LINK PARA O DRIVE COM TODOS OS ARQUIVOS MOSTRADOS NESSE DOCUMENTO:

aluguel_banco (1)

INTRODUÇÃO

Neste projeto, criei um banco de dados para gerenciamento propriedades imobiliárias usando SQLAlchemy. O banco de dados possui cinco tipos principais de dados: Pessoa, Corretor, Proprietário, Inquilino e Imóvel, sendo estas as entidades.

Pessoa é a base de tudo, representando qualquer individuo envolvida no sistema, como corretores, proprietários e inquilinos. Corretor é um tipo especial de Pessoa que cuida da administração de imóveis. Proprietário e Inquilino também são tipos de Pessoa, mas com papéis diferentes: um é o dono do imóvel e o outro é quem aluga. Por fim, Imóvel é o objeto principal, representando as propriedades que são geridas pelos proprietários e alugadas pelos inquilinos.

O objetivo é criar um sistema que permita adicionar, consultar, atualizar e excluir informações sobre todos esses elementos, facilitando a gestão de propriedades imobiliárias.

DESCRIÇÃO TEXTUAL

1. Pessoa

Pessoa é usada para representar qualquer indivíduo que esteja envolvida no sistema, ou seja corretores, proprietários e inquilinos. Todos possuem os seguintes atributos:

- id: Um número único que identifica cada pessoa.
- nome: O nome completo da pessoa, com até 100 caracteres.
- cpf: O número do CPF, que é único para cada pessoa e tem 14 caracteres.
- telefone: O número de telefone da pessoa, com no máximo 20 caracteres.
- email: O e-mail da pessoa, limitado a 100 caracteres.
- endereco: O endereço da pessoa, com até 255 caracteres.

2. Corretor

Corretor é uma especialização de Pessoa que representa o profissional responsável pela intermediação de imóveis entre proprietário e inquilino. Cada corretor é caracterizado por:

- id: Um identificador único que também serve como chave primária e referência à tabela Pessoa.
- creci: O número do CRECI (registo do corretor), é único e tem até 20 caracteres.

3. Proprietário

Proprietario é uma Pessoa que possui imóveis, sendo assim uma especialização de Pessoa. Além dos atributos de uma pessoa, o proprietário tem:

- id: Um número único que também é a chave primária e se refere à tabela Pessoa.
- corretor_id: Um número que se refere ao corretor que cuida do proprietário, vinculado à tabela Corretor.

4. Inquilino

Inquilino representa uma Pessoa que aluga um imóvel. Possuindo os seguintes atributos:

- id: Um número único que identifica o inquilino e serve como chave primária, se referindo à tabela Pessoa.
- corretor_id: Um número que aponta para o corretor que cuida do inquilino, referenciado na tabela Corretor.

5. Imóvel

Imovel representa as propriedades que estão no sistema. Cada imóvel tem as seguintes características:

- id: Um número único para identificar o imóvel.
- descricao: Uma descrição do imóvel, com até 255 caracteres.
- endereco: O local onde o imóvel está situado, com até 255 caracteres.
- valor: O preço do imóvel, em formato numérico de ponto flutuante.
- tipo: O tipo de imóvel, como apartamento ou casa, com até 50 caracteres.
- proprietario_id: Número que referencia o proprietário do imóvel, vindo da tabela Proprietario.
- inquilino_id: Número que aponta para o inquilino que está alugando o imóvel, vindo da tabela Inquilino.

RELACIONAMENTO ENTRE ENTIDADES:

Proprietário possui imóvel

- Um único proprietário pode ter vários imóveis e cada imóvel pertence a um único proprietário.

Pessoa e Proprietário

- Cada proprietário é uma pessoa. A entidade Pessoa serve como base para a entidade Proprietário.

Proprietário contacta Corretor

- Um corretor pode estar associado a vários proprietários e cada proprietário pode ser associado a um único corretor.

Pessoa e Corretor

- Cada corretor é uma pessoa. A entidade Pessoa é a base para a entidade Corretor.

Corretor atende Inquilino

- Um corretor pode atender vários inquilinos e cada inquilino é atendido por um único corretor.

Pessoa e Inquilino

- Cada inquilino é uma pessoa. A entidade Pessoa serve como base para a entidade Inquilino.

