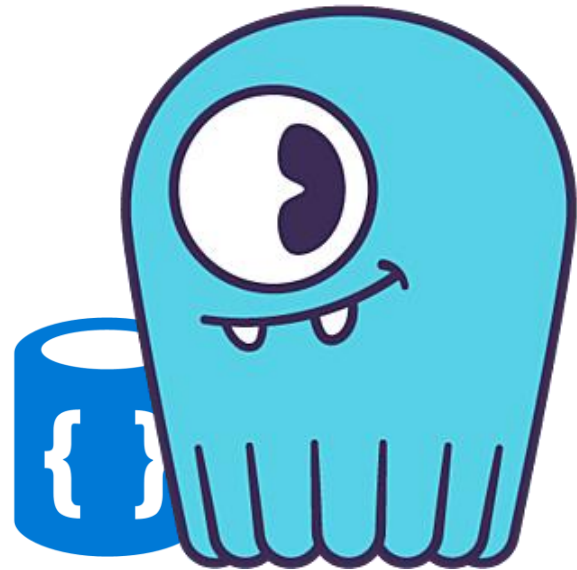


Aplicações Framework *Flask*

Prof. Dr. Diego Bruno

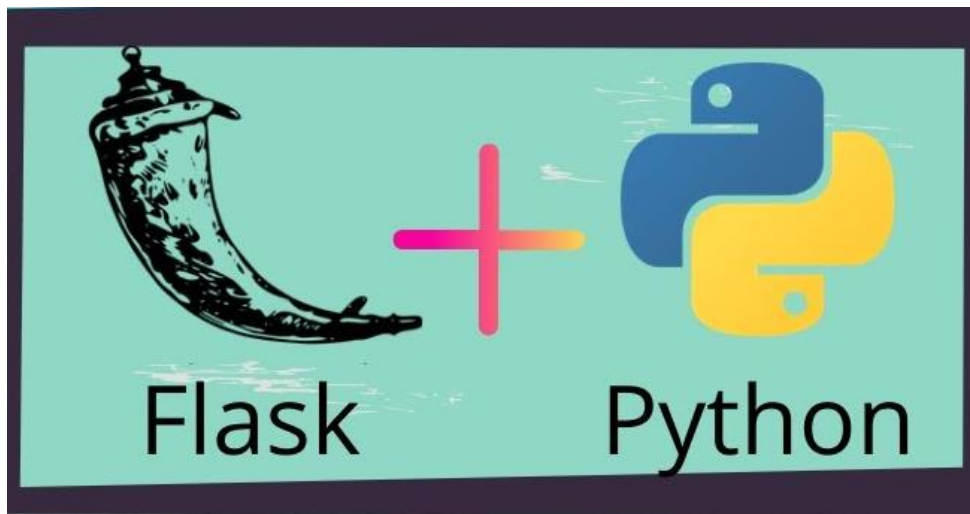
Education Tech Lead na DIO

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



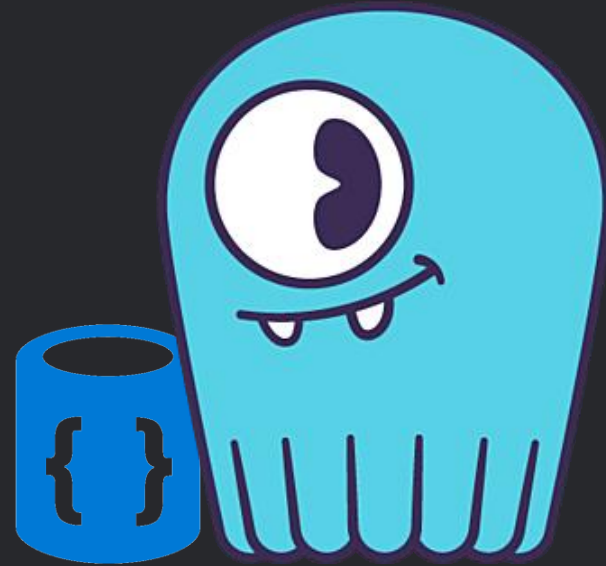
Roteiro para nossa aula de hoje

- Introdução
- Objetivos
- Base teórica
- Motivação
- Conclusões



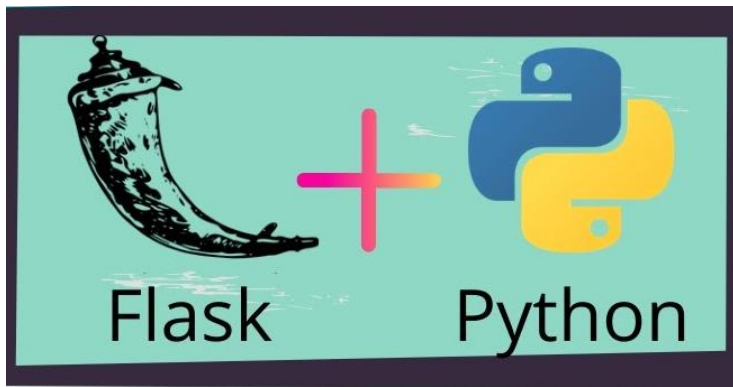
Flask

Prof. Dr. Diego Bruno



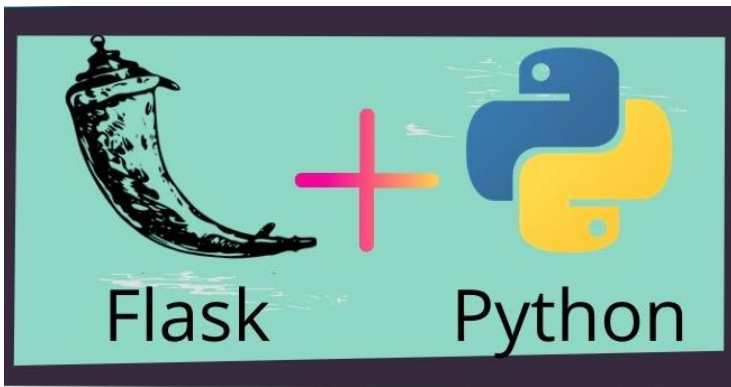
Mas o que é Flask?

Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível.



Mas o que é Flask?

O **Flask Python** é basicamente um framework do tipo “faça você mesmo”. Isso significa que não tem nenhuma interação interna com banco de dados, mas o pacote flask-sqlalchemy irá conectar o banco de dados SQL a um aplicativo Flask. O pacote flask-sqlalchemy precisa somente de uma coisa para se conectar o banco de dados SQL: o banco de dados URL.



Mas o que é Flask?

O Flask precisa que o banco de dados URL seja parte da sua configuração central através do atributo **SQLALCHEMY_DATABASE_URI**. Uma solução rápida e suja pra isso é fazer um hardcode do banco de dados dentro do aplicativo.

