

Machine Learning Engineer

# Python para ML e IA

Desenvolvimento de API com Flask II

Leonardo Pena

# / Seja muito bem vindo



## DEFINIÇÕES

Iniciar configuração de Flask com banco de dados e JWT



## CONTEXTO

Vamos apresentar a classe Config e criar nosso banco (SQLite)



## DINÂMICA

Dividido em 3 partes



## ESTRUTURAÇÃO

Foco no setup inicial (ORM + Models) antes das rotas

# Objetivo dessa primeira parte



Passo 1

Mostrar  
classe Config  
com  
informações  
de SWAGGER,  
SECRET\_KEY  
e DB



Passo 2

Instanciar  
Flask e aplicar  
configurações  
(app.config.from\_object(Config))



Passo 3

Integrar  
SQLAlchemy  
ao projeto,  
explicando  
vantagens  
do ORM



Passo 4

Criar  
modelos  
User e  
Recipe com  
atributos  
básicos



Passo 5

Testar a  
criação de  
tabelas  
usando  
db.create\_all()  
)



# Configurações iniciais



# / API de Receitas Gourmet

- Vamos montar uma API de receitas!
- Comece criando o ambiente virtual (agora você já sabe, da aula 02)
- Instale, via terminal, o flask também



## / A classe Config

- Até então criamos um app assim:

```
app.py > ...  
1  from flask import Flask  
2  
3  app = Flask(__name__)  
4
```

E logo já iniciamos as criações de rotas

```
8  @app.route('/')  
9  def home():  
10     return 'Pagina Inicial'
```



## / A classe Config

- Mas, muitas vezes será necessário configurar o projeto previamente!
- Por exemplo, você pode querer, antes de começar o projeto, configurar:
  - Uma chave secreta pro projeto
  - Mudar o projeto para modo de teste
  - Etc

## / A classe Config

- Para isso, em Flask temos a opção de alterar as configurações via:
  - *app.config*

<https://flask.palletsprojects.com/en/stable/config/>



/

# Vamos ver um exemplo e explicar em seguida

Note que agora, em vez de apenas instanciar a aplicação e depois as rotas, estamos configurando o projeto

```
3  # Configurações da Aplicação
4  class Config:
5      SECRET_KEY = 'sua_chave_secreta'
6      CACHE_TYPE = 'simple'
7      SWAGGER = {
8          'title': 'Catálogo de Receitas Gourmet',
9          'uiversion': 3
10     }
11     SQLALCHEMY_DATABASE_URI = 'sqlite:///recipes.db'
12     SQLALCHEMY_TRACK_MODIFICATIONS = False
13     JWT_SECRET_KEY = 'sua_chave_jwt_secreta'
14
15     app = Flask(__name__)
16
17     app.config.from_object(Config)
18
```

/

# Para testar se as configurações foram criadas

```
app.py > ...
1  from flask import Flask
2
3  # Configurações da Aplicação
4  class Config:
5      SECRET_KEY = 'sua_chave_secreta'
6      CACHE_TYPE = 'simple'
7      SWAGGER = {
8          'title': 'Catálogo de Receitas Gourmet',
9          'uiversion': 3
10     }
11     SQLALCHEMY_DATABASE_URI = 'sqlite:///recipes.db'
12     SQLALCHEMY_TRACK_MODIFICATIONS = False
13     JWT_SECRET_KEY = 'sua_chave_jwt_secreta'
14
15     app = Flask(__name__)
16
17     app.config.from_object(Config)
18
19     print(app.config['SECRET_KEY'])
20     print(app.config['SQLALCHEMY_DATABASE_URI'])
21     print(app.config['SWAGGER'])
22     print(app.config['CACHE_TYPE'])
23
```

