1. **Описать структуру базы данных;**

Структура базы данных, основанная на предоставленных моделях Python (SQLAlchemy ORM), включает в себя три таблицы: users, chats и messages.

1. Таблица users:

Хранит информацию о зарегистрированных пользователях системы.

Столбцы:

* + id (Integer, PRIMARY KEY): Уникальный идентификатор пользователя. Автоинкрементное целое число.
  + username (String, UNIQUE, NOT NULL): Имя пользователя. Должно быть уникальным и не может быть пустым.
  + email (String, UNIQUE, NOT NULL): Адрес электронной почты пользователя. Должен быть уникальным и не может быть пустым.
  + password\_hash (String, NOT NULL): Хэш пароля пользователя. Важно: Хранить пароли в открытом виде категорически запрещено. Здесь должен храниться только хэш.

2. Таблица chats:

Хранит информацию о чатах, существующих в системе.

Столбцы:

* + id (Integer, PRIMARY KEY): Уникальный идентификатор чата. Автоинкрементное целое число.
  + name (String, NOT NULL): Название чата. Не может быть пустым.

3. Таблица messages:

Хранит сообщения, отправленные в чатах.

Столбцы:

* + id (Integer, PRIMARY KEY): Уникальный идентификатор сообщения. Автоинкрементное целое число.
  + chat\_id (Integer, FOREIGN KEY -> chats.id): Идентификатор чата, к которому относится сообщение. Является внешним ключом, ссылающимся на столбец id в таблице chats.
  + sender\_id (Integer, FOREIGN KEY -> users.id): Идентификатор пользователя, отправившего сообщение. Является внешним ключом, ссылающимся на столбец id в таблице users.
  + content (String, NOT NULL): Текст сообщения. Не может быть пустым.
  + timestamp (DateTime, DEFAULT datetime.utcnow): Дата и время отправки сообщения. По умолчанию устанавливается текущее время в UTC.

1. **Скрипты**

from sqlalchemy.orm import sessionmaker

from sqlalchemy import create\_engine

from models import User, Chat, Message

from sqlalchemy import Column, Integer, String, ForeignKey, DateTime

from sqlalchemy.orm import relationship

from sqlalchemy.ext.declarative import declarative\_base

from datetime import datetime

from faker import Faker

faker = Faker()

DATABASE\_URL = "postgresql://user:password@localhost:5432/messenger"

engine = create\_engine(DATABASE\_URL)

Session = sessionmaker(bind=engine)

session = Session()

Base = declarative\_base()

# Таблица пользователей

class User(Base):

    \_\_tablename\_\_ = 'users'

    id = Column(Integer, primary\_key=True, autoincrement=True)

    username = Column(String, unique=False, nullable=False)

    email = Column(String, unique=True, nullable=False)

    password\_hash = Column(String, nullable=False)

# Таблица чатов

class Chat(Base):

    \_\_tablename\_\_ = 'chats'

    id = Column(Integer, primary\_key=True, autoincrement=True)

    name = Column(String, nullable=False)

# Таблица сообщений

class Message(Base):

    \_\_tablename\_\_ = 'messages'

    id = Column(Integer, primary\_key=True, autoincrement=True)

    chat\_id = Column(Integer, ForeignKey('chats.id'))

    sender\_id = Column(Integer, ForeignKey('users.id'))

    content = Column(String, nullable=False)

    timestamp = Column(DateTime, default=datetime.utcnow)

    chat = relationship("Chat")

    sender = relationship("User")

Base.metadata.create\_all(engine)

'''with engine.connect() as conn:

    conn.execute(text("ALTER SEQUENCE users\_id\_seq RESTART WITH 1;"))

    conn.execute(text("ALTER SEQUENCE chats\_id\_seq RESTART WITH 1;"))

    conn.execute(text("ALTER SEQUENCE messages\_id\_seq RESTART WITH 1;"))

    conn.commit()'''

# Создание пользователей

user1 = User(username=faker.name(), email=faker.email(), password\_hash="hashed\_password")

user2 = User(username=faker.name(), email=faker.email(), password\_hash="hashed\_password")

# Создание чатов

chat1 = Chat(name=faker.text())

chat2 = Chat(name=faker.text())

# Добавление и коммит пользователей и чатов (чтобы получить их id)

session.add\_all([user1, user2, chat1, chat2])

session.commit()  # Важно: commit нужен для получения id!

