

# Gerenciador de Condomínio

Leonardo Fajardo Grupioni - RA00319703

Turma COM-MA7 Banco de Dados 2025



# Problema

- Síndicos enfrentam controles manuais (planilhas, grupos de WhatsApp)
- Falta de transparência para moradores
- Consequências → erros em cobranças, retrabalho



# Objetivo do Projeto

Criar uma aplicação simples, mas funcional, que:

- centralize dados de apartamentos, despesas, moradores
- permita comunicação (avisos) entre sindico e morador
- demonstre domínio de banco de dados



# Benchmark: uCondo

O projeto foi inspirado no App da uCondo, uma empresa especializada em criar plataformas para atender sindicos e condomínios.

Com os Pontos Chaves:

- Reservas de espaços;
- Comunicação em tempo real;
- Gestão financeira e de pagamentos;
- Avisos e anuncios.



# BMC - Businesses Model Canvas

## Proposta de Valor

Transparência,  
facilidade para  
síndicos e  
moradores

## Segmentos

Condomínios  
residenciais de até  
100 unidades

## Canais

Web app simples,  
auto-serviço

## Relacionamento

Suporte via e-mail

## Fluxo de receita

Assinatura básica  
(R\$ 100/unidade)

## Recursos chave

Banco de dados,  
infraestrutura  
cloud, time dev

## Atividades chave

Manutenção,  
suporte,  
segurança

## Parcerias

Gateways de  
pagamento,  
serviços de  
notificação

## Estrutura de custos

Servidor,  
SMS/email,  
desenvolvimento

# Requisitos Funcionais

1. RF-01 Gerenciar apartamentos (CRUD)
2. RF-02 Gerenciar moradores e papéis (síndico, morador)
3. RF-03 Lançar/consultar despesas
4. RF-04 Registrar pagamentos e status
5. RF-05 Enviar e ler notificações
6. RF-06 Exportar relatórios CSV/PDF



# Requisitos Não-Funcionais



1. RNF-01 Persistência em MySQL
2. RNF-02 Senhas com bcrypt
3. RNF-03 Interface Web responsiva  
(Streamlit)
4. RNF-04 Código modular (ORM,  
camada security)
5. RNF-05 Relatórios portáveis (CSV,  
PDF)

