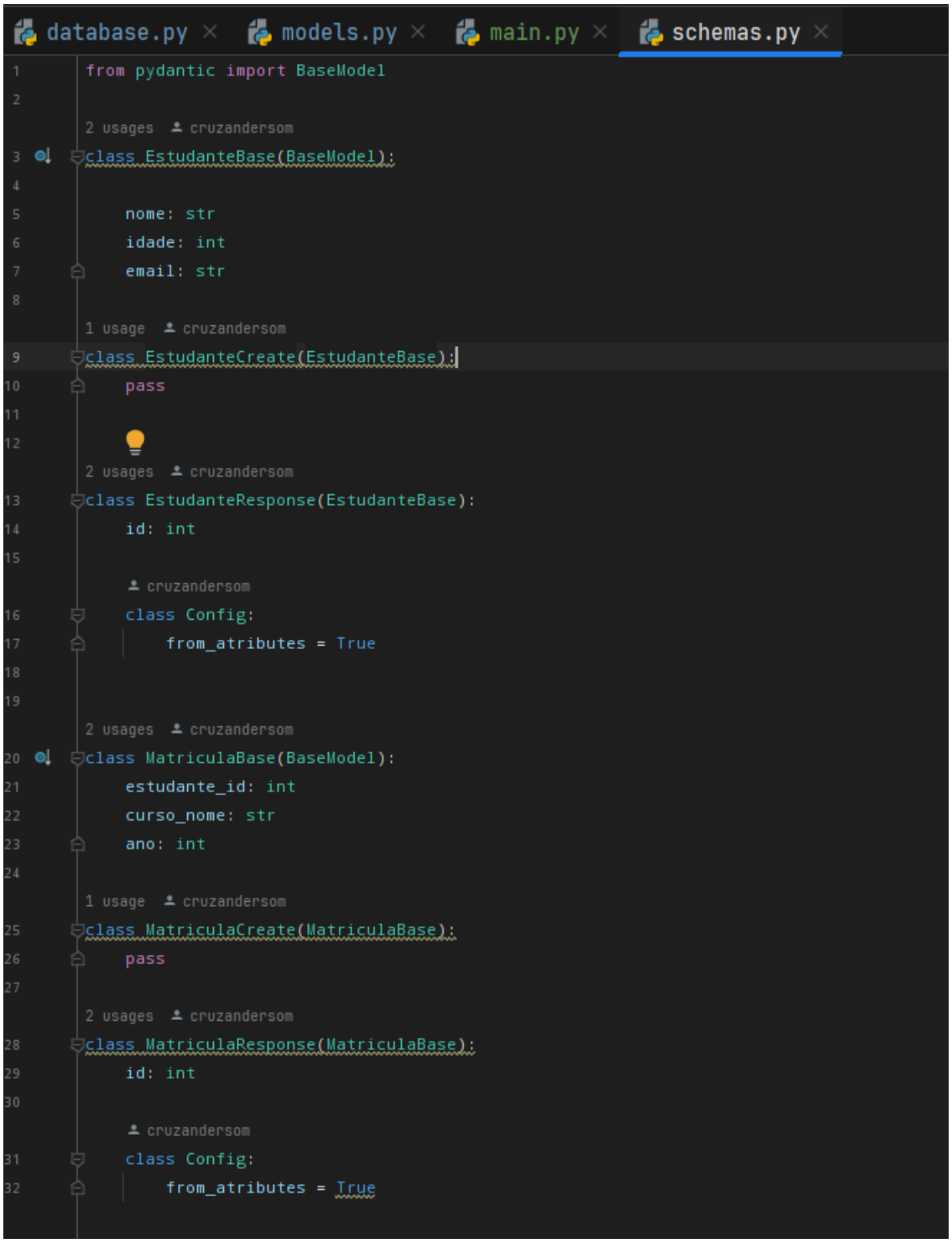
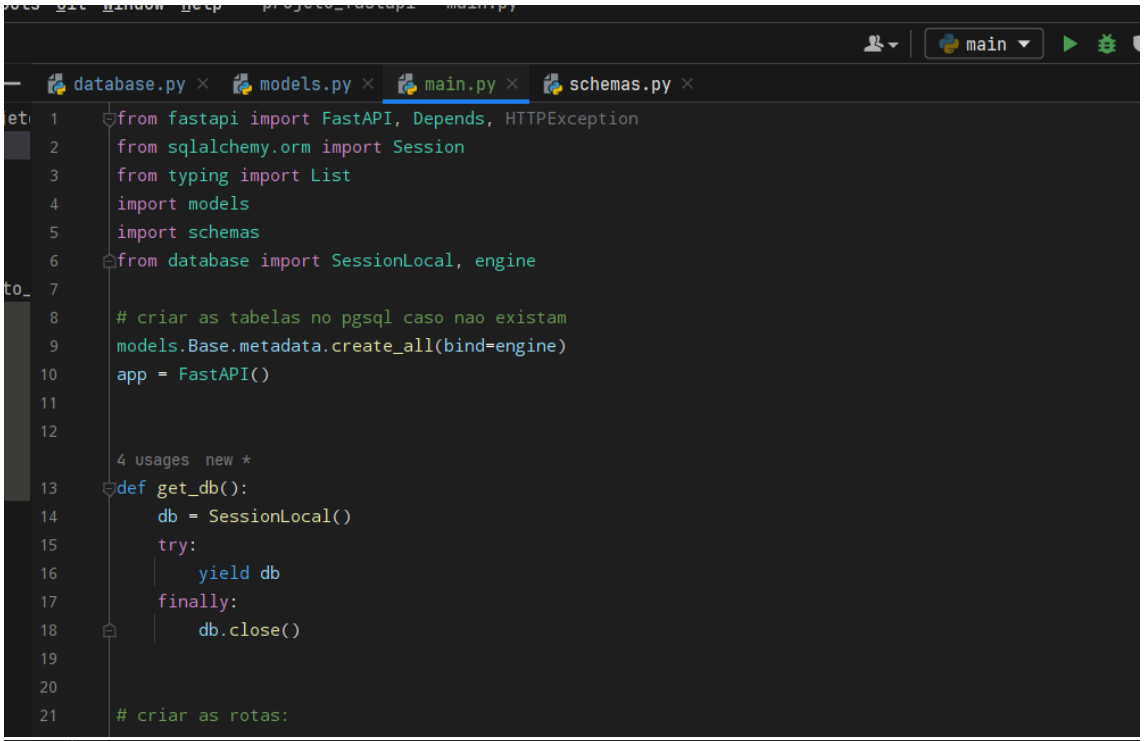
database.py