```
1 # top of app.py
2 from flask import Flask
3 from flask_sqlalchemy import SQLAlchemy
4
5 app = Flask(__name__)
6 app.config['SQLALCHEMY_DATABASE_URI'] = 'postgres://localhost:5432/flask_todo'
7 db = SQLAlchemy(app)
```

Simplicidade

No mesmo lugar que você declarou o `FLASK_APP`, declare um `DATABASE_URL` apontando para o lugar do seu banco de dados Postgres. O desenvolvimento tende a funcionar localmente, então aponte para o seu banco de dados local.

```
1 # Também no seu script active
2
3 export DATABASE_URL='postgres://localhost:5432/flask_todo'
```

Agora em **app.py** inclua o banco de dados URL no seu aplicativo web.

Python

```
1 app.config['SQLALCHEMY_DATABASE_URI'] =
2 os.environ.get('DATABASE_URL', '')
db = SQLAlchemy(app)
```

Simplicidade

No mesmo lugar que você declarou o `FLASK_APP`, declare um `DATABASE_URL` apontando para o lugar do seu banco de dados Postgres. O desenvolvimento tende a funcionar localmente, então aponte para o seu banco de dados local.

```
1 # Também no seu script active
2
3 export DATABASE_URL='postgres://localhost:5432/flask_todo'
```