# ARQUITETURA



**FRONT-END**

Streamlit WEB



**BACK-END/ORM**

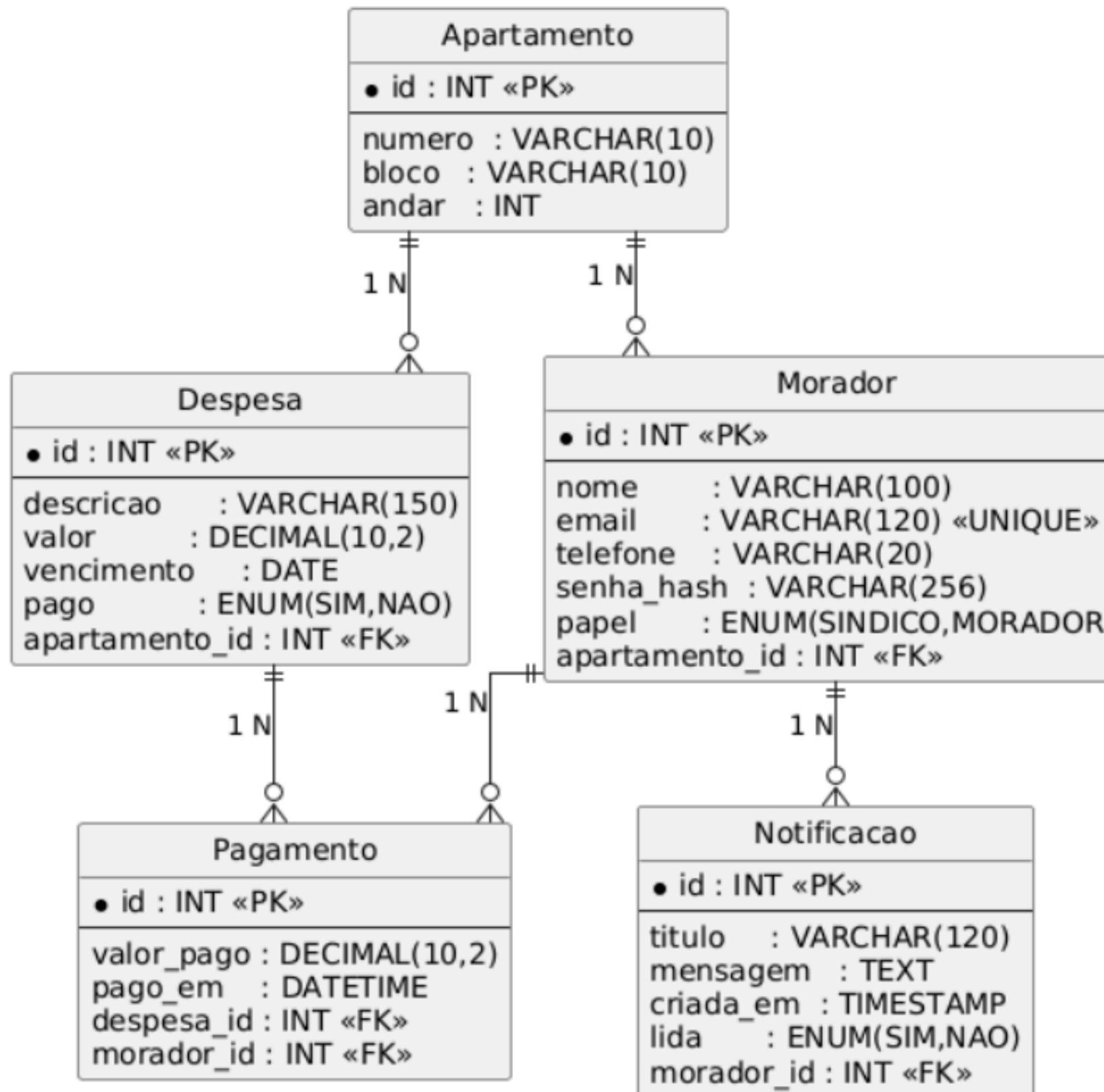
Python + SQL Alchemy



**BANCO DE DADOS**

MySQL Local

# MODELO DE DADOS (ER)



## Índices

idx\_despesa\_apto :  
despesa(apartamento\_id)  
idx\_notif\_morador :  
notificacao(morador\_id)

# Scripts DDL - Visão Geral

## Criação do Banco de Dados Condominio:

```
schema.sql
1 CREATE DATABASE IF NOT EXISTS condominio DEFAULT CHARSET=utf8mb4;
2 USE condominio;
```

## Índices Auxiliares:

```
-- Índices auxiliares
CREATE INDEX idx_despesa_apto ON despesa (apartamento_id);
CREATE INDEX idx_notif_morador ON notificacao (morador_id);
```

# Script Tabela: Apartamento

```
4  -- Apartamentos
5  CREATE TABLE apartamento (
6      id INT AUTO_INCREMENT PRIMARY KEY,
7      numero VARCHAR(10) NOT NULL,
8      bloco  VARCHAR(10),
9      andar  INT,
10     UNIQUE (numero, bloco)
11 );
```

# Script Tabela: Morador

```
13    -- Moradores
14    CREATE TABLE morador (
15        id INT AUTO_INCREMENT PRIMARY KEY,
16        nome      VARCHAR(100) NOT NULL,
17        email     VARCHAR(120) UNIQUE,
18        telefone  VARCHAR(20),
19        senha_hash VARCHAR(256),
20        papel ENUM('SINDICO','MORADOR') DEFAULT 'MORADOR',
21        apartamento_id INT,
22        FOREIGN KEY (apartamento_id) REFERENCES apartamento(id)
23            ON DELETE SET NULL ON UPDATE CASCADE
24    );
```

