INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO

PROJETO DE BLOCO FUNDAMENTOS DE DADOS

TESTE DE PERFORMANCE – TP04

ALUNO: GABRIEL GOMES DE SOUZA

PROFESSOR(A): LUIZ PAULO BOMENY MAIA

E-MAIL: gabriel.gsouza@al.infnet.edu.br

Sumário

1. Conteúdo

main.py

```
import pandas as pd
from os.path import exists
from models import Livro, Autor, conectar_bd,
desconectar_bd

# Verificar se o banco de dados já existe
db_exists = exists('biblioteca.db')

# Conectar ao banco de dados
session = conectar_bd()

if db_exists:
    # Leitura dos arquivos CSV
    autores_df =
pd.read_csv('TP04/Dados_CSV/autores.csv', delimiter=';',
header=None, names=['id', 'nome', 'sobrenome'])
    livros_df = pd.read_csv('TP04/Dados_CSV/livros.csv',
delimiter=';', header=None, names=['id', 'titulo',
'data_lancamento', 'preco'])
    livros_autores_df =
pd.read_csv('TP04/Dados_CSV/livros-autores.csv',
delimiter=';', header=None, names=['id_livro',
'id_autor'])

    # Tratamento da data de lançamento
    livros_df['data_lancamento'] =
pd.to_datetime(livros_df['data_lancamento'],
format='%d/%m/%Y').dt.strftime('%Y-%m-%d')

    # Inserir autores
    for _, row in autores_df.iterrows():
        autor = Autor(id_autor=row['id'],
```

```
nome=row['nome'], sobrenome=row['sobrenome'])
            session.add(autor)

        # Inserir livros
        for _, row in livros_df.iterrows():
            livro = Livro(id_livro=row['id'],
titulo=row['titulo'],
data_lancamento=row['data_lancamento'],
preco=row['preco'])
            session.add(livro)

        # Commit para salvar os dados de autores e livros
        session.commit()

        # Inserir relacionamentos muitos para muitos
        for _, row in livros_autores_df.iterrows():
            livro = session.query(Livro).get(row['id_livro'])
            autor = session.query(Autor).get(row['id_autor'])
            livro.autores.append(autor)

        # Commit para salvar os relacionamentos
        session.commit()

def consultar_autores_por_livro(titulo_livro):
    """
    Consulta os autores de um livro específico.

    Args:
        titulo_livro (str): O título do livro a ser
consultado.

    Returns:
        pandas.DataFrame: DataFrame contendo os nomes e
sobrenomes dos autores do livro.
    """
    query = session.query(Livro).filter(Livro.titulo ==
```

```
titulo_livro).first()
    if query:
        return pd.DataFrame([{'nome': autor.nome,
'sobrenome': autor.sobrenome} for autor in
query.autores])
    else:
        return pd.DataFrame()

def consultar_livros_por_autor(nome_autor,
sobrenome_autor):
    """
    Consulta os livros de um autor específico.

    Args:
        nome_autor (str): O nome do autor a ser
consultado.
        sobrenome_autor (str): O sobrenome do autor a ser
consultado.

    Returns:
        pandas.DataFrame: DataFrame contendo os títulos,
datas de lançamento e preços dos livros do autor.
    """
    query = session.query(Autor).filter(Autor.nome ==
nome_autor, Autor.sobrenome == sobrenome_autor).first()
    if query:
        return pd.DataFrame([{'titulo': livro.titulo,
'data_lancamento': livro.data_lancamento, 'preco':
livro.preco} for livro in query.livros])
    else:
        return pd.DataFrame()

# Consultas e geração dos arquivos JSON
autores_df = consultar_autores_por_livro('Livro L1')
autores_df.to_json('autores_livro_l1.json',
orient='records', indent=4)
```

```
livros_df = consultar_livros_por_autor('João', 'Maia')
livros_df.to_json('livros_joao_maia.json',
orient='records', indent=4)

# Fechar sessão
desconectar_bd(session)
```

models.py

```
from sqlalchemy import Column, Integer, String, Date,
Float, ForeignKey, Table, create_engine
from sqlalchemy.orm import relationship, declarative_base,
sessionmaker

Base = declarative_base()

# Associação muitos para muitos
livro_autor = Table('livro_autor', Base.metadata,
    Column('id_livro', Integer,
ForeignKey('livro.id_livro')),
    Column('id_autor', Integer,
ForeignKey('autor.id_autor'))
)

class Livro(Base):
    """
    Classe que representa a tabela 'livro' no banco de
dados.

    Atributos:
        id_livro (int): ID do livro.
        titulo (str): Título do livro.
        data_lancamento (date): Data de lançamento do
livro.
        preco (float): Preço do livro.
        autores (list): Lista de autores associados ao
```

```
livro.
    """
    __tablename__ = 'livro'
    id_livro = Column(Integer, primary_key=True)
    titulo = Column(String(100), nullable=False)
    data_lancamento = Column(Date, nullable=False)
    preco = Column(Float, nullable=False)
    autores = relationship('Autor', secondary=livro_autor,
back_populates='livros')

class Autor(Base):
    """
    Classe que representa a tabela 'autor' no banco de
dados.

    Atributos:
        id_autor (int): ID do autor.
        nome (str): Nome do autor.
        sobrenome (str): Sobrenome do autor.
        livros (list): Lista de livros associados ao autor.
    """
    __tablename__ = 'autor'
    id_autor = Column(Integer, primary_key=True)
    nome = Column(String(50), nullable=False)
    sobrenome = Column(String(50), nullable=False)
    livros = relationship('Livro', secondary=livro_autor,
back_populates='autores')

def conectar_bd():
    """
    Conecta ao banco de dados SQLite 'biblioteca.db' e
retorna uma sessão.

    Returns:
        Session: Sessão do SQLAlchemy para interagir com o
banco de dados.
    """
    engine = create_engine('sqlite:///biblioteca.db')
```

```
    Base.metadata.create_all(engine)
    Session = sessionmaker(bind=engine)
    return Session()

def desconectar_bd(session):
    """
    Fecha a sessão do banco de dados.

    Args:
        session (Session): Sessão do SQLAlchemy a ser
fechada.
    """
    session.close()
```