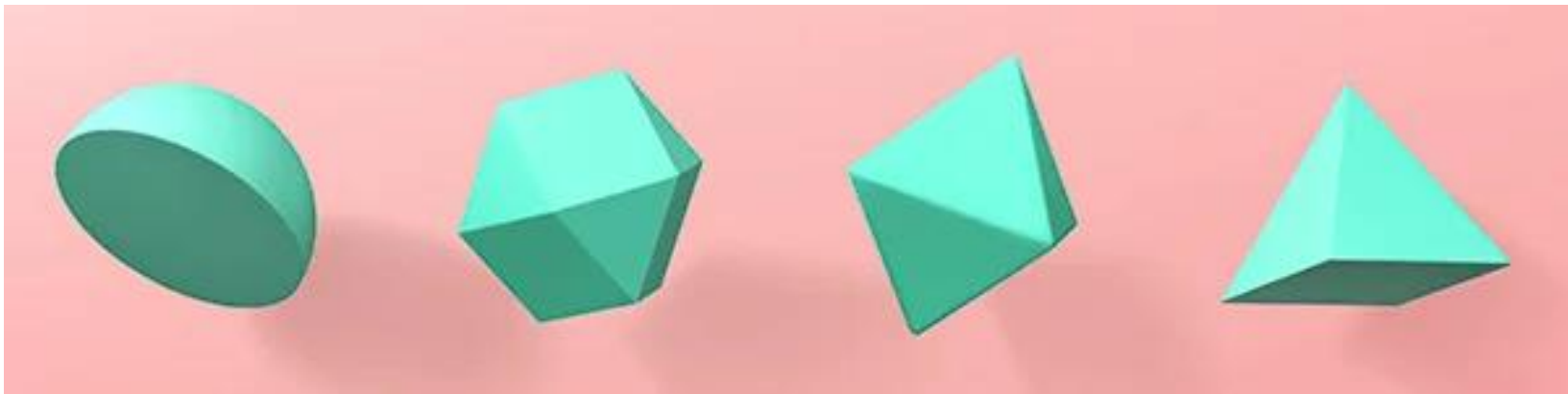
Agora em **app.py** inclua o banco de dados URL no seu aplicativo web.

Python

```
1 app.config['SQLALCHEMY_DATABASE_URI'] =
2 os.environ.get('DATABASE_URL', '')
db = SQLAlchemy(app)
```

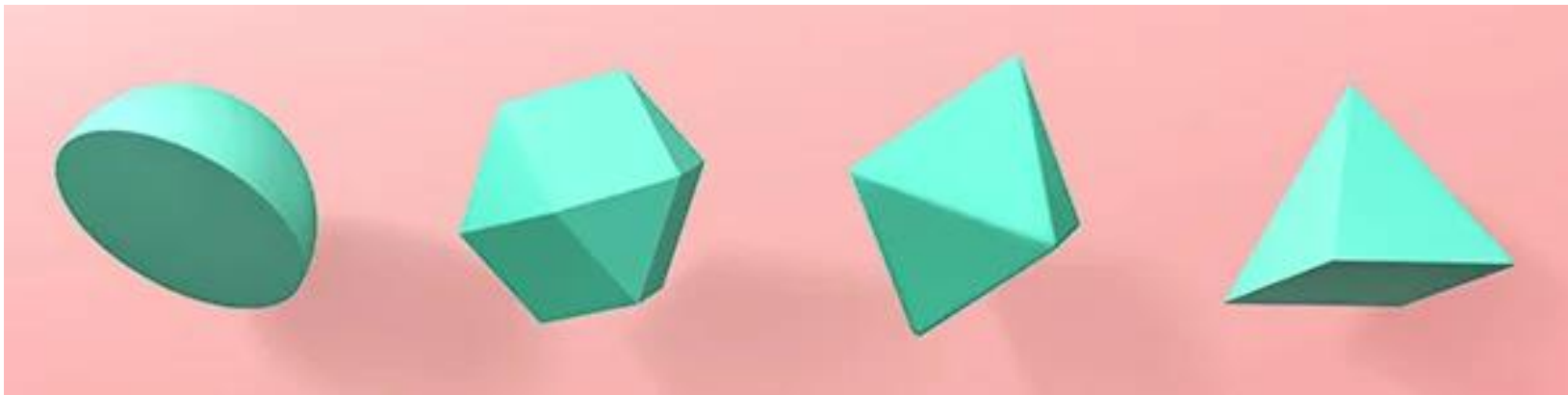
Definindo Objetos em Flask

Ter um banco de dados para estar conectado é um ótimo primeiro passo. Agora é hora de definir alguns objetos para preencher o banco de dados..