Inquilino aluga imóvel

- Um imóvel pode ser alugado por um inquilino ou estar vazio e cada imóvel pode ter no máximo um inquilino

CARDINALIDADES:

Proprietário (0, N) ---- Imóvel (1, 1)

Corretor (0, N) ---- Proprietário (1, 1)

Corretor (0, N) ---- Inquilino (1, 1)

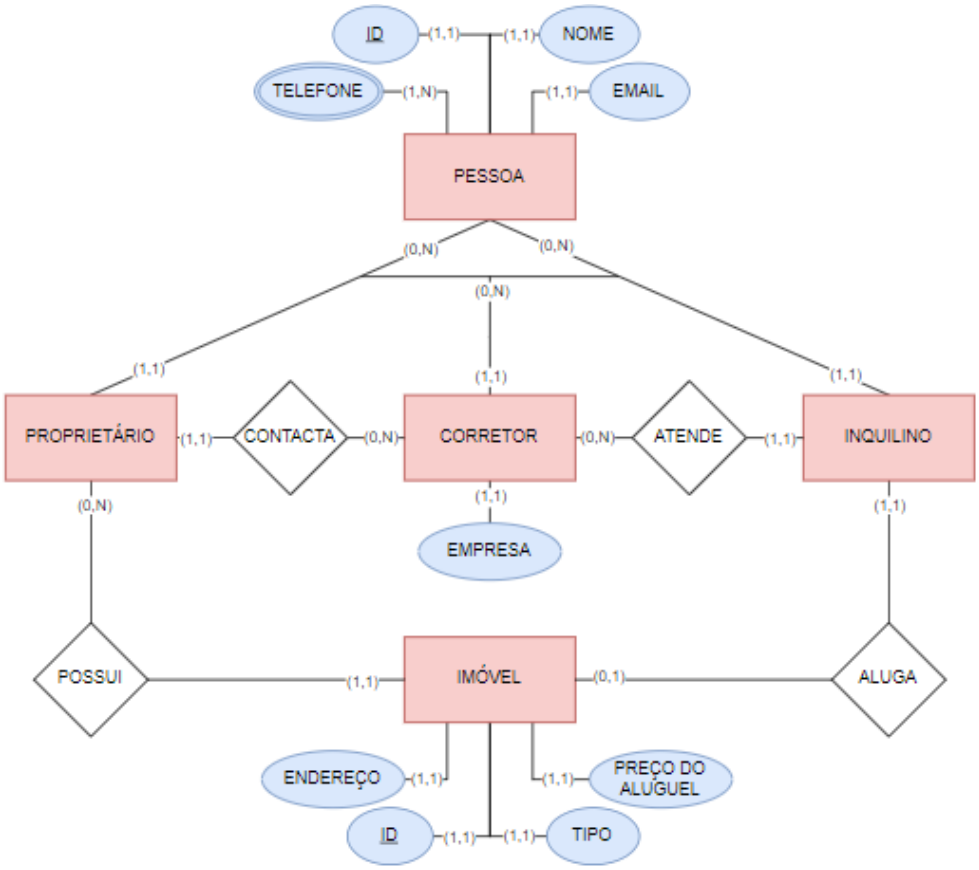
Imóvel (0, 1) ---- Inquilino (1, 1)

Pessoa (0, N) ---- Proprietário (1, 1)

Pessoa (0, N) ---- Corretor (1, 1)

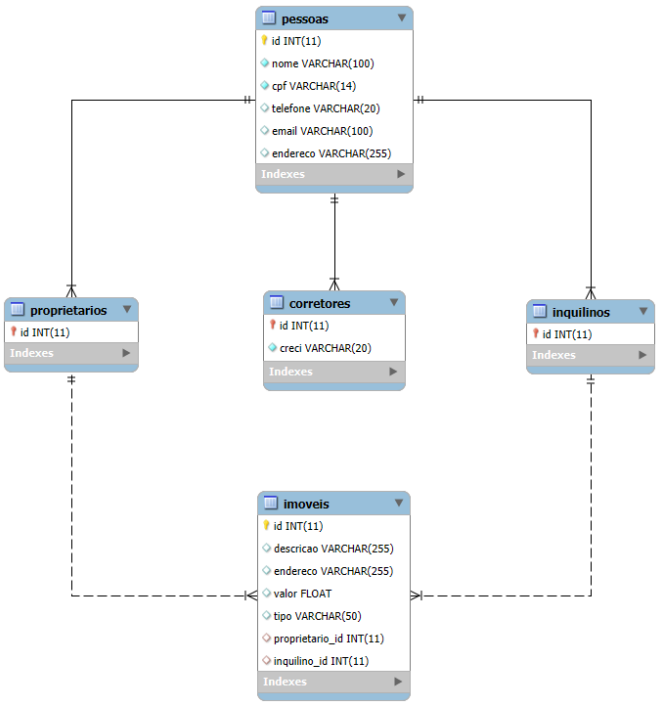
Pessoa (0, N) ---- Inquilino (1, 1)

MAPA CONCEITUAL



MYSQL WORKBENCH

Modelagem visual das tabelas: pessoas, proprietários, corretores, inquilinos e imóveis.