/

# Rodando no terminal, vemos

## ✓ TERMINAL

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula3-new % python app.py  
sua_chave_secreta  
sqlite:///recipes.db  
{'title': 'Catálogo de Receitas Gourmet', 'uiversion': 3}  
simple
```

Exatamente as configurações que queríamos!



## / A classe Config

- Fizemos então a criação de uma classe "Config", que possui as configurações dos projetos. Colocamos ela então no *app.config*

<https://flask.palletsprojects.com/en/stable/config/>

## / A classe Config

- Outra opção interessante, seria criar um arquivo .py auxiliar, com as configurações e importar elas. Por exemplo

```
app.py > ...
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  app.config.from_object('config')
6
7  print(app.config['SECRET_KEY'])
8  print(app.config['SQLALCHEMY_DATABASE_URI'])
9  print(app.config['SWAGGER'])
10 print(app.config['CACHE_TYPE'])
11
```

```
config.py > ...
1  SECRET_KEY = 'sua_chave_secreta'
2  CACHE_TYPE = 'simple'
3  SWAGGER = {
4      'title': 'Catálogo de Receitas Gourmet',
5      'uiversion': 3
6  }
7  SQLALCHEMY_DATABASE_URI = 'sqlite:///recipes.db'
8  SQLALCHEMY_TRACK_MODIFICATIONS = False
9  JWT_SECRET_KEY = 'sua_chave_jwt_secreta'
10
```



## / A classe Config

- Mas, o que são essas configurações?  
Como sei as possíveis?
- Há diversas, como as nativas: DEBUG, TESTING, SECRET\_KEY... E outras de extensões também:  
SQLALCHEMY\_DATABASE\_URI,  
JWT\_SECRET\_KEY

Leia a documentação para ver muitas outras opções

<https://flask.palletsprojects.com/en/stable/config/>

<https://flask-sqlalchemy.readthedocs.io/en/stable/config/>



## / A classe Config

- Vamos utilizar por enquanto:
- `SECRET_KEY` e `JWT_SECRET_KEY` para segurança
