Лабораторная работа №2

Работа с SQLAlchemy и Alembic

Студент(ка): Зырянова Татьяна Евгеньевна

Группа: РИМ – 150950

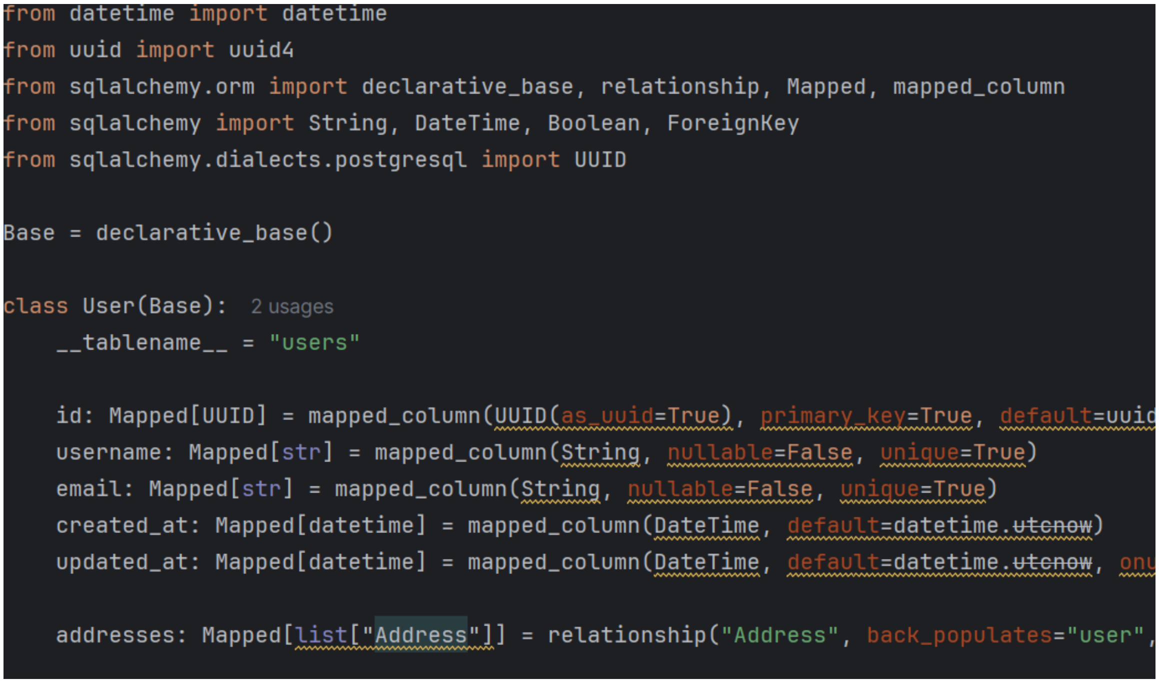
Предмет: Разработка приложений

Цель работы

Освоить принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучить механизмы миграции базы данных.

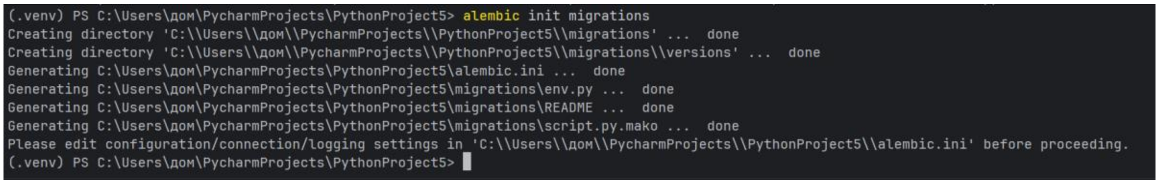
Ход работы

1) Создание ОРМ для пользователя



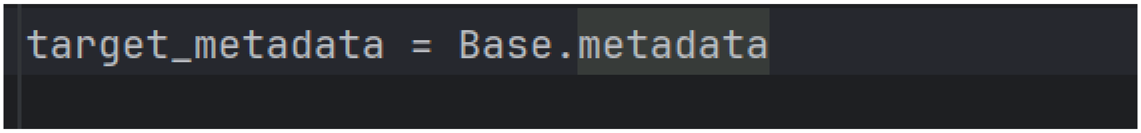
```
from datetime import datetime
from uuid import uuid4
from sqlalchemy.orm import declarative_base, relationship, Mapped, mapped_column
from sqlalchemy import String, DateTime, Boolean, ForeignKey
from sqlalchemy.dialects.postgresql import UUID

Base = declarative_base()

class User(Base):  2 usages
    __tablename__ = "users"

    id: Mapped[UUID] = mapped_column(UUID(as_uuid=True), primary_key=True, default=uuid
    username: Mapped[str] = mapped_column(String, nullable=False, unique=True)
    email: Mapped[str] = mapped_column(String, nullable=False, unique=True)
    created_at: Mapped[datetime] = mapped_column(DateTime, default=datetime.utcnow)
    updated_at: Mapped[datetime] = mapped_column(DateTime, default=datetime.utcnow, onu

    addresses: Mapped[list["Address"]] = relationship("Address", back_populates="user",
```