# Создание сообщений

message1 = Message(chat\_id=chat1.id, sender\_id=user1.id, content=faker.words(6))

message2 = Message(chat\_id=chat1.id, sender\_id=user2.id, content=faker.words(6))

message3 = Message(chat\_id=chat2.id, sender\_id=user1.id, content=faker.words(6))

message4 = Message(chat\_id=chat2.id, sender\_id=user2.id, content=faker.words(6))

# Добавление сообщений

session.add\_all([message1, message2, message3, message4])

session.commit() # Закоммитить сообщения после добавления

# Чтение данных (сообщений из общего чата)

print("\nСообщения из общего чата:")

for message in session.query(Message).filter(Message.chat\_id == chat1.id).all():

    sender = session.query(User).filter(User.id == message.sender\_id).first() # Retrieve sender from the user table

    print(f"  {sender.username}: {message.content} (at {message.timestamp})") # Use sender.username

# Чтение данных (сообщений из разговора John & Jane)

print("\nСообщения из личного чата John & Jane:")

for message in session.query(Message).filter(Message.chat\_id == chat2.id).all():

    sender = session.query(User).filter(User.id == message.sender\_id).first() # Retrieve sender from the user table

    print(f"  {sender.username}: {message.content} (at {message.timestamp})") # Use sender.username

# Печать списка пользователей в каждом чате

print("\nПользователи в чате:", chat1.name)

users\_in\_chat1 = session.query(User).join(Message).filter(Message.chat\_id == chat1.id).distinct().all()

for user in users\_in\_chat1:

    print(f"  {user.username}")

print("\nПользователи в чате:", chat2.name)

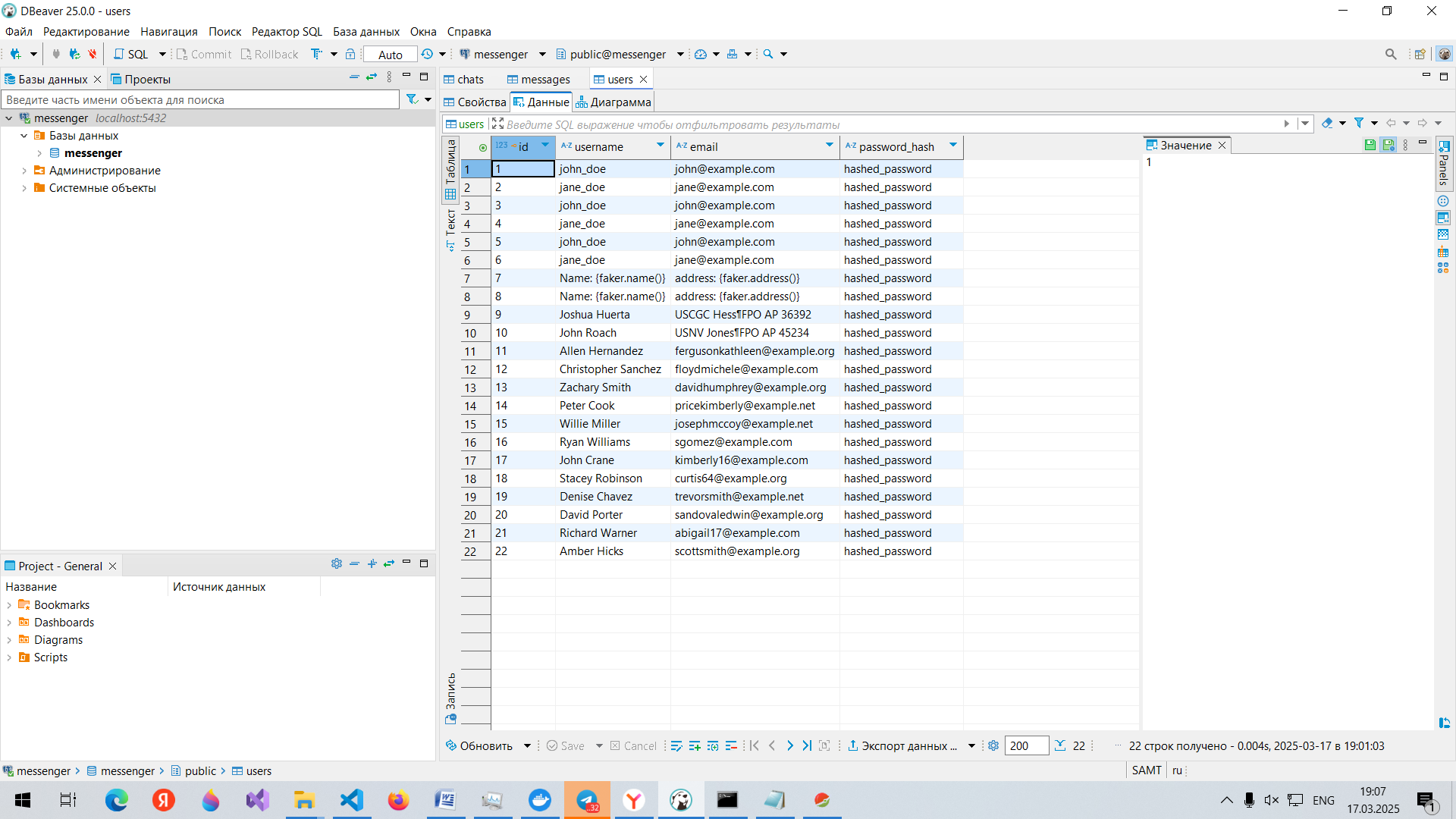
users\_in\_chat2 = session.query(User).join(Message).filter(Message.chat\_id == chat2.id).distinct().all()