Script das tabelas demonstradas acima:

aluguel_banco*

Limit to 1000 rows

```sql
create DATABASE aluguel_bancos;

use aluguel_banco;

CREATE TABLE pessoas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cpf VARCHAR(14) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    email VARCHAR(100),
    endereco VARCHAR(255)
);

CREATE TABLE corretores (
    id INT PRIMARY KEY,
    creci VARCHAR(20) UNIQUE NOT NULL,
    FOREIGN KEY (id) REFERENCES pessoas(id) ON DELETE CASCADE
);
```

aluguel_banco*

Limit to 1000 rows

```sql

CREATE TABLE proprietarios (
    id INT PRIMARY KEY,
    FOREIGN KEY (id) REFERENCES pessoas(id) ON DELETE CASCADE
);

CREATE TABLE inquilinos (
    id INT PRIMARY KEY,
    FOREIGN KEY (id) REFERENCES pessoas(id) ON DELETE CASCADE
);
```

aluguel_banco*

Limit to 1000 rows

```sql

CREATE TABLE imoveis (
    id INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(255),
    endereco VARCHAR(255),
    valor FLOAT,
    tipo VARCHAR(50),
    proprietario_id INT,
    inquilino_id INT,
    FOREIGN KEY (proprietario_id) REFERENCES proprietarios(id) ON DELETE SET NULL,
    FOREIGN KEY (inquilino_id) REFERENCES inquilinos(id) ON DELETE SET NULL
);
```

aluguel_banco*

Limit to 1000 rows

```sql

CREATE TABLE imoveis (
    id INT AUTO_INCREMENT PRIMARY KEY,
    descricao VARCHAR(255),
    endereco VARCHAR(255),
    valor FLOAT,
    tipo VARCHAR(50),
    proprietario_id INT,
    inquilino_id INT,
    FOREIGN KEY (proprietario_id) REFERENCES proprietarios(id) ON DELETE SET NULL,
    FOREIGN KEY (inquilino_id) REFERENCES inquilinos(id) ON DELETE SET NULL
);
```