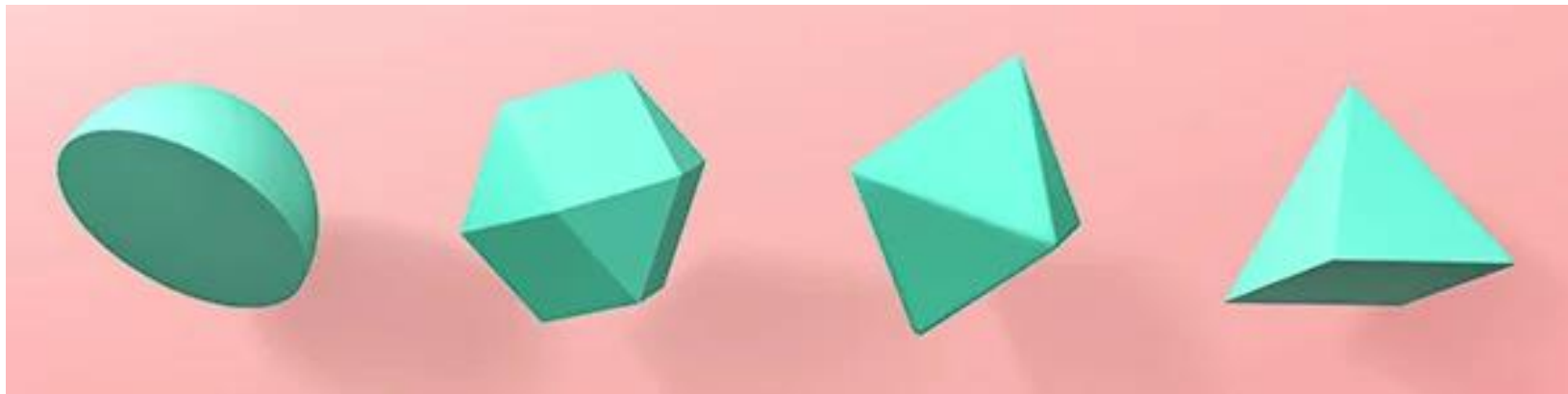
Definindo Objetos em Flask

No desenvolvimento de aplicativos, um “model” refere-se a representação real ou conceitual de algum objeto. Por exemplo, caso esteja fazendo um app para uma concessionária de carros, você talvez defina um model chamado **car** que encapsule todos os atributos e comportamentos de um carro.



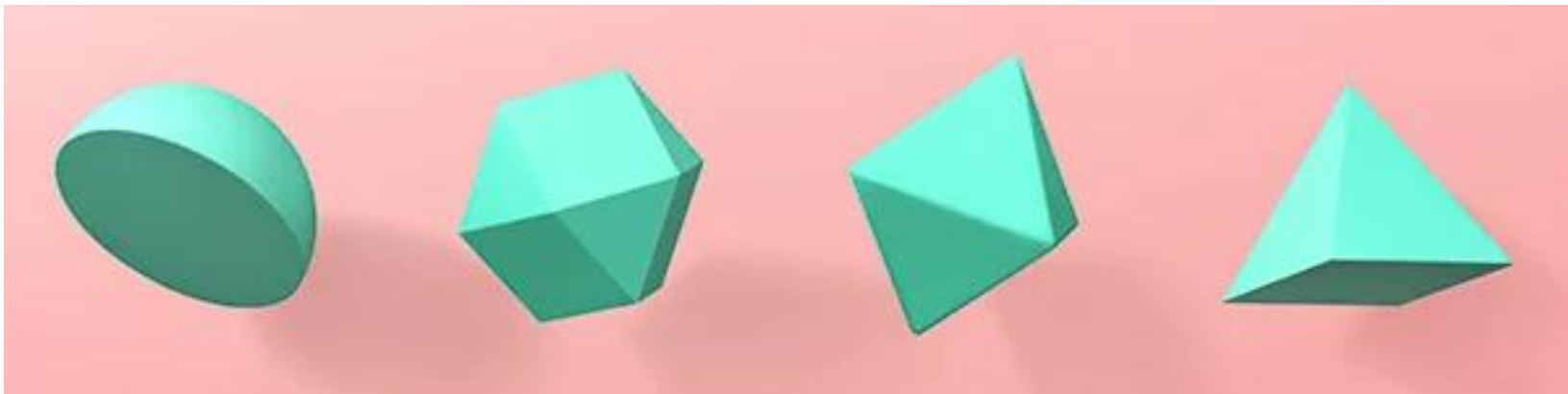
Definindo Objetos em Flask

Nesse caso, você estará fazendo um To-do List com Tasks, e cada Task pertence a um usuário. Antes de você pensar mais a fundo sobre como eles estão relacionados, comece definindo os objetos para Tasks e Users.



Definindo Objetos em Flask

O pacote **flask-sqlalchemy** aproveita o [SQLAlchemy](#) para configurar e informar a estrutura do banco de dados. Você irá definir um modelo que irá viver no banco de dados herdando do objeto **db.Model** e irá definir o atributo desses models como instâncias **db.Column**. Para cada coluna, você deve especificar um tipo de dado, e então você irá passar esse data type para o comando **db.Column** como primeiro argumento..