# Script Tabela: Despesa

```
26    -- Despesas
27    CREATE TABLE despesa (
28        id INT AUTO_INCREMENT PRIMARY KEY,
29        descricao VARCHAR(150) NOT NULL,
30        valor DECIMAL(10,2) NOT NULL,
31        vencimento DATE,
32        pago ENUM('SIM','NAO') DEFAULT 'NAO',
33        apartamento_id INT,
34        FOREIGN KEY (apartamento_id) REFERENCES apartamento(id)
35            ON DELETE CASCADE ON UPDATE CASCADE
36    );
```

# Script Tabela: Notificação

```
38  -- Notificações
39  CREATE TABLE notificacao (
40      id INT AUTO_INCREMENT PRIMARY KEY,
41      morador_id INT,
42      titulo VARCHAR(120),
43      mensagem TEXT,
44      criada_em TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
45      lida ENUM('SIM','NAO') DEFAULT 'NAO',
46      FOREIGN KEY (morador_id) REFERENCES morador(id)
47          ON DELETE CASCADE ON UPDATE CASCADE
48  );
```

# Script Tabela: Pagamento

```
50    -- Pagamentos
51    CREATE TABLE pagamento (
52        id INT AUTO_INCREMENT PRIMARY KEY,
53        despesa_id INT NOT NULL,
54        morador_id INT NOT NULL,
55        valor_pago DECIMAL(10,2) NOT NULL,
56        pago_em DATETIME DEFAULT CURRENT_TIMESTAMP,
57        FOREIGN KEY (despesa_id) REFERENCES despesa(id)
58            ON DELETE CASCADE ON UPDATE CASCADE,
59        FOREIGN KEY (morador_id) REFERENCES morador(id)
60            ON DELETE CASCADE ON UPDATE CASCADE,
61        UNIQUE (despesa_id, morador_id)
62    );
```

# Inicialização da Aplicação (init\_db.py)

```
1  from db import SessionLocal, Base, engine
2  from models import Morador
3  from security import hash_pwd
4
5  def main():
6      # Cria tabelas se ainda não existirem
7      Base.metadata.create_all(bind=engine)
8
9      db = SessionLocal()
10     # Verifica se já existe um síndico
11     if not db.query(Morador).filter(Morador.papel == "SINDICO").first():
12         admin = Morador(
13             nome="Síndico",
14             email="admin@condo.local",
15             senha_hash=hash_pwd("1234"),
16             papel="SINDICO"
17         )
18         db.add(admin)
19         db.commit()
20         print("Usuário síndico criado (email: admin@condo.local / senha: 1234)")
21     else:
22         print("Síndico já existe, nada a fazer.")
23     db.close()
24
25 if __name__ == "__main__":
26     main()
```