- `CACHE_TYPE = 'simple'` habilita caching básico
- `SWAGGER` config para título e versão da doc interativa
- `SQLALCHEMY_DATABASE_URI` define qual banco (ex.: `sqlite:///recipes.db`)
- `SQLALCHEMY_TRACK_MODIFICATIONS = False` para evitar warnings

Nessa aula veremos mais sobre JWT, sqlalchemy e cache!



# Banco de dados e SQLAlchemy

# / Introdução ao Banco de Dados e SQLAlchemy

- **O que é um banco de dados?**

- Sistema que armazena e organiza dados de forma estruturada.
- Permite acesso, gerenciamento e manipulação de dados.
- Exemplos: SQLite, MySQL, PostgreSQL, Oracle.

- **O que é SQLAlchemy?**

- Biblioteca em Python para interagir com bancos de dados.
- Abstrai comandos SQL, permitindo uso de objetos Python.
- Suporta múltiplos bancos de dados como SQLite, PostgreSQL e MySQL.

# / ORM - Mapeamento Objeto-Relacional

- **O que é ORM?**

- Mapeamento entre tabelas do banco e objetos/classe no código.
- Facilita o uso de banco de dados sem comandos SQL diretos.
- Garante consistência e facilita a manutenção do código.

**SQLAlchemy é um ORM (Object-Relational Mapper)**



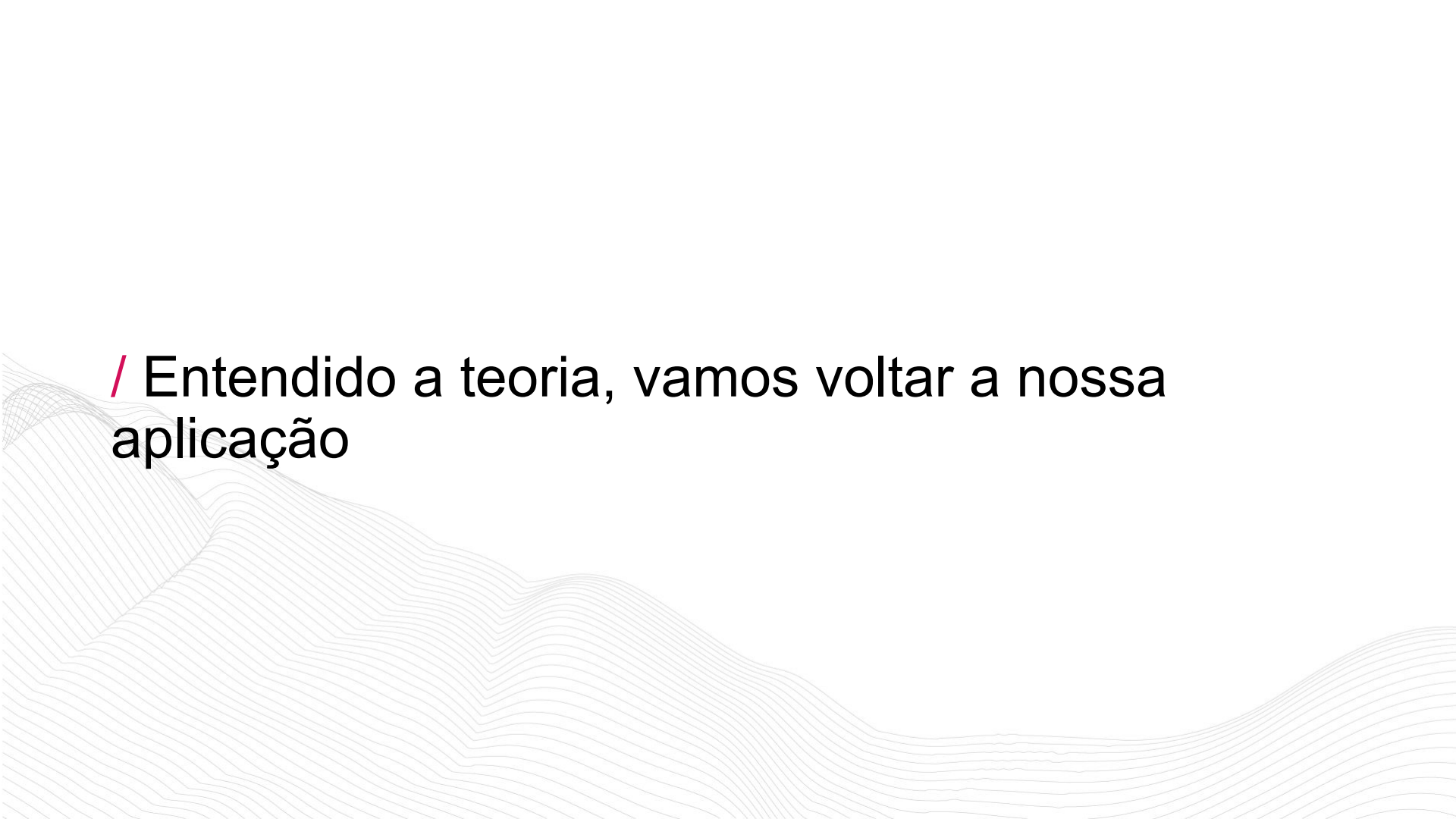
# / Modelagem de Dados no SQLAlchemy

- **Como criar Modelos?**

- Cada modelo representa uma tabela no banco.
- Definido como uma classe que herda de *db.Model*.
- Colunas definidas como atributos de classe.

## Exemplo

```
class User(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    username = db.Column(db.String(80), unique=True, nullable=False)  
    password = db.Column(db.String(120), nullable=False)
```



/ Entendido a teoria, vamos voltar a nossa aplicação

# / API de Receitas Gourmet

- Via pip install, instale o *flask-sqlalchemy*

