

Machine Learning Engineer

Python para ML e IA

Ferramentas Necessárias

Leonardo Pena

/ Seja muito bem vindo



FERRAMENTAS

Foco em validação de dados (Pydantic) e mapeamento de DB (SQLAlchemy)



MIGRAÇÕES

Migrações de schema com Alembic para evoluir banco de dados



VANTAGEM

Independência de framework: conceitos aplicáveis no Flask, FastAPI ou Django

Objetivo dessa primeira parte



Passo 1

Mostrar como
Pydantic
ajuda a
validar dados
de entrada e
saída



Passo 2

Entender
SQLAlchemy
como ORM
para mapear
tabelas em
classes Python



Passo
3

Explicar
Alembic
como
ferramenta
para
versionar
esquema do
banco



Passo 4

Demonstrar
fluxo básico:
criar modelo,
gerar
migração,
aplicar ao DB



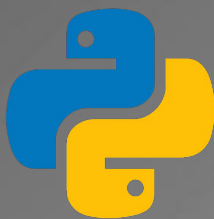
Passo
5

Preparar
terreno para
integrações
e projetos
reais



Validação de Dados





/ Conceito de Validação de Dados



- Inputs e outputs inconsistentes geram bugs e problemas em produção
- Pydantic adiciona tipagem forte à aplicação Python
- Rejeita dados fora do formato esperado (ex.: `age="abc"` se deve ser `int`)
- Evita poluir o DB com dados “quebrados”
- Facilita debugging e reforça confiança no sistema


```
# Dados recebidos de uma API (JSON)
data = {
    "username": "alice",
    "email": "alice@example.com",
    "age": "30", # Problema: idade deveria ser int, mas veio como string
    "is_active": "true" # Problema: booleano veio como string
}
```

Validação manual sem Pydantic

```
def validate_data(data):
    if not isinstance(data.get("username"), str):
        raise ValueError("username deve ser uma string")
    if not isinstance(data.get("email"), str) or "@" not in data["email"]:
        raise ValueError("email inválido")
    if not isinstance(data.get("age"), int):
        raise ValueError("age deve ser um número inteiro")
    if not isinstance(data.get("is_active"), bool):
        raise ValueError("is_active deve ser um booleano")
    return True

try:
    validate_data(data)
except ValueError as e:
    print(f"Erro na validação: {e}")
```

/ Validação de dados manual



- Problema:
 - Precisamos escrever várias verificações manuais.
 - Código fica repetitivo e difícil de manter conforme os requisitos aumentam.
 - Tipos errados precisam ser convertidos manualmente.



/ Validação de dados com Pydantic

```
from pydantic import BaseModel, EmailStr

# Definindo o modelo de validação com Pydantic
class UserModel(BaseModel):
    username: str
    email: EmailStr # Validação automática para formato de e-mail
    age: int
    is_active: bool

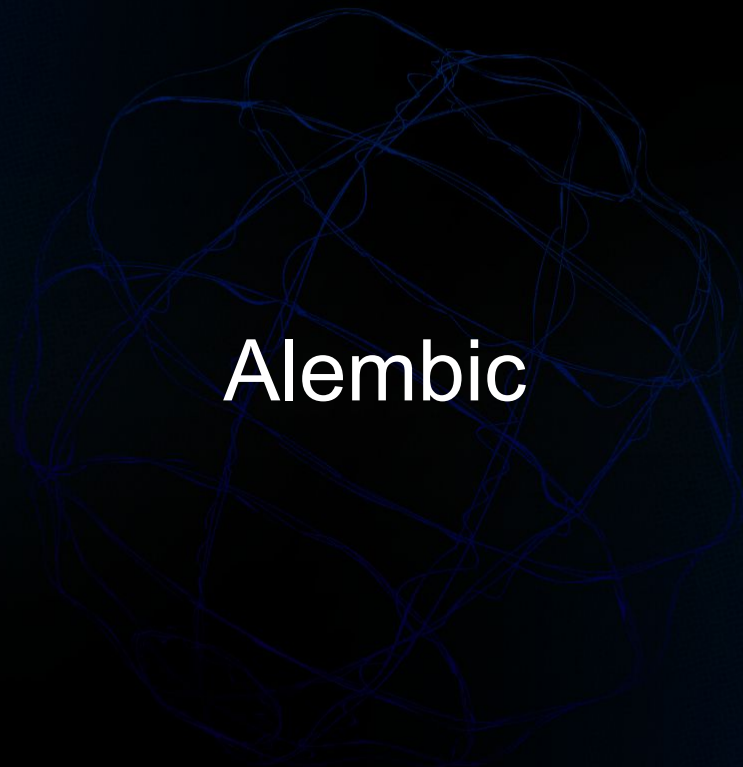
# Validação automática e conversão de tipos
try:
    user = UserModel(**data) # Passa os dados diretamente
    print(user) # Dados já validados e convertidos corretamente
except Exception as e:
    print(f"Erro na validação: {e}")
```