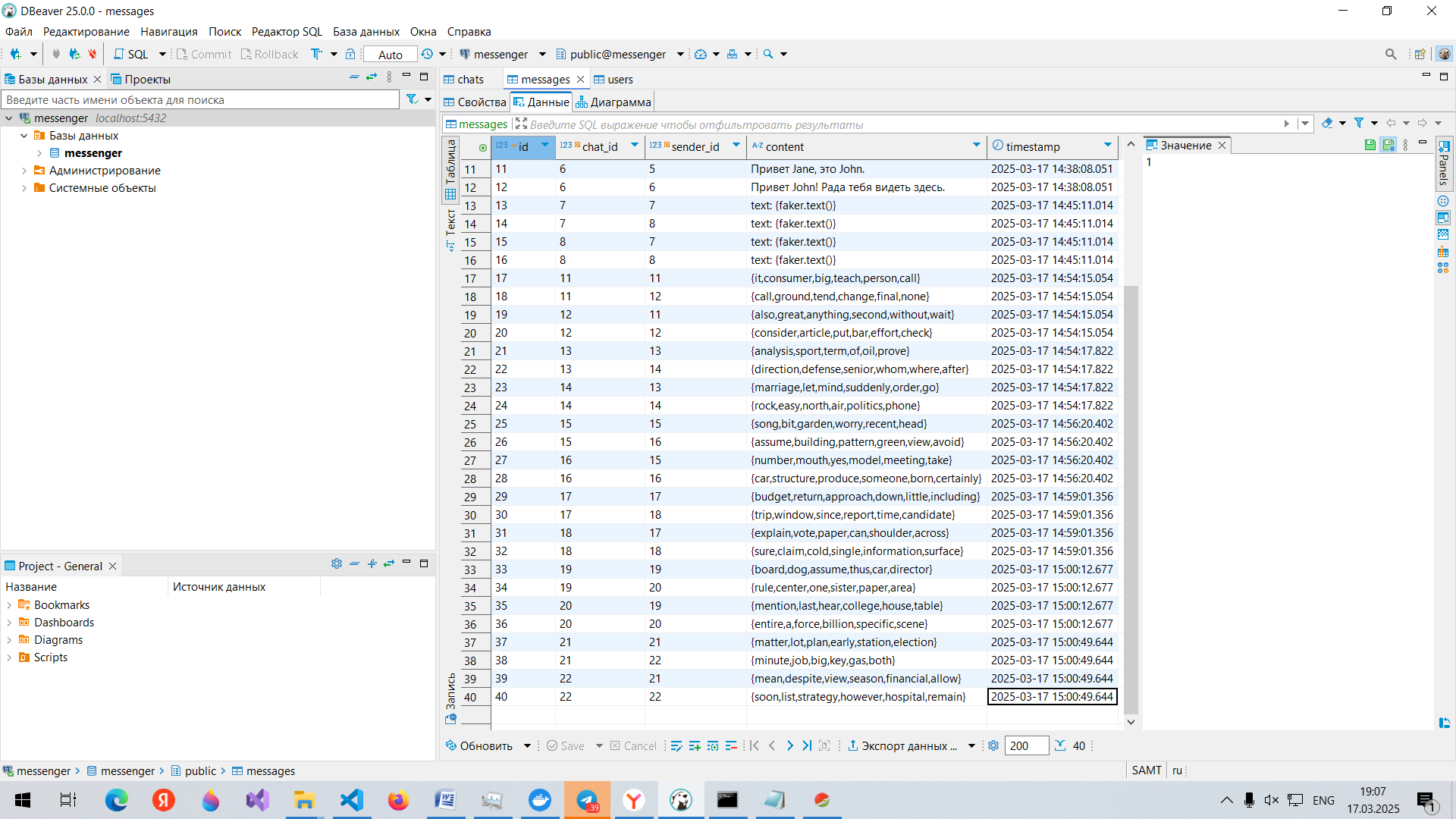
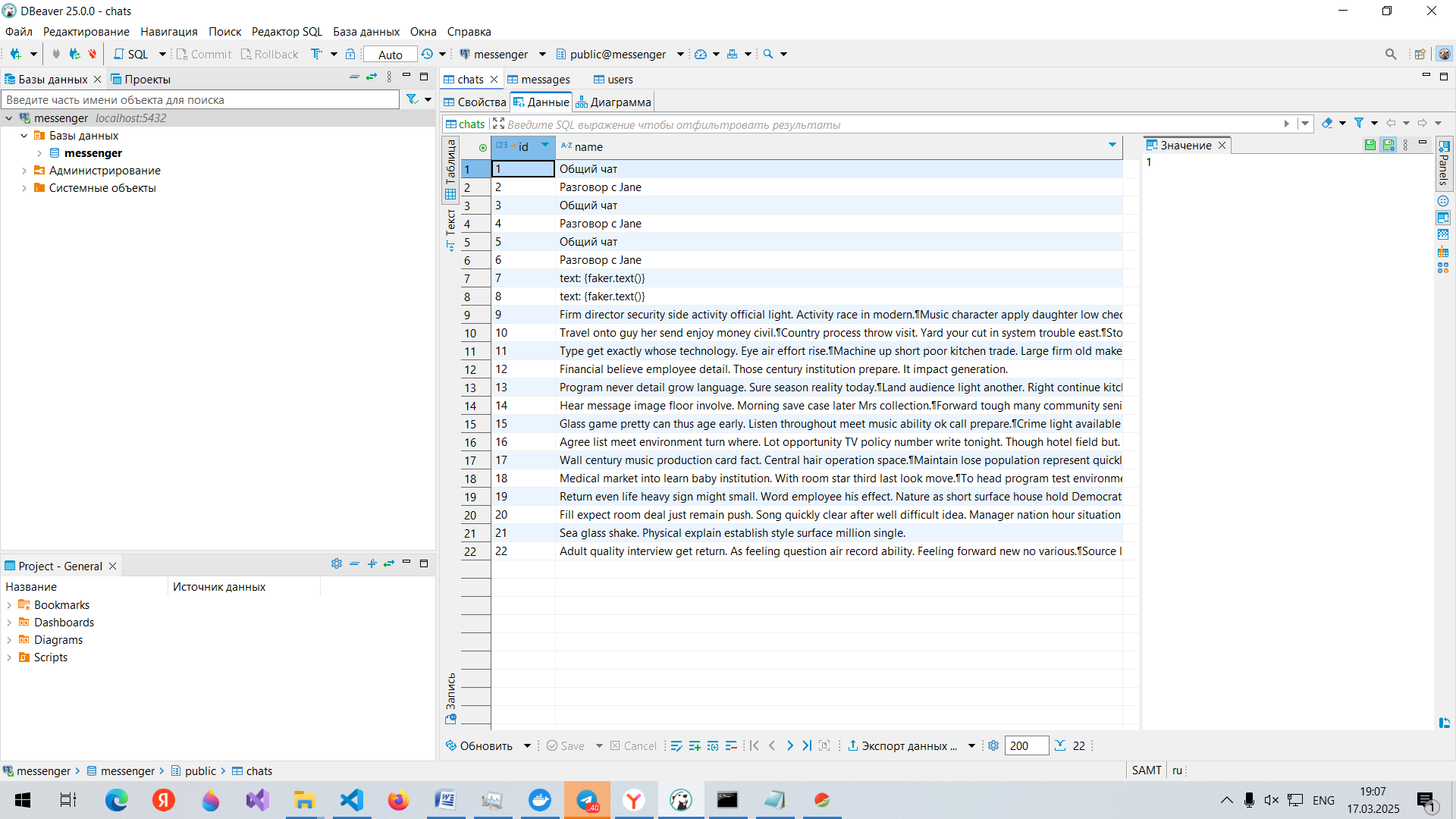
for user in users\_in\_chat2:

    print(f"  {user.username}")

session.close()

Результат:





Сообщения из общего чата:

Catherine Miller: {wait,heart,to,air,prepare,whose} (at 2025-03-17 15:06:30.790023)

Juan Benson: {sport,near,evening,success,admit,hour} (at 2025-03-17 15:06:30.790023)

Сообщения из личного чата John & Jane:

Catherine Miller: {individual,old,dog,necessary,him,power} (at 2025-03-17 15:06:30.790535)

Juan Benson: {ahead,hard,matter,seem,year,executive} (at 2025-03-17 15:06:30.790535)

Пользователи в чате: Big else economy not key dream us. Painting check front maintain. Third once firm family floor parent.

Even race represent but.

Difference PM dream build.

Catherine Miller

Juan Benson

Пользователи в чате: No trade light car trouble. Stop hot choose. Choice home suffer describe.

Consumer eye at. Scientist eat offer participant might yourself could.

Catherine Miller

Juan Benson

1. **Настроены миграции (Alembic):**

alembic init migrations

sqlalchemy.url = postgresql://user:password@localhost:5432/messenger

alembic revision --autogenerate -m "Initial migration"

alembic upgrade head