CÓDIGO PYTHON

```python
from sqlalchemy import create_engine, Column, Integer, String, Float,
ForeignKey
from sqlalchemy.orm import sessionmaker, declarative_base, relationship
from sqlalchemy.engine.url import URL

# CONFIGURAR CONEXÃO
DATABASE_URL = URL.create(
    drivername="mysql+mysqlconnector",
    username="root",
    password="",
    host="localhost",
    database="aluguel_bancos"
)

engine = create_engine(DATABASE_URL, echo=True)
Base = declarative_base()
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

# CLASSES
class Pessoa(Base):
    __tablename__ = 'pessoas'
    id = Column(Integer, primary_key=True, index=True)
    nome = Column(String(100))
    cpf = Column(String(14), unique=True)
    telefone = Column(String(20))
    email = Column(String(100))
    endereco = Column(String(255))

class Corretor(Base):
    __tablename__ = 'corretores'
    id = Column(Integer, ForeignKey('pessoas.id'), primary_key=True)
```

```
    creci = Column(String(20), unique=True)

class Proprietario(Base):
    __tablename__ = 'proprietarios'
    id = Column(Integer, ForeignKey('pessoas.id'), primary_key=True)
    corretor_id = Column(Integer, ForeignKey('corretores.id'))

class Inquilino(Base):
    __tablename__ = 'inquilinos'
    id = Column(Integer, ForeignKey('pessoas.id'), primary_key=True)
    corretor_id = Column(Integer, ForeignKey('corretores.id'))

class Imovel(Base):
    __tablename__ = 'imoveis'
    id = Column(Integer, primary_key=True, index=True)
    descricao = Column(String(255))
    endereco = Column(String(255))
    valor = Column(Float)
    tipo = Column(String(50))
    proprietario_id = Column(Integer, ForeignKey('proprietarios.id'))
    inquilino_id = Column(Integer, ForeignKey('inquilinos.id'),
nullable=True)

# CRUD

# CRIAR
def criar(session, model, **kwargs):
    obj = session.query(model).filter_by(**{k: v for k, v in
kwargs.items() if k != 'id'}).first()
    if obj: return obj
    obj = model(**kwargs)
    session.add(obj)
    session.commit()
    session.refresh(obj)
    return obj

# LISTAR
def listar(session, model):
    return session.query(model).all()

#ATUALIZAR
def atualizar(session, model, obj_id, **kwargs):
    obj = session.query(model).get(obj_id)
    for k, v in kwargs.items():
        if v: setattr(obj, k, v)
    session.commit()

# DELETAR
def deletar(session, model, obj_id):
```

```
    obj = session.query(model).get(obj_id)
    session.delete(obj)
    session.commit()

# MENU INTERATIVO
def menu():
    session = SessionLocal()
    while True:
        opcao = int(input(
            "\n=== Menu Sistema de Aluguel de Imóveis ===\n"
            "1. Adicionar Pessoa\n2. Listar Pessoas\n3. Atualizar
Pessoa\n4. Deletar Pessoa\n"
            "5. Adicionar Corretor\n6. Listar Corretores\n7. Atualizar
Corretor\n8. Deletar Corretor\n"
            "9. Adicionar Proprietário\n10. Listar Proprietários\n11.
Atualizar Proprietário\n12. Deletar Proprietário\n"
            "13. Adicionar Inquilino\n14. Listar Inquilinos\n15.
Atualizar Inquilino\n16. Deletar Inquilino\n"
            "17. Adicionar Imóvel\n18. Listar Imóveis\n19. Atualizar
Imóvel\n20. Deletar Imóvel\n"
            "21. Sair\nEscolha uma opção: "))

        if opcao == 1:
            criar(session, Pessoa, nome=input("Nome: "), cpf=input("CPF:
"), telefone=input("Telefone: "), email=input("Email: "),
endereco=input("Endereço: "))
        elif opcao == 2:
            for p in listar(session, Pessoa): print(f"{p.id} - {p.nome},
{p.cpf}")
        elif opcao == 3:
            atualizar(session, Pessoa, int(input("ID da Pessoa: ")),
nome=input("Novo Nome: "), telefone=input("Novo Telefone: "),
email=input("Novo Email: "), endereco=input("Novo Endereço: "))
        elif opcao == 4:
            deletar(session, Pessoa, int(input("ID da Pessoa: ")))

        elif opcao == 5:
            criar(session, Corretor, id=int(input("ID da Pessoa: ")),
creci=input("CRECI: "))
        elif opcao == 6:
            for c in listar(session, Corretor): print(f"{c.id} - CRECI:
{c.creci}")
        elif opcao == 7:
            atualizar(session, Corretor, int(input("ID do Corretor: ")),
creci=input("Novo CRECI: "))
        elif opcao == 8:
            deletar(session, Corretor, int(input("ID do Corretor: ")))

        elif opcao == 9:
```

```
            criar(session, Proprietario, id=int(input("ID da Pessoa: ")),
corretor_id=int(input("ID do Corretor: ")))
        elif opcao == 10:
            for p in listar(session, Proprietario): print(f"{p.id} -
Corretor ID: {p.corretor_id}")
        elif opcao == 11:
            atualizar(session, Proprietario, int(input("ID do
Proprietário: ")), corretor_id=int(input("Novo Corretor ID: ")))
        elif opcao == 12:
            deletar(session, Proprietario, int(input("ID do Proprietário:
")))

        elif opcao == 13:
            criar(session, Inquilino, id=int(input("ID da Pessoa: ")),
corretor_id=int(input("ID do Corretor: ")))
        elif opcao == 14:
            for i in listar(session, Inquilino): print(f"{i.id} -
Corretor ID: {i.corretor_id}")
        elif opcao == 15:
            atualizar(session, Inquilino, int(input("ID do Inquilino:
")), corretor_id=int(input("Novo Corretor ID: ")))
        elif opcao == 16:
            deletar(session, Inquilino, int(input("ID do Inquilino: ")))

        elif opcao == 17:
            criar(session, Imovel, descricao=input("Descrição: "),
endereco=input("Endereço: "), valor=float(input("Valor: ")),
tipo=input("Tipo: "), proprietario_id=int(input("ID do Proprietário: ")))
        elif opcao == 18:
            for im in listar(session, Imovel): print(f"{im.id} -
{im.descricao}, {im.endereco}")
        elif opcao == 19:
            atualizar(session, Imovel, int(input("ID do Imóvel: ")),
descricao=input("Nova Descrição: "), endereco=input("Novo Endereço: "),
valor=float(input("Novo Valor: ")), tipo=input("Novo Tipo: "))
        elif opcao == 20:
            deletar(session, Imovel, int(input("ID do Imóvel: ")))

        elif opcao == 21:
            session.close()
            break

if __name__ == "__main__":
    menu()
```

EXPLICAÇÃO:

1° faz a conexão com o banco de dados já criado no MySQL

2° faz a declaração de classes que representam as tabelas criadas no banco de dados

3° Cria a sessão que gerencia os objetos e as transações de informações no banco de dados

4° Operações CRUD, que servem para criar, listar, atualizar e deletar dados no banco.