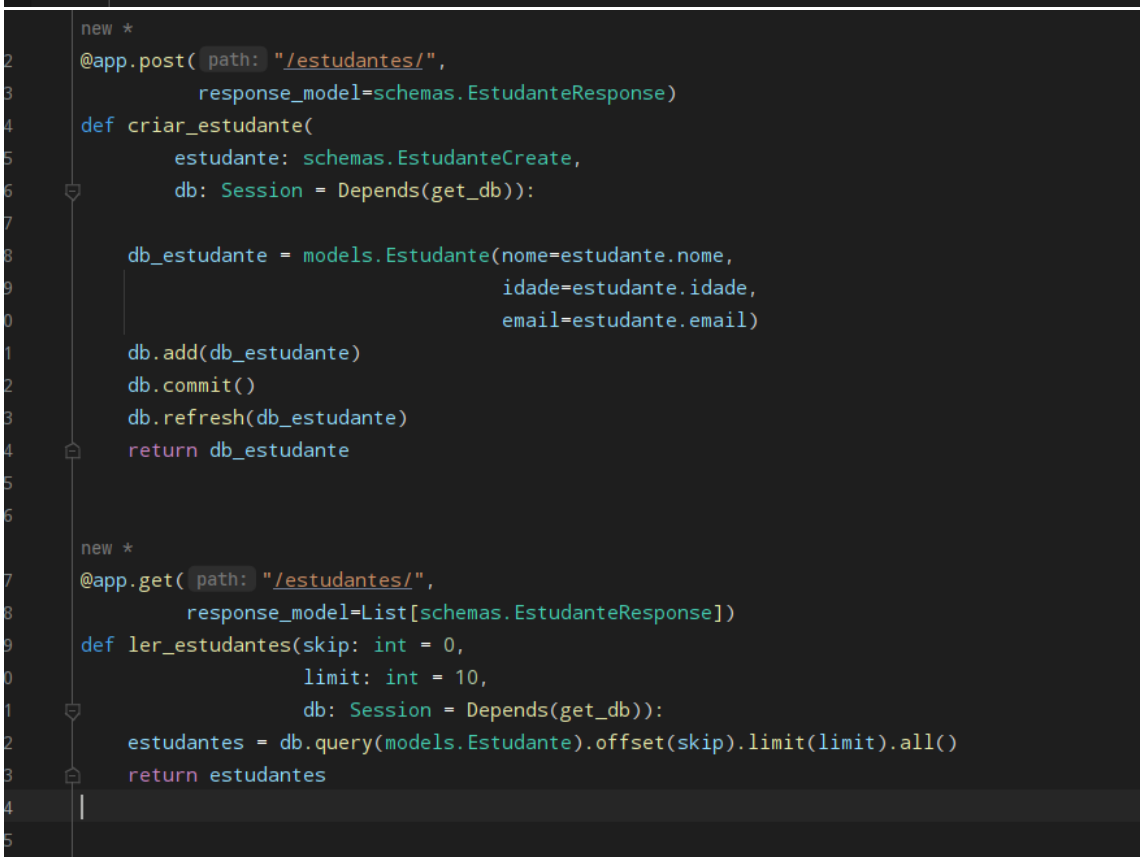
```python
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

DATABASE_URL = "postgresql://postgres:144233377@localhost/escola"

engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()
```