# Organização do Projeto

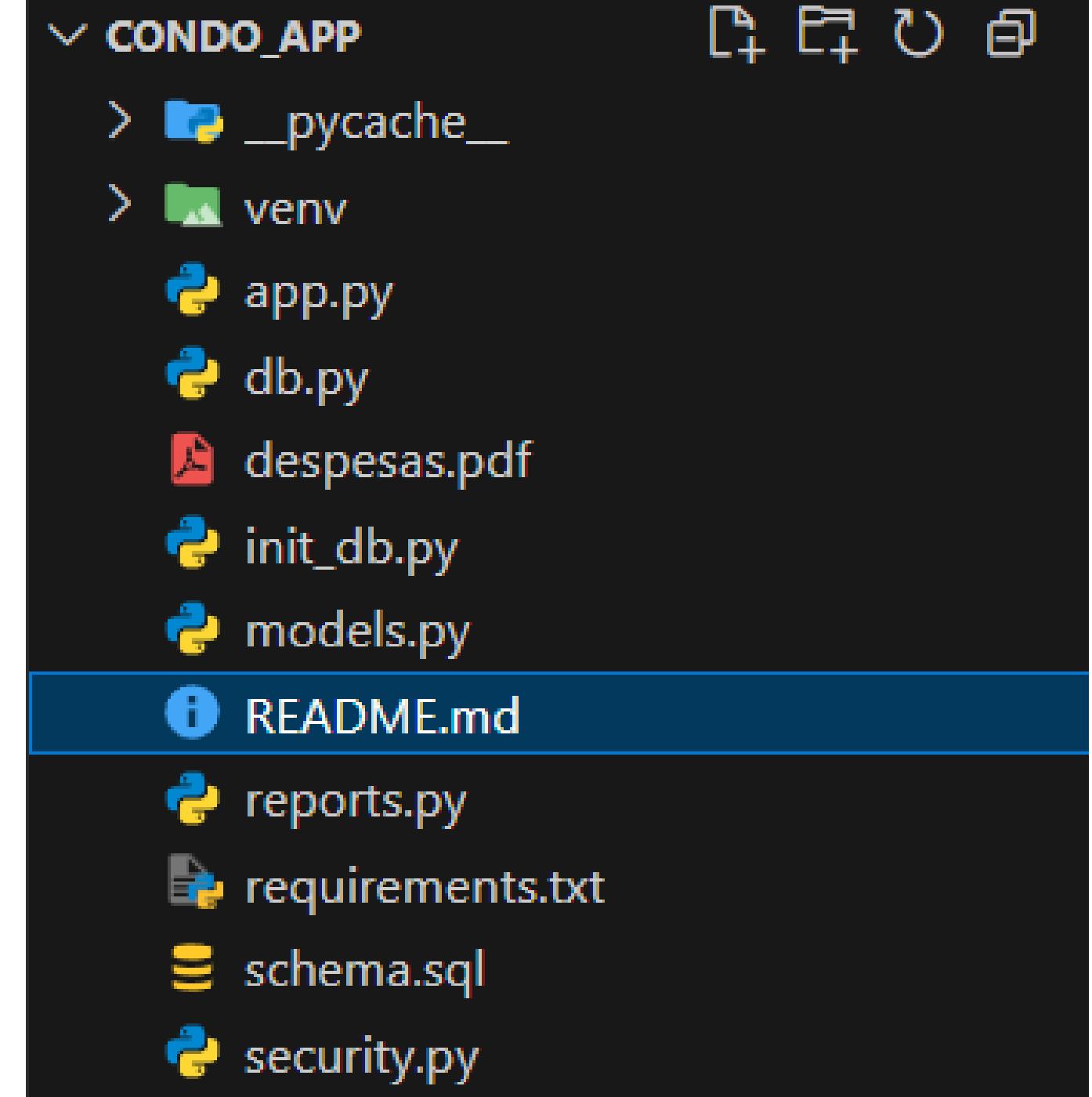
## Estrutura

```
condo_app/
├── venv          # Ambiente containerizado/venv
├── app.py
├── db.py
├── init_db.py    # cria tabelas e um síndico padrão
├── models.py
├── reports.py
├── security.py
└── schema.sql    # script DDL completo
└── requirements.txt
```