- Solução:
 - **Validação automática:** Checa tipos e formatos (como e-mail) sem necessidade de código extra.
 - **Conversão de tipos:** Converte automaticamente valores como "30" para int e "true" para bool.
 - **Erros claros:** Mensagens amigáveis explicam por que os dados são inválidos.
 - **Facilidade de manutenção:** Adicionar novos campos ou validações é simples.

/ Vantagens do Pydantic



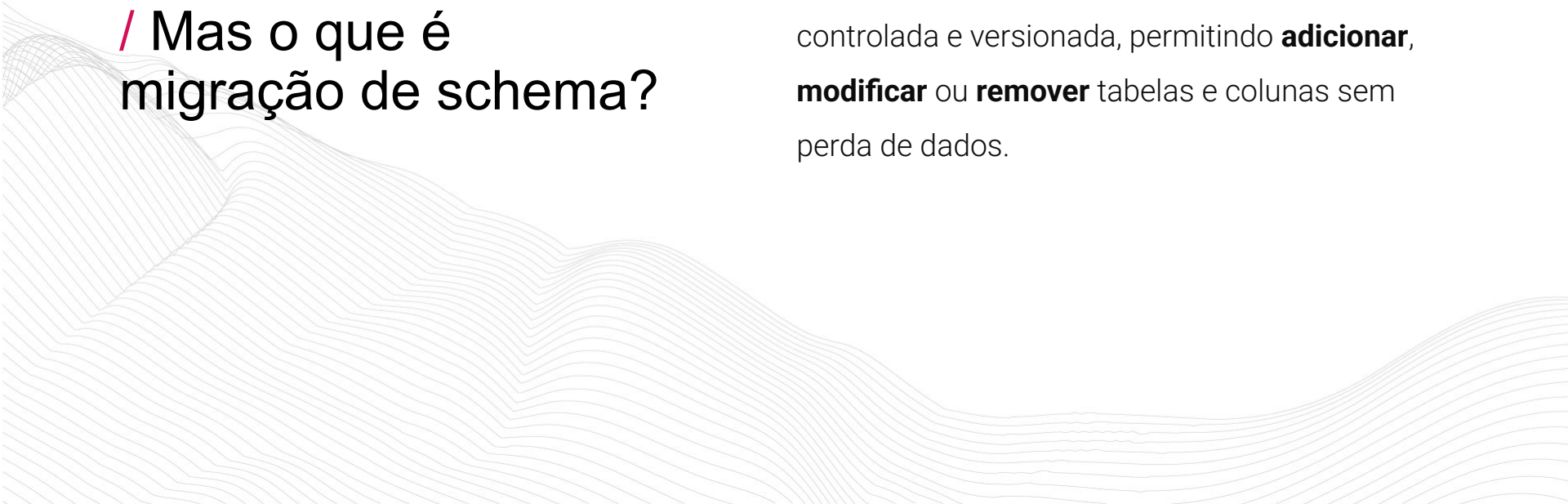
- Foco em performance e velocidade, mesmo com validação de tipos
- Retorna mensagens de erro claras e detalhadas
- Pode validar tipos complexos (List, Dict, submodels)
- Integra-se facilmente com FastAPI e Flask
- Reduz retrabalho ao lidar com dados externos (JSON, formulários)