models.py

```python
from sqlalchemy import \
    Column, Integer, String, ForeignKey

from database import Base


2 usages  cruzandersom
class Estudante(Base):
    __tablename__ = 'estudantes'

    id = Column(Integer, primary_key=True, index=True)
    nome = Column(String, index=True)
    idade = Column(Integer)
    email = Column(String, unique=True, index=True)


2 usages  cruzandersom *
class Matricula(Base):
    __tablename__ = 'matriculas'

    id = Column(Integer, primary_key=True, index=True)
    estudante_id = Column(Integer, ForeignKey('estudantes.id'), nullable=False)
    curso_nome = Column(String, index=True)
    ano = Column(Integer)
```


database.py
models.py
main.py
schemas.py
from pydantic import BaseModel
2 usages cruzandersom
class EstudanteBase(BaseModel):
nome: str
idade: int
email: str
1 usage cruzandersom
class EstudanteCreate(EstudanteBase):
pass
2 usages cruzandersom
class EstudanteResponse(EstudanteBase):
id: int
cruzandersom
class Config:
from_atributes = True
2 usages cruzandersom
class MatriculaBase(BaseModel):
estudante_id: int
curso_nome: str
ano: int
1 usage cruzandersom
class MatriculaCreate(MatriculaBase):
pass
2 usages cruzandersom
class MatriculaResponse(MatriculaBase):
id: int
cruzandersom
class Config:
from_atributes = True


main
database.py
models.py
main.py
schemas.py
from fastapi import FastAPI, Depends, HTTPException
from sqlalchemy.orm import Session
from typing import List
import models
import schemas
from database import SessionLocal, engine
# criar as tabelas no pgsql caso nao existam
models.Base.metadata.create_all(bind=engine)
app = FastAPI()
4 usages new *
def get_db():
db = SessionLocal()
try:
yield db
finally:
db.close()
# criar as rotas:


new *
@app.post(path: "/estudantes/",
response_model=schemas.EstudanteResponse)
def criar_estudante(
estudante: schemas.EstudanteCreate,
db: Session = Depends(get_db)):
db_estudante = models.Estudante(nome=estudante.nome,
idade=estudante.idade,
email=estudante.email)
db.add(db_estudante)
db.commit()
db.refresh(db_estudante)
return db_estudante
new *
@app.get(path: "/estudantes/",
response_model=List[schemas.EstudanteResponse])
def ler_estudantes(skip: int = 0,
limit: int = 10,
db: Session = Depends(get_db)):
estudantes = db.query(models.Estudante).offset(skip).limit(limit).all()
return estudantes

```python
new *
@app.post( path: "/matriculas/",
          response_model=schemas.MatriculaResponse)
def criar_matricula(
        matricula: schemas.MatriculaCreate,
        db: Session = Depends(get_db)):
    db_matricula = models.Matricula(estudante_id=matricula.estudante_id,
                                    curso_nome=matricula.curso_nome,
                                    ano=matricula.ano)
    db.add(db_matricula)
    db.commit()
    db.refresh(db_matricula)
    return db_matricula


new *
@app.get( path: "/matriculas/",
         response_model=List[schemas.MatriculaResponse])
def ler_matriculas(skip: int = 0,
                   limit: int = 10,
                   db: Session = Depends(get_db)):
    matriculas = db.query(models.Matricula).offset(skip).limit(limit).all()
    return matriculas
```