Fonte: <https://harve.com.br/blog/programacao-python-blog/introducao-e-tutorial-ao-flask-python/>

Definindo Objetos em Flask

Pelo fato da definição do model ocupar um espaço conceitual diferente do que da configuração do aplicativo, faça com que o **models.py** mantenha definições de model de forma separada do **app.py**. O modelo do Task deve ser construído para que tenha os seguintes atributos:

- **id**: Um valor que é um identificador único para puxar do banco de dados.
- **name**: O nome ou o título da task que o usuário irá ver quando task for listada.
- **note**: Quaisquer comentários adicionais que uma pessoa queira deixar com sua task.
- **creation_date**: A data e o horário que a task foi criada.
- **due_date**: A data e o horário que a task deve ser concluída (se houver).
- **completed**: Uma forma de indicar se a task foi concluída ou não.

Definindo Objetos em Flask

Dado essa lista de atributos para objetos Task, o objeto **Task** do aplicativo pode ser definida dessa forma:

```
1 from .app import db
2 from datetime import datetime
3
4 class Task(db.Model):
5     """Tasks for the To Do list."""
6     id = db.Column(db.Integer, primary_key=True)
7     name = db.Column(db.Unicode, nullable=False)
8     note = db.Column(db.Unicode)
9     creation_date = db.Column(db.DateTime, nullable=False)
10    due_date = db.Column(db.DateTime)
11    completed = db.Column(db.Boolean, default=False)
12
13    def __init__(self, *args, **kwargs):
14        """On construction, set date of creation."""
15        super().__init__(*args, **kwargs)
16        self.creation_date = datetime.now()
```

Fonte: <https://harve.com.br/blog/programacao-python-blog/introducao-e-tutorial-ao-flask-python/>

Relacionamento dos models

Em certos aplicativos web, você talvez queira expressar relacionamentos entre objetos. No exemplo do To-do list, usuários são donos de várias tasks, e cada tarefa pertence a somente a um usuário.

Esse é um exemplo de um relacionamento “many-to-one”, também conhecido como foreign key relationship, onde as tasks são os “many” (muitos) e o usuário dono delas é o “one” (um).

Relacionamento dos models

No Flask Python, um relacionamento many-to-one pode ser especificado utilizando a função **db.relationship**. Primeiro, construa o objeto User.

```
1 class User(db.Model):
2     """The User object that owns tasks."""
3     id = db.Column(db.Integer, primary_key=True)
4     username = db.Column(db.Unicode, nullable=False)
5     email = db.Column(db.Unicode, nullable=False)
6     password = db.Column(db.Unicode, nullable=False)
7     date_joined = db.Column(db.DateTime, nullable=False)
8     token = db.Column(db.Unicode, nullable=False)
9
10    def __init__(self, *args, **kwargs):
11        """On construction, set date of creation."""
12        super().__init__(*args, **kwargs)
13        self.date_joined = datetime.now()
14        self.token = secrets.token_urlsafe(64)
```

Inicializando o banco de dados

Agora que os modelos e os relacionamentos de modelos estão configurados, comece configurando o seu banco de dados. O Flask Python não vem com a sua própria utilidade de gerenciamento de banco de dados, então você terá que escrever o seu próprio (até um certo ponto).

Inicializando o banco de dados

Crie um script chamado `initializedb.py` próximo do `setup.py` para gerenciar o banco de dados. (Claro, não precisa ser chamado assim, mas porque não dar nomes que são apropriados a função do arquivo?) Dentro de `initializedb.py`, importe o objeto `db` de `app.py` e use-o para criar e eliminar tabelas. `initializedb.py` deve parecer dessa forma:

```
1 from todo.app import db
2 import os
3
4 if bool(os.environ.get('DEBUG', '')):
5     db.drop_all()
6     db.create_all()
```

Views e Configuração da URL



As últimas partes necessárias para conectar o aplicativo inteiro são os views e routes. Em desenvolvimento web, um “view” (conceito) é uma funcionalidade que roda quando um ponto de acesso específico (“route”) é atingido.

Em Flask, views aparecem como funções; por exemplo, consegue ver a view “hello world” ali em cima. Pra ser prático, vamos mostrar aqui novamente:

Quando o route do **http://domainname/** é acessado, o cliente recebe a resposta, “Hello, world!”.

```
1 @app.route('/')
2 def hello_world():
3     """Print 'Hello, world!' as the response body."""
4     return 'Hello, world!'
```