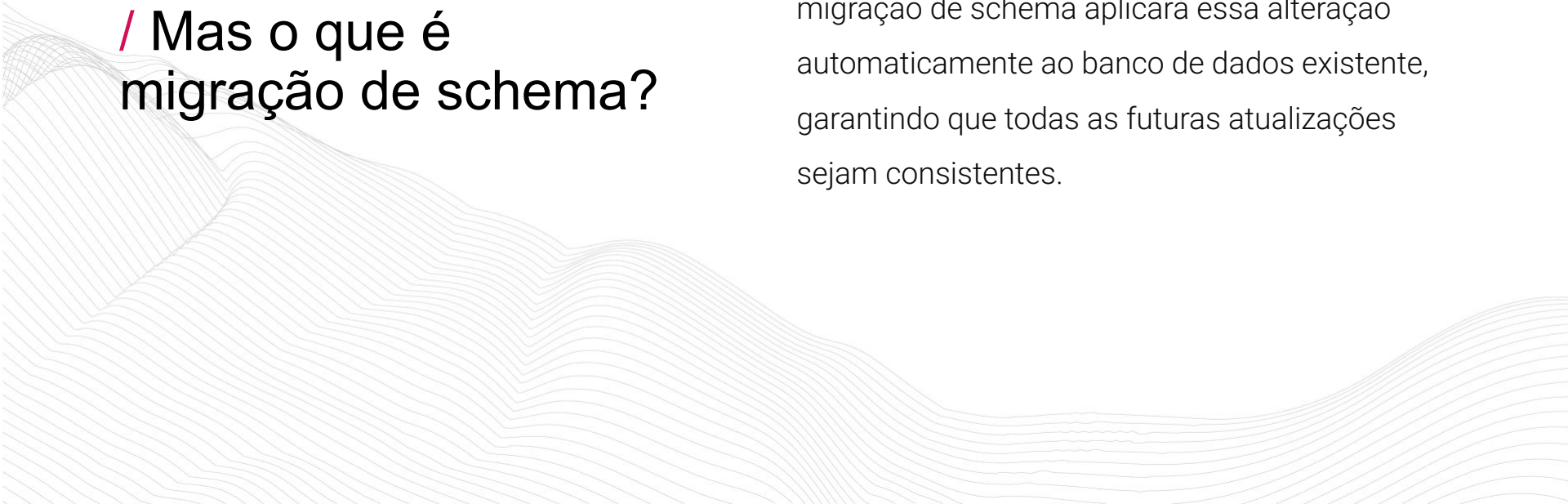
/ O que é?

- Ferramenta de **migração de schema**
- Gera scripts de migração quando alteramos modelos SQLAlchemy
- Permite upgrades (adicionar colunas, tabelas) e downgrades (reverter alterações)
- Mantém histórico de mudanças em diretório versionado
- Essencial para projetos de longa duração que mudam o DB frequentemente



/ Mas o que é migração de schema?

Migração de schema é o processo de **alterar a estrutura do banco** de dados de forma controlada e versionada, permitindo **adicionar**, **modificar** ou **remover** tabelas e colunas sem perda de dados.



/ Mas o que é migração de schema?

Por exemplo, ao decidir incluir uma nova coluna "*data_de_nascimento*" na tabela "usuarios", uma migração de schema aplicará essa alteração automaticamente ao banco de dados existente, garantindo que todas as futuras atualizações sejam consistentes.



/ API de receitas

Por exemplo, na API de receitas desenvolvida na última aula, ao adicionar uma nova coluna "tempo_de_preparo" na tabela "receitas", uma migração de schema com Alembic atualiza automaticamente o banco de dados para incluir essa nova informação sem afetar os dados existentes.