## Pré-requisitos

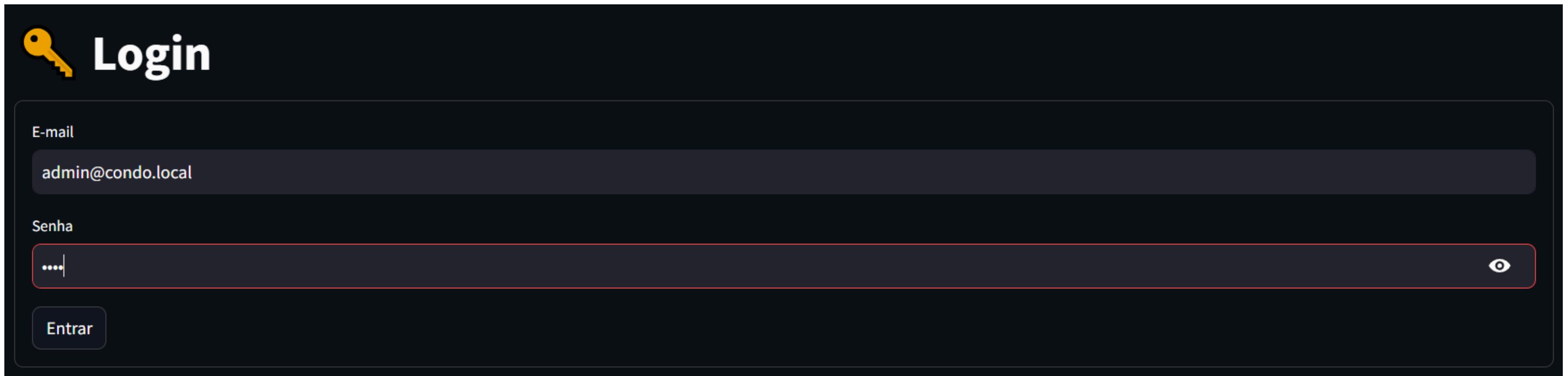
- Python ≥ 3.9
- MySQL ≥ 8

## CONDO\_APP



```
> __pycache__
> venv
python app.py
python db.py
pdf despesas.pdf
python init_db.py
python models.py
info README.md
python reports.py
text requirements.txt
text schema.sql
python security.py
```

# Aplicação: Tela de Login



A screenshot of a login interface. At the top left is a yellow key icon. To its right, the word "Login" is written in white. Below this is a form with two input fields: "E-mail" containing "admin@condo.local" and "Senha" containing "....". To the right of the password field is an "eye" icon for password visibility. At the bottom left is a red "Entrar" button.

```
python security.py > ...
1 import bcrypt
2
3 def hash_pwd(plain: str) -> str:
4     return bcrypt.hashpw(plain.encode(), bcrypt.gensalt()).decode()
5
6 def check_pwd(plain: str, hashed: str) -> bool:
7     return bcrypt.checkpw(plain.encode(), hashed.encode())
```

# Aplicação (Síndico): Apartamentos

Olá, Gru (SINDICO)

Logout

Menu

- Apartamentos
- Moradores
- Despesas
- Notificações

## Apartamentos

_sa_instance_state	bloco	id	numero	andar
<sqlalchemy.orm.stateInstanceState object at 0x0000000000000000>	A	1	11	1
<sqlalchemy.orm.stateInstanceState object at 0x0000000000000000>	A	2	12	1
<sqlalchemy.orm.stateInstanceState object at 0x0000000000000000>	A	3	21	2
<sqlalchemy.orm.stateInstanceState object at 0x0000000000000000>	A	4	22	2

+ Novo Apartamento

Número

Bloco

Andar

0

- +

Criar

# Aplicação (Síndico): Moradores

Olá, Gru (SINDICO)

Logout

Menu

- Apartamentos
- Moradores**
- Despesas
- Notificações

## Moradores

ID	Nome	Email	Telefone	Papel	Apartamento
1	Gru	admin@condo.local	None	SINDICO	-
2	Kevin	Kevin@condo.local	(11) 91234-1234	MORADOR	11-A
3	Stuart	Stuart@condo.local	(11) 94321-4321	MORADOR	12-A
4	Bob	Bob@condo.local	(11) 92321-2321	MORADOR	11-A

+ Novo Morador

Nome  
Otto

Email  
Otto@condo.local

Telefone  
(11) 98888-2222

Senha  
....