Views e Configuração da URL



Comece por um view que lida somente com solicitações **get**, e responda com o JSON representando todas os routes que serão acessíveis e os métodos que podem ser utilizados para acessar elas:

```
1 from flask import jsonify
2
3 @app.route('/api/v1', methods=["GET"])
4 def info_view():
5     """List of routes for this API."""
6     output = {
7         'info': 'GET /api/v1',
8         'register': 'POST /api/v1/accounts',
9         'single profile detail': 'GET /api/v1/accounts/<username>',
10        'edit profile': 'PUT /api/v1/accounts/<username>',
11        'delete profile': 'DELETE /api/v1/accounts/<username>',
12        'login': 'POST /api/v1/accounts/login',
13        'logout': 'GET /api/v1/accounts/logout',
14        "user's tasks": 'GET /api/v1/accounts/<username>/tasks',
15        "create task": 'POST /api/v1/accounts/<username>/tasks',
16        "task detail": 'GET /api/v1/accounts/<username>/tasks/<id>',
17        "task update": 'PUT /api/v1/accounts/<username>/tasks/<id>',
18        "delete task": 'DELETE /api/v1/accounts/<username>/tasks/<id>'
19    }
20    return jsonify(output)
```

Views e Configuração da URL

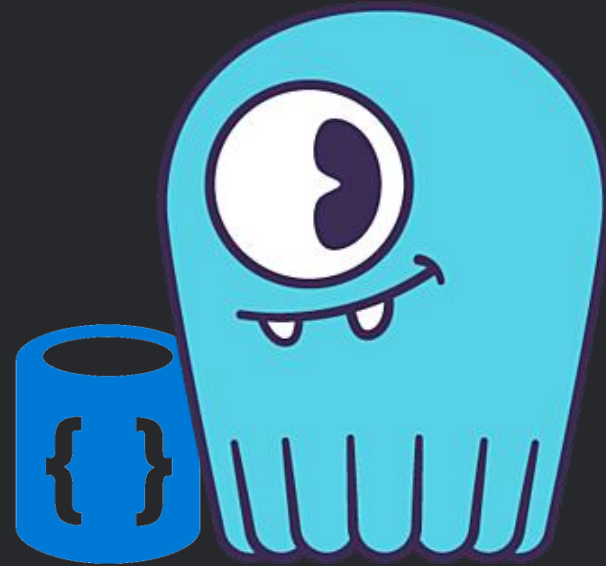


A função view acima pega o que é efetivamente uma lista de todo o route que esse API pretende lidar e envia ao cliente todas as vezes que o route `http://domainname/api/v1` for acessada. Tenha em mente que, por si mesmo, o Flask oferece suporte ao roteamento para URLs exatamente iguais, então acessar essa rota com um trailing `/` iria criar um erro 404. Se você quisesse lidar com a mesma função view, você precisaria de um stack de decoradores dessa forma:

```
1 @app.route('/api/v1', methods=["GET"])
2 @app.route('/api/v1/', methods=["GET"])
3 def info_view():
4     # blah blah blah more code
```

Obrigado!

Prof. Dr. Diego Bruno



Boas Prática com Python

Formação Python Developer

Juliana Mascarenhas

Tech Education Specialist DIO / Owner @Simplificandoredes e @SimplificandoProgramação

Mestre em modelagem computacional | Cientista de dados

@in/juliana-mascarenhas-ds/



<https://github.com/julianazanelatto>

Juliana Mascarenhas

Tech Education Specialist

@SimplificandoRedes

@SimplificandoProgramação

Cientista de dados

Desenvolvedora Java/Python

Me Modelagem Computacional - LNCC

Objetivo Geral

Convenções de nomeação

Indentação

DocStrings

Overview sobre as boas práticas de programação
em Python com o guia PEP 8.

Espaços em branco

Espaços em branco

Code Layout

Etapa 1

O que é a PEP 8? Qual sua importância?

// Integração com Python



PEP

- Julho 2000
- PEP – Python Enhancement Proposals
- Convenções de acordo com tópicos
- Possui controle de versão

[Documentação PEP](#)

[Documentação PEP 8](#)

PEP 8 – Guia de Estilo

Objetivo:

- Nomeação de classes e métodos
- Code Layout
- Indentação
- Docstrings - comentários
- ...