/ Exemplo

Abaixo vamos dar um exemplo, em que vamos:

- **Configurar o Projeto:** Preparar o ambiente e instalar as dependências necessárias.
- **Definir os Modelos:** Criar as classes de modelos utilizando SQLAlchemy.
- **Inicializar o Alembic:** Configurar o Alembic no projeto para gerenciar migrações.
- **Gerar Migrações:** Criar scripts de migração automaticamente com base nas alterações nos modelos.
- **Aplicar e Gerenciar Migrações:** Executar as migrações no banco de dados e manter o histórico de mudanças.

/

Vamos
começar
criando uma
estrutura
com:
app.py

app.py 2 x

app.py > ...

```
1  from flask import Flask
2  from flask_sqlalchemy import SQLAlchemy
3
4  app = Flask(__name__)
5  # Configuração do banco de dados SQLite
6  app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///meu_banco.db'
7  app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
8
9  db = SQLAlchemy(app)
10
11 # Importa os modelos
12 import models
13
14 if __name__ == '__main__':
15     app.run(debug=True)
16
```

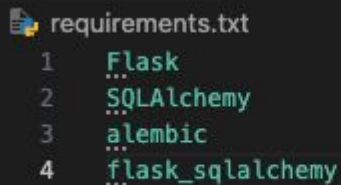
/

Vamos começar criando uma estrutura com: models.py

```
models.py > ...
1  from app import db
2
3  class Usuario(db.Model):
4      __tablename__ = 'usuarios'
5      id = db.Column(db.Integer, primary_key=True)
6      nome = db.Column(db.String(50), nullable=False)
7      email = db.Column(db.String(120), unique=True, nullable=False)
8
9      def __repr__(self):
10         return f'<Usuario {self.nome}>'
11
12  class Receita(db.Model):
13      __tablename__ = 'receitas'
14      id = db.Column(db.Integer, primary_key=True)
15      titulo = db.Column(db.String(100), nullable=False)
16      descricao = db.Column(db.Text, nullable=False)
17      usuario_id = db.Column(db.Integer, db.ForeignKey('usuarios.id'), nullable=False)
18
19      usuario = db.relationship('Usuario', backref=db.backref('receitas', lazy=True))
20
21      def __repr__(self):
22         return f'<Receita {self.titulo}>'
23
```

/

Vamos
começar
criando uma
estrutura
com:
requirements.
txt



```
requirements.txt
1  Flask
2  SQLAlchemy
3  alembic
4  flask_sqlalchemy
```

A decorative graphic at the bottom of the slide consisting of numerous thin, light gray wavy lines that create a sense of movement and depth, resembling a stylized landscape or water ripples.

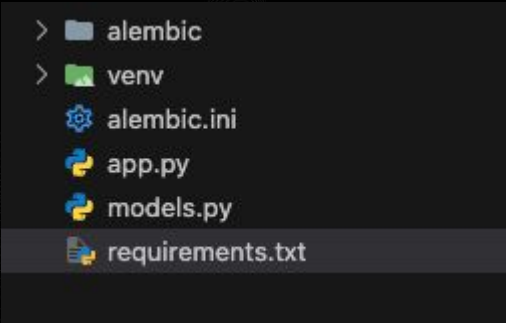
/ Inicialize o ambiente

- Crie o venv
- `pip install -r requirements.txt`
- **`alembic init alembic`**

/

Por fim, a estrutura
ficará assim.