# Aplicação (Síndico): Despesas

Olá, Gru (SINDICO)

Logout

Menu

- Apartamentos
- Moradores
- Despesas
- Notificações

## Despesas

ID	Descrição	Valor	Vencimento	Pago	Apartamento
1	Problema no Encanamento do Apto 11-A no Banheiro Social	350	2025-05-30	NAO	11-A
2	Apto 21-A Com Problema na Caixa de Força (Trocar Fiação)	1,153.28	2025-05-23	SIM	21-A

+ Registrar Despesa

Descrição  
Necessário trocar a privada do banheiro da suite

Valor (R\$)  
257,39 - +

Vencimento  
2025/06/04

Apartamento  
22-A ▼

Lançar

# Aplicação (Síndico): Relatório

Olá, Gru (SINDICO)

Logout

Menu

- Apartamentos
- Moradores
- Despesas
- Notificações

+ Registrar Despesa

▼

## Quitar Despesa

ID da despesa

2

Quitar agora

Exportar despesas

Formato

CSV  PDF

Gerar

Baixar

A	B	C	D	E	F	
1	ID	Descrição	Valor	Vencimento	Pago	Apartamento
2	1	Problema no Encanamento do Apto 11-A no Banheiro Social	350	5/30/2025	NAO	11
3	2	Apto 21-A Com Problema na Caixa de Força (Trocá-la) (Trocá-la)	1153.28	5/23/2025	SIM	21
4	3	Necessário trocar a privada do banheiro da suite	257.39	6/4/2025	NAO	22

### Relatório de Despesas – 27/05/2025

1 – Problema no Encanamento do Apto 11-A no Banheiro Social – R\$ 350.00 – 2025-05-30

2 – Apto 21-A Com Problema na Caixa de Força (Trocá-la) – R\$ 1153.28 – 2025-05-23

3 – Necessário trocar a privada do banheiro da suite – R\$ 257.39 – 2025-06-04

# Aplicação (Síndico): Notificação

Olá, Gru (SINDICO)

Logout

Menu

- Apartamentos
- Moradores
- Despesas
- Notificações

## Notificações

+ Enviar Notificação

Morador

Bob

Título

Festa Junina do Condomínio Dia 08/06/2025 (Domingo)!

Mensagem

Festa Junina do Condomínio Dia 08/06/2025 (Domingo)!  
Tragam Comidas e Venham a Caráter!

Enviar

## Histórico de notificações

ID	Morador	Título	Mensagem	Criada em	Lida?
5	Bob	Festa Junina do Condomínio Dia 08/06/2025 (Domingo)!	Festa Junina do Condomínio Dia 08/06/2025 (Domingo)! Tragam Comidas e Ve	27/05/2025 15:52	
4	Stuart	Festa Junina do Condomínio Dia 08/06/2025 (Domingo)!	Festa Junina do Condomínio Dia 08/06/2025 (Domingo)! Tragam Comidas e Ve	27/05/2025 15:52	

# Aplicação (Morador): Notificação

The screenshot shows the user interface of a mobile application for residents. On the left, a dark sidebar contains a profile icon with the text "Olá, Otto (MORADOR)", a "Logout" button, a "Menu" button, and a "Notificações" button which is currently selected. The main content area has a title "Notificações" with a bell icon. Below it is a table listing two notifications:

ID	Título	Mensagem	Criada em	Lida
2	Cotação realizada para trocar a caixa de Força do Apto 21-A	Cotação realizada para trocar a caixa de força. Valor total de: R\$ 1.153,28	2025-05-27 15:51:05	SIM
1	Inspeção realizada na caixa de Força do Apto 21-A	Inspeção realizada na caixa de Força do Apto 21-A feita pela técnica Margo. Necessário	2025-05-27 15:50:00	NAO

Below the table, there is a text input field labeled "ID para marcar lida" with the value "2" and a "+" button.

```
180     else:
181         notas = db.query(Notificacao).filter(Notificacao.morador == user).order_by(Notificacao.criada_em.desc()).all()
182         st.dataframe([
183             "ID": n.id,
184             "Título": n.titulo,
185             "Mensagem": n.mensagem,
186             "Criada em": n.criada_em,
187             "Lida": n.lida
188         } for n in notas], hide_index=True, use_container_width=True)
189
190         not_id = st.number_input("ID para marcar lida", step=1, format="%d")
191         if st.button("Marcar como lida"):
192             n = db.get(Notificacao, not_id)
193             if n and n.morador_id == user.id:
194                 n.lida = "SIM"; db.commit(); st.experimental_rerun()
```

