STRESS AI

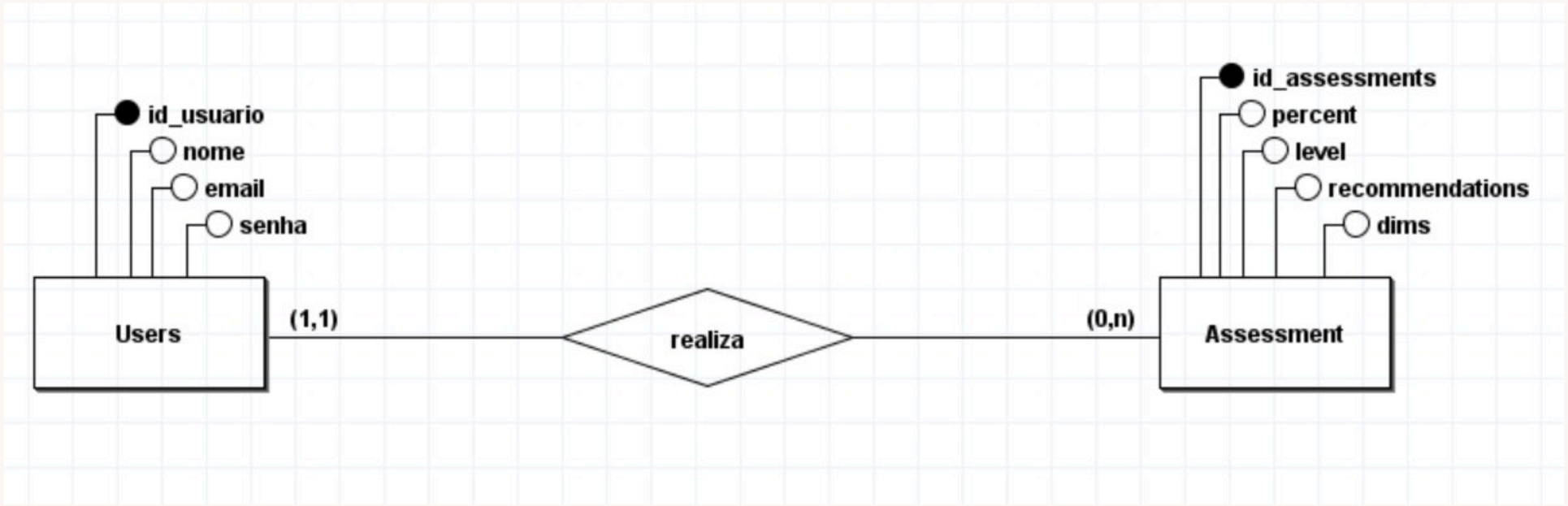
Guilherme Almeida, Lucas Castilho, Matheus Hirata

Desenvolvimento Web 2025.2 - UTFPR Campus Apucarana

OBJETIVOS

Desenvolver uma aplicação web capaz de avaliar o nível de estresse de estudantes por meio de um questionário interativo e fornecer recomendações personalizadas com o auxílio de inteligência artificial, promovendo autoconhecimento e apoio à saúde mental.

MODELAGEM BD



1 – INTEGRAÇÃO MAIS COMPLEXA

Modelagem do banco + Relacionamentos

- Definição das tabelas (ORM) com SQLModel
- Tipos de dados
- Campos opcionais
- primary key
- unique index
- foreign key
- Relacionamentos 1:N entre User ⇄ Assessments
- Integração com o session engine
- As rotas irão depender do formato dessas tabelas.
- Campos calculados e serialização
- O campo dims (armazenado como JSON string).
- Campo created_at com default_factory.
- Camada de segurança
- email único
- hashed_password
- scopes para permissões

```python
from sqlmodel import SQLModel, Field, Relationship
from datetime import datetime
from typing import Optional, List

# MODELO DE USUÁRIO
class UserModel(SQLModel, table=True):
    __tablename__ = "users"

    id: Optional[int] = Field(default=None, primary_key=True)
    name: str
    email: str = Field(index=True, unique=True)
    hashed_password: str
    disabled: bool = Field(default=False)
    scopes: str

    assessments: List["AssessmentModel"] = Relationship(back_populates="user")

# MODELO DE AVALIAÇÃO
class AssessmentModel(SQLModel, table=True):
    __tablename__ = "assessments"

    id: Optional[int] = Field(default=None, primary_key=True)
    user_id: int = Field(foreign_key="users.id")
    created_at: datetime = Field(default_factory=datetime.utcnow)

    percent: float
    level: str
    recommendations: str

    # guardamos dims (dimensões calculadas) como JSON string
    dims: str

    user: Optional[UserModel] = Relationship(back_populates="assessments")
```

2 – INTEGRAÇÃO MAIS COMPLEXA

Criar as tabelas no SQLite + Usuário ADM

- Criação automática das tabelas usando SQLModel.metadata.create_all(engine)
- Integração do metadata global com o engine configurado
- Inicialização completa da estrutura do banco sem migrations
- Abertura manual de sessão interna com Session(engine)
- Execução de consulta para verificar existência do usuário admin
- Inserção automática do usuário administrador na primeira execução
- Hash seguro da senha do admin usando Argon2
- Commit manual da operação de seed no banco
- Garantia de idempotência (não duplica admin ao rodar novamente)
- Preparação do ambiente de banco antes do servidor FastAPI iniciar

```python
from sqlmodel import SQLModel, Session, select
from app.db.session import engine
from app.db.models import UserModel
from app.db.session import get_session
from app.db.session import engine
from passlib.hash import argon2


def init_db():
    """Cria as tabelas no SQLite se não existirem."""
    SQLModel.metadata.create_all(engine)


def seed_admin():
    """Cria um admin inicial se não existir."""
    with Session(engine) as session:
        stmt = select(UserModel).where(UserModel.email == "admin@example.com")
        existing = session.exec(stmt).first()

        if existing:
            return

        admin = UserModel(
            name="Admin",
            email="admin@example.com",
            hashed_password=argon2.hash("admin123"),
            disabled=False,
            scopes="admin"
        )

        session.add(admin)
        session.commit()
```