A pasta alembic será
criada com alguns
arquivos



- > alembic
- > venv
- alembic.ini
- app.py
- models.py
- requirements.txt

/

alembic.ini:
Configuração principal do alembic, onde são definidas as informações de conexão com o banco de dados e as opções necessárias para gerenciar as migrações de schema.

```
alembic.ini x
alembic.ini
1  # A generic, single database configuration.
2
3  [alembic]
4  # path to migration scripts
5  # Use forward slashes (/) also on windows to provide an os agnostic path
6  script_location = alembic
7
8  # template used to generate migration file names; The default value is %(rev)s_%(slug)s
9  # Uncomment the line below if you want the files to be prepended with date and time
10 # see https://alembic.sqlalchemy.org/en/latest/tutorial.html#editing-the-ini-file
11 # for all available tokens
12 # file_template = %(year)d_%(month).2d_%(day).2d_%(hour).2d_%(minute).2d_%(rev)s_%(slug)s
13
14 # sys.path path, will be prepended to sys.path if present.
15 # defaults to the current working directory.
16 prepend_sys_path = .
17
18 # timezone to use when rendering the date within the migration file
19 # as well as the filename.
20 # If specified, requires the python>=3.9 or backports.zoneinfo library.
21 # Any required deps can installed by adding 'alembic[tz]' to the pip requirements
22 # string value is passed to ZoneInfo()
23 # leave blank for localtime
24 # timezone =
25
26 # max length of characters to apply to the "slug" field
27 # truncate_slug_length = 40
```


/

Lá vamos
configurar o url
do banco de
dados criado

```
63  
64 sqlalchemy.url = sqlite:///meu_banco.db  
65  
66
```

/

env.py: é um script que configura o ambiente de migração do Alembic, estabelecendo a conexão com o banco de dados e definindo o metadata dos modelos para gerenciar as migrações.

```
env.py 2 X
alembic > env.py > ...
1  from logging.config import fileConfig
2
3  from sqlalchemy import engine_from_config
4  from sqlalchemy import pool
5
6  from alembic import context
7
8  # this is the Alembic Config object, which provides
9  # access to the values within the .ini file in use.
10 config = context.config
11
12 # Interpret the config file for Python logging.
13 # This line sets up loggers basically.
14 if config.config_file_name is not None:
15     fileConfig(config.config_file_name)
16
17 # add your model's MetaData object here
18 # for 'autogenerate' support
19 # from myapp import mymodel
20 # target_metadata = mymodel.Base.metadata
21 target_metadata = None
22
23 # other values from the config, defined by the needs of env.py,
24 # can be acquired:
25 # my_important_option = config.get_main_option("my_important_option")
26 # ... etc.
```

/

O novo env.py será

```
1  from logging.config import fileConfig
2  from sqlalchemy import engine_from_config
3  from sqlalchemy import pool
4  from alembic import context
5
6  import sys
7  import os
8  sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
9  from app import db
10 import models
11
12 # Configuração de logging
13 fileConfig(context.config.config_file_name)
14
15 # MetaData para o Alembic
16 target_metadata = db.metadata
17
18 def run_migrations_offline():
19     """Executa migrações no modo offline."""
20     url = context.config.get_main_option("sqlalchemy.url")
21     context.configure(
22         url=url, target_metadata=target_metadata, literal_binds=True
23     )
24
25     with context.begin_transaction():
26         context.run_migrations()
```

/

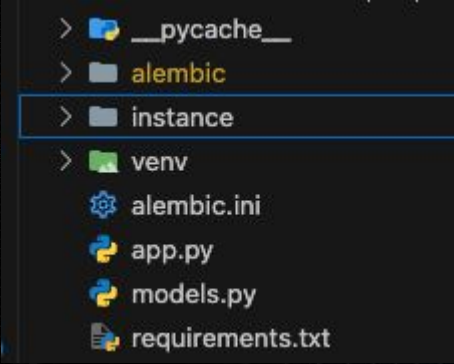
O novo env.py será

```
28 def run_migrations_online():
29     """Executa migrações no modo online."""
30     connectable = engine_from_config(
31         context.config.get_section(context.config.config_ini_section),
32         prefix='sqlalchemy.',
33         poolclass=pool.NullPool,
34     )
35
36     with connectable.connect() as connection:
37         context.configure(connection=connection, target_metadata=target_metadata)
38
39         with context.begin_transaction():
40             context.run_migrations()
41
42 if context.is_offline_mode():
43     run_migrations_offline()
44 else:
45     run_migrations_online()
46
```