# Resumo das Funcionalidades Implementadas

Módulo	Funcionalidade	Status
Apartamentos	CRUD Completo	✓
Moradores	CRUD + Papéis (Morador/Síndico)	✓
Despesas	Lançar, Listar, Quitar	✓
Notificações	Envio (Síndico) → Inbox (Morador)	✓
Relatórios	Exportar CSV / PDF da tela de despesas	✓
Autenticação + Sessão	Login, Logou e uso do BCrypt para autenticação	✓
Controle de Acesso	Tela restrita ao papel do usuário (Morador/Síndico)	✓

# Segurança e Boas Práticas

- Senhas armazenadas e protegidas com BCrypt
- Uso do SQLAlchemy ORM em Camadas com queries parametrizadas que evitam SQL Injection
- Uso de FK, UNIQUE e ENUM, que garantem coerência dos dados no próprio Banco de Dados;
- Conexões fechadas automaticamente, utilizando o DB apenas quando necessário “with get\_db() as db:”.

```
models.py > Morador
1  from sqlalchemy import Column, Integer, String, DECIMAL, Date, DateTime, Enum, ForeignKey, Text, func
2  from sqlalchemy.orm import relationship
3  from db import Base
4
5  class Apartamento(Base):
6      __tablename__ = "apartamento"
7      id = Column(Integer, primary_key=True)
8      numero = Column(String(10), nullable=False)
9      bloco = Column(String(10))
10     andar = Column(Integer)
11     moradores = relationship("Morador", back_populates="apartamento", cascade="all, delete")
12     despesas = relationship("Despesa", back_populates="apartamento", cascade="all, delete")
13
14 class Morador(Base):
15     __tablename__ = "morador"
16     id = Column(Integer, primary_key=True)
17     nome = Column(String(100), nullable=False)
18     email = Column(String(120), unique=True)
19     telefone = Column(String(20))
20     senha_hash = Column(String(256))
21     papel = Column(Enum("SINDICO", "MORADOR"), default="MORADOR")
22     apartamento_id = Column(Integer, ForeignKey("apartamento.id"))
23
24 # Helpers
25 @contextmanager
26 def get_db():
27     db = SessionLocal()
28     try:
29         yield db
30     finally:
31         db.close()
32
33 # Auth
34 def login_user(db, email, pwd):
35     user = db.query(Morador).filter(Morador.email == email).first()
36     if user and check_pwd(pwd, user.senha_hash):
37         return user
```

# Ideias para Versões Futuras



- Reservas de Áreas Comuns
- Integração com Gateway de Pagamento
- Aplicativo Mobile (Próxima Disciplina?)
- Dashboard analíticos utilizando Matplotlib e Seaborn
- Módulo de assembleias on-line com votação

# O que foi entregue

- Modelo relacional Normalizado, com 5 entidades, FKs, índices e regras de integridade;
- Scripts DDL documentados → para criar, derrubar e recriar todo o schema;
- Consultas CRUD + transações DML (SQLAlchemy ORM);
- Integração completa com código fonte:
  - MySQL 8  $\rightleftarrows$  Python (ORM + Bcrypt)  $\rightleftarrows$  Streamlit (Web);
- Relatórios em CSV/PDF direto do BD (pandas + ReportLab);
- Boas práticas adotadas, como: senhas com salt (bcrypt), princípio do mínimo privilégio, consultas parametrizadas (anti-injeção);
- Requisitos funcionais e não funcionais atendidos para uma aplicação de gestão condominial, com: cadastro de aptos e moradores, controle de despesas e pagamentos e notificações direcionadas.

# Obrigado!

**Contato**

**Leonardo Grupioni**

RA00319703

✉️ [leofgrupioni@gmail.com](mailto:leofgrupioni@gmail.com)

🌐 <https://www.linkedin.com/in/leonardo-grupioni/>