“Readability counts.”
— *The Zen of Python*

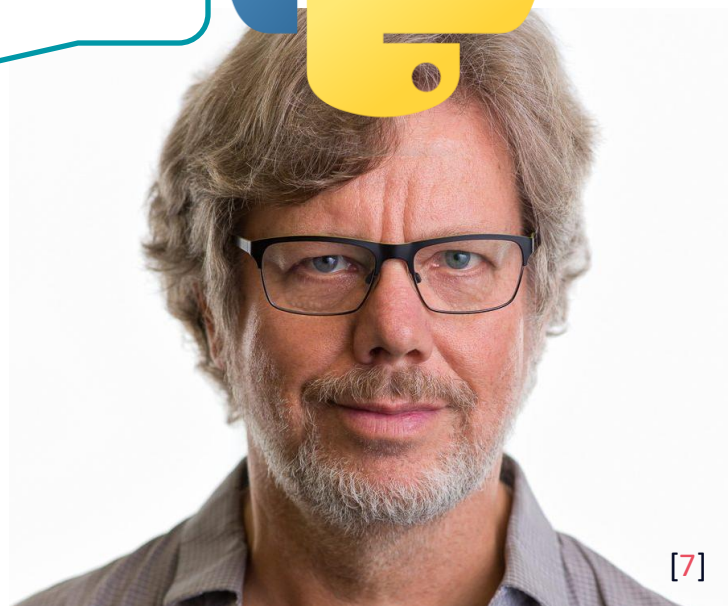
PEP 8 – Legibilidade

O que eu fiz aqui mesmo?



“Code is read much more often than it is written.”

Guido van Rossum



PEP 8 – Legibilidade

- Time de desenvolvimento
- Trabalho em equipe
- Facilidade de entendimento
- Manutenção facilitada



Etapa 2

The Guide Line – Padronização com Python

// Integração com Python

PEP 8 - Nomeando

- Classes
- Variáveis
- Métodos
- Funções

Nomes com significado

```
# two spaces before a python class
```

```
class ClasseComposta:  
    def __init__(self):  
        pass  
    pass
```

```
# imports no TOP
```

```
class Classe:  
    pass
```

```
# functions
```

```
def soma():  
    pass
```

“Explicit is better than implicit.”

— *The Zen of Python*

PEP 8 - Nomeando

Type	Naming Convention	Examples
Function	Use a lowercase word or words. Separate words by underscores to improve readability.	<code>function, my_function</code>
Variable	Use a lowercase single letter, word, or words. Separate words with underscores to improve readability.	<code>x, var, my_variable</code>
Class	Start each word with a capital letter. Do not separate words with underscores. This style is called camel case or pascal case .	<code>Model, MyClass</code>

PEP 8 - Nomeando

Method	Use a lowercase word or words. Separate words with underscores to improve readability.	<code>class_method</code> , <code>method</code>
Constant	Use an uppercase single letter, word, or words. Separate words with underscores to improve readability.	<code>CONSTANT</code> , <code>MY_CONSTANT</code> , <code>MY_LONG_CONSTANT</code>
Module	Use a short, lowercase word or words. Separate words with underscores to improve readability.	<code>module.py</code> , <code>my_module.py</code>
Package	Use a short, lowercase word or words. Do not separate words with underscores.	<code>package</code> , <code>mypackage</code>

PEP 8 - Nomeando

```
>>> # Not recommended
>>> x = 'John Smith'
>>> y, z = x.split()
>>> print(z, y, sep=',') 'Smith, John'
```



```
>>> # Recommended
>>> name = 'John Smith'
>>> first_name, last_name = name.split()
>>> print(last_name, first_name, sep=',') 'Smith, John'
```


PEP 8 - Nomeando

```
# Not recommended  
def db(x):  
    return x * 2
```



```
# Recommended  
def multiply_by_two(x):  
    return x * 2
```



Etapa 2

The Guide Line – Code Layout & Tópicos Relacionados

// Integração com Python

Code Layout

O que é isso?



```
class Classe:
    pass
# functions
def soma():
    pass
# two spaces before a python class or method
class ClasseComposta:
    def __init__(self):
        inicio = 'Bem-vindo'
        return self.inicio
    pass
def function_compost():
    pass
def function_sum(number_one, number_two):
    total = number_one + number_two
    return total
```

```
return total
total = number_one + number_two
def function_sum(number_one, number_two):
    pass
```

“Beautiful is better than ugly.”

— *The Zen of Python*

PEP 8 – Code Layout

```
def function_compost():  
    pass  
  
def function_sum(number_one, number_two):  
    total = number_one + number_two  
    return total
```

two spaces before a python class or method

```
class ClasseComposta:  
    def __init__(self):  
        inicio = 'Bem-vindo'  
        return self.inicio  
    pass
```

```
class Classe:  
    pass  
  
# functions  
  
def soma():  
    pass
```

“Beautiful is better than ugly.”
— *The Zen of Python*

PEP 8 – Code Layout

- Blank lines
- Espaçamento
- Tamanho máximo
- Quebra de linha

“Beautiful is better than ugly.”
— *The Zen of Python*

PEP 8 - Indentação

Define o agrupamento das linhas de código e a lógica da aplicação

```
x = 3
if x > 5:
    print('x is larger than 5')
```

“There should be one—and preferably only one—obvious way to do it.”