```
pip install flask-sqlalchemy
```

- Importe no código

```
app.py > ...  
1  from flask import Flask  
2  from flask_sqlalchemy import SQLAlchemy  
3
```

# / API de Receitas Gourmet

- Será necessário inicializar o SQLAlchemy na aplicação flask

```
app.py > ...  
1  from flask import Flask  
2  from flask_sqlalchemy import SQLAlchemy  
3  
4  app = Flask(__name__)  
5  app.config.from_object('config')  
6  
7  db = SQLAlchemy(app)  
8
```

# / API de Receitas Gourmet

- Como aprendemos, agora você pode criar um pequeno modelo de banco de dados, com *db.model*

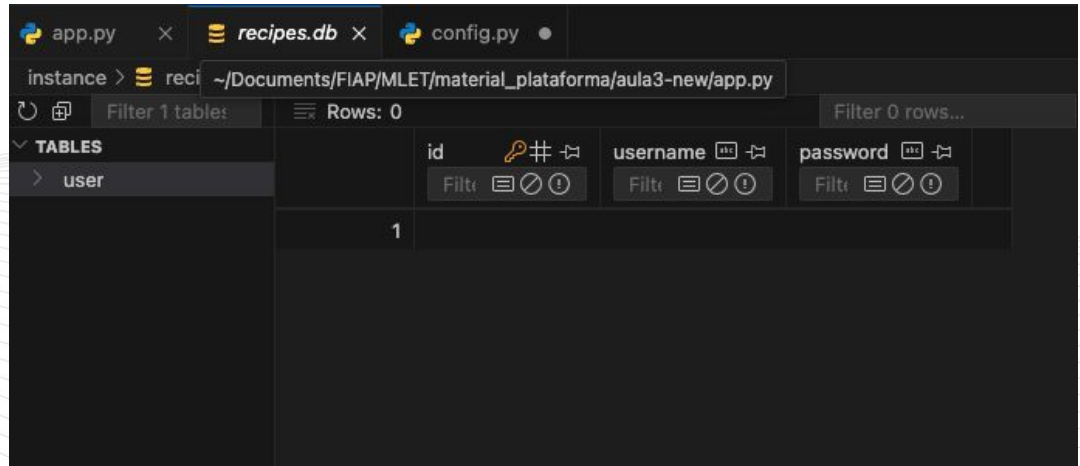
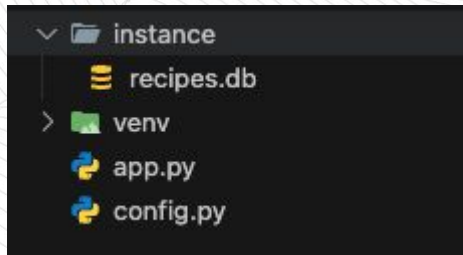
```
7 db = SQLAlchemy(app)
8 |
9 class User(db.Model):
10     id = db.Column(db.Integer, primary_key=True)
11     username = db.Column(db.String(80), unique=True, nullable=False)
12     password = db.Column(db.String(120), nullable=False)
13
14 if __name__ == '__main__':
15     with app.app_context():
16         db.create_all()
17         print("Banco de dados criado!")
```



Obs: Você pode visualizar o arquivo .db com a extensão do vscode: SQLite Viewer

## / API de Receitas Gourmet

- Rodando no terminal, é criado o banco de dados em uma nova pasta "instance"



# / API de Receitas Gourmet

- Vamos criar um banco de receitas

```
7 db = SQLAlchemy(app)
8 |
9 class User(db.Model):
10     id = db.Column(db.Integer, primary_key=True)
11     username = db.Column(db.String(80), unique=True, nullable=False)
12     password = db.Column(db.String(120), nullable=False)
13
14 if __name__ == '__main__':
15     with app.app_context():
16         db.create_all()
17         print("Banco de dados criado!")
```



## / API de Receitas Gourmet

- Mas e agora? Onde entra nossa API?
  - Isso veremos no próximo vídeo!

POSTECH

FIAP + alura