/

Agora só rodar.
Primeiro a
aplicação pra
criar o banco
inicial

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula4-alembic % python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 118-084-665
```



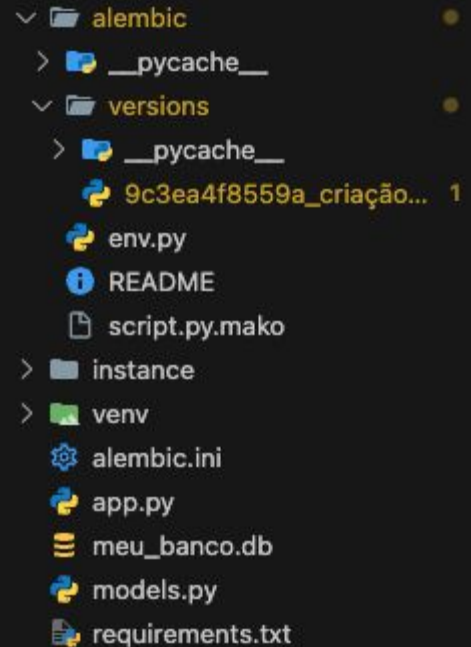
```
> __pycache__
> alembic
> instance
> venv
  alembic.ini
  app.py
  models.py
  requirements.txt
```

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula4-alembic % alembic revision --autogenerate -m "Criação das tabelas iniciais"

INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate.compare] Detected added table 'usuarios'
INFO [alembic.autogenerate.compare] Detected added table 'receitas'
Generating
/Users/leonardopena/Documents/FIAP/MLET/material_plataforma/aula4-alembic/alembic/versions/9c3ea4f8559a_criação_das_tabelas_iniciais.py ... done
```

/

Depois no terminal, criamos o arquivo de migração na pasta alembic/versions/

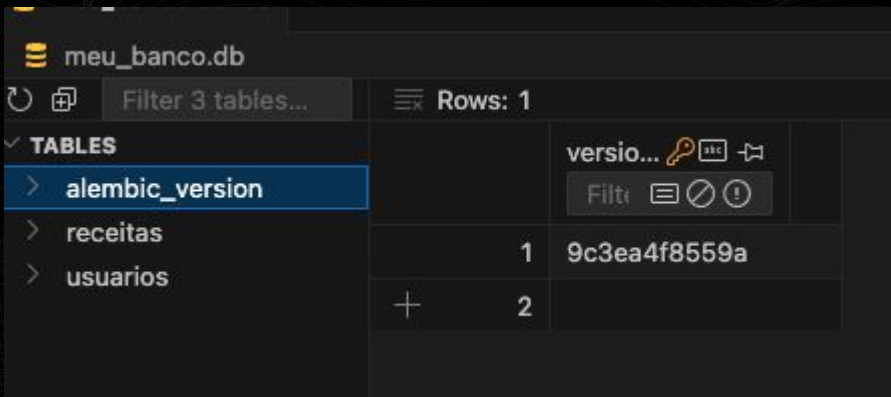


- alembic
 - __pycache__
 - versions
 - __pycache__
 - 9c3ea4f8559a_criação... 1
 - env.py
 - README
 - script.py.mako
- instance
- venv
- alembic.ini
- app.py
- meu_banco.db
- models.py
- requirements.txt