— *The Zen of Python*

PEP 8 – Tab e Espaço

- Esqueça a tecla Tab
- 4 espaços

Pode misturar?

Tecla tab



4 espaços

```
code.py: inconsistent use of tabs and spaces in indentation
```


PEP 8 – Tab e Espaço

Limite = 79 caracteres

- Quebra estruturas e métodos

```
def function( arg_one, arg_two,  
             arg_three, arg_four):  
    return arg_one
```

```
def function(  
    arg_one,  
    arg_two,  
    arg_three,  
    arg_four):  
    return arg_one
```



PEP 8 – Testando o Código

Shell

```
$ python2 -t code.py
code.py: inconsistent use of tabs and spaces in indentation
```



Shell

```
$ python2 -tt code.py
File "code.py", line 3
    print(i, j)
          ^
TabError: inconsistent use of tabs and spaces in indentation
```



PEP 8 – Quebrando linhas

```
list_of_numbers = [  
    1, 2, 3,  
    4, 5, 6,  
    7, 8, 9  
]
```

```
list_of_numbers = [  
    1, 2, 3,  
    4, 5, 6,  
    7, 8, 9  
]
```



PEP 8 – Quebrando linhas

```
# Correct:  
import os  
import sys
```

```
# Wrong:  
import sys, os
```

It's okay to say this though:

```
# Correct:  
from subprocess import Popen, PIPE
```



Etapa 1

The Guide Layout - Comentários no código

// Integração com Python

Comentários no código



- Comentários tanto quanto necessário
- Objetivo: facilitar o entendimento
- Código bem escrito também é documentação

“If the implementation is hard to explain, it’s a bad idea.”

— *The Zen of Python*

Comentários no código



- Limite da linha = 79 caracteres
- Sentenças completas iniciando com letra maiúscula
- Atualização de comentários

“If the implementation is hard to explain, it’s a bad idea.”

— *The Zen of Python*

Blocos de comentários



Python

```
for i in range(0, 10):  
    # Loop over i ten times and print out the value of i, followed by a  
    # new line character  
    print(i, '\n')
```

“If the implementation is hard to explain, it’s
a bad idea.”

— *The Zen of Python*

Blocos de comentários



Python

```
empty_list = [] # Initialize empty list
```

```
x = 5
```

```
x = x * 5 # Multiply x by 5
```

Não explique o óbvio!

Docstring

```
"""
    Este é um exemplo de docstring

    Exemplificação sobre nomeação de recursos com python.
    tais recursos são: variáveis, classes, funções e espaçamento

    Editor do pycharm pode "reclamar" das palavras em português. Isso acarreta em avisos
    (linhas verdes) sobre as nomeações

    Os métodos e classes aqui podem não ser utilizados. São apenas para exemplificação da PEP 8
"""
```

- Caracteres: """ ou '''
- Dentro e classes, funções e início de programas
- Escritos para módulos públicos

Etapa 1

The Guide Line - Espaços em Branco em Expressões

// Integração com Python

Espaçamento

```
# Correct:  
spam(ham[1], {eggs: 2})
```

```
# Correct:  
if x == 4: print(x, y); x, y = y, x
```

```
# Wrong:  
if x == 4 : print(x , y) ; x , y = y , x
```

Não exagere nos espaços!

```
# Correct:  
foo = (0,)
```

```
# Wrong:  
bar = (0, )
```

Python

```
# Recommended  
if x>5 and x%2==0:  
    print('x is larger than 5 and divisible by 2!')
```

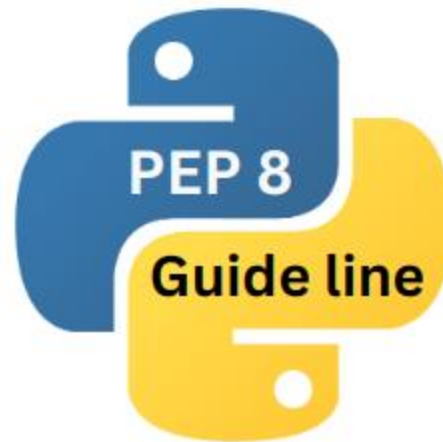
Quando evitar?

Poucos espaços

- Difícil de ler

Esparso demais

- Muitos espaços



Dúvidas?

- > Fórum/Artigos
- > Comunidade Online (Discord)



Para saber mais

Referências principais:

- <https://peps.python.org/pep-0008/>
- <https://peps.python.org/pep-0257/>
- <https://pypi.org/project/pylint/>
- <https://pypi.org/project/flake8/>

<https://github.com/julianazanelatto>