/

O comando alembic upgrade head irá aplicar a migração

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula4-alembic % alembic upgrade head
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> 9c3ea4f8559a, Criação das tabelas iniciais
```



meu_banco.db

Filter 3 tables...

Rows: 1

TABLES

- > alembic_version
- > receitas
- > usuarios

alembic_version	version
1	9c3ea4f8559a
+	2

/

Poderíamos
realizar
alteracoes,
mudando
models.py e
adicionando a
coluna
*data_de_nascim
ento*

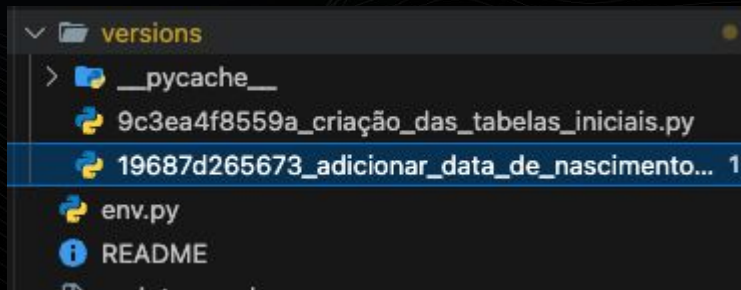
```
models.py x
~/Documents/FIAP/MLET/material_plataforma/aula4-alembic/models.py
1  from app import db
2
3  class Usuario(db.Model):
4      __tablename__ = 'usuarios'
5      id = db.Column(db.Integer, primary_key=True)
6      nome = db.Column(db.String(50), nullable=False)
7      email = db.Column(db.String(120), unique=True, nullable=False)
8      data_de_nascimento = db.Column(db.Date, nullable=True) # Nova coluna
9
10     def __repr__(self):
11         return f'<Usuario {self.nome}>'
12
13
14     class Receita(db.Model):
15         __tablename__ = 'receitas'
16         id = db.Column(db.Integer, primary_key=True)
17         titulo = db.Column(db.String(100), nullable=False)
18         descricao = db.Column(db.Text, nullable=False)
19         usuario_id = db.Column(db.Integer, db.ForeignKey('usuarios.id'), nullable=False)
20
21         usuario = db.relationship('Usuario', backref=db.backref('receitas', lazy=True))
22
23     def __repr__(self):
24         return f'<Receita {self.titulo}>'
25
```

/

Da mesma forma, criamos a migração e vemos em alembic/versions ela gerada

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula4-alembic % alembic revision --autogenerate -m "Adicionar data_de_nascimento na tabela usuarios"
```

```
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.autogenerate.compare] Detected added column 'usuarios.data_de_nascimento'
Generating /Users/leonardopena/Documents/FIAP/MLET/material_plataforma/aula4-alembic/alembic/versions/19687d265673_adicionar_data_de_nascimento_na_tabela_.py ... done
```



```
4 Revises: 9c3ea4f8559a
5 Create Date: 2025-01-10 16:58:13.843745
6
7 """
8 from typing import Sequence, Union
9
10 from alembic import op
11 import sqlalchemy as sa
12
13
14 # revision identifiers, used by Alembic.
15 revision: str = '19687d265673'
16 down_revision: Union[str, None] = '9c3ea4f8559a'
17 branch_labels: Union[str, Sequence[str], None] = None
18 depends_on: Union[str, Sequence[str], None] = None
19
20
21 def upgrade() -> None:
22     """ commands auto generated by Alembic - please adjust! """
23     op.add_column('usuarios', sa.Column('data_de_nascimento', sa.Date(), nullable=True))
24     """ end Alembic commands """
25
26
27 def downgrade() -> None:
28     """ commands auto generated by Alembic - please adjust! """
29     op.drop_column('usuarios', 'data_de_nascimento')
30     """ end Alembic commands """
```

/

Com alembic upgrade head veremos a alteração no *meu_banco.db*

```
(venv) leonardopena@MacBook-Pro-de-Leonardo aula4-alembic % alembic upgrade head
```

```
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
```

```
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
```

```
INFO [alembic.runtime.migration] Running upgrade 9c3ea4f8559a -> 19687d265673, Adicionar data_de_nascimento na tabela usuarios
```

The screenshot shows a database browser interface for 'meu_banco.db'. The left sidebar lists tables: 'alembic_version', 'receitas', and 'usuarios' (selected). The main area shows the 'usuarios' table with columns: 'id' (primary key), 'nome', 'email', and 'data_de...'. The table is currently empty (Rows: 0). A blue 'Upgrade' button is visible in the top right corner.

TABLES		id	nome	email	data_de...
> alembic_version					
> receitas					
> usuarios	+	1			

POSTECH

FIAP + alura