2) Инициализация миграций



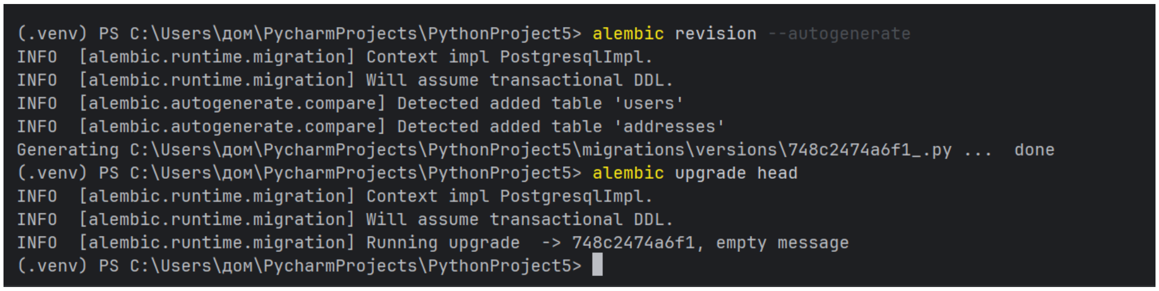
```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> alembic init migrations
Creating directory 'C:\\Users\\дом\\PycharmProjects\\PythonProject5\\migrations' ...  done
Creating directory 'C:\\Users\\дом\\PycharmProjects\\PythonProject5\\migrations\\versions' ...  done
Generating C:\Users\дом\PycharmProjects\PythonProject5\alembic.ini ...  done
Generating C:\Users\дом\PycharmProjects\PythonProject5\migrations\env.py ...  done
Generating C:\Users\дом\PycharmProjects\PythonProject5\migrations\README ...  done
Generating C:\Users\дом\PycharmProjects\PythonProject5\migrations\script.py.mako ...  done
Please edit configuration/connection/logging settings in 'C:\\Users\\дом\\PycharmProjects\\PythonProject5\\alembic.ini' before proceeding.
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5>
```

3) В env.py подтсавляем метаданные для миграции



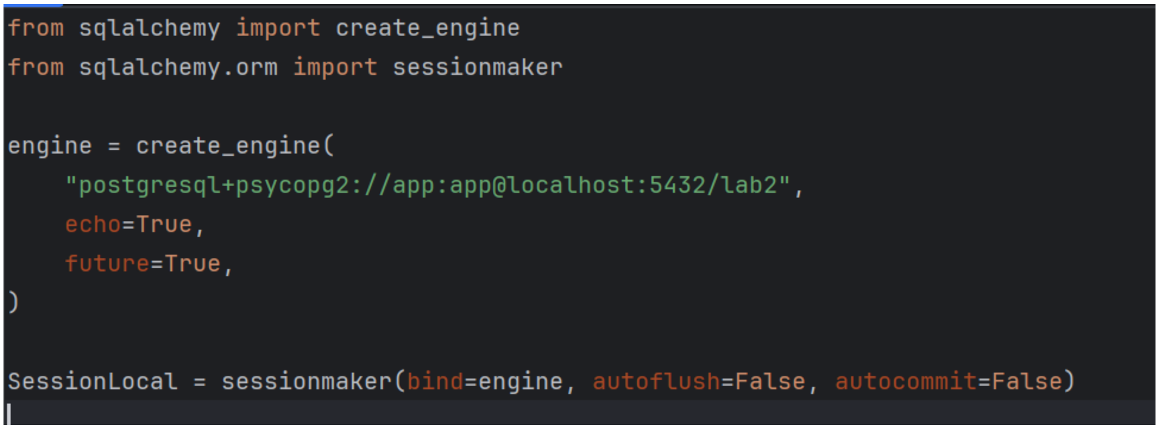
```
target_metadata = Base.metadata
```

4) Создание и применение к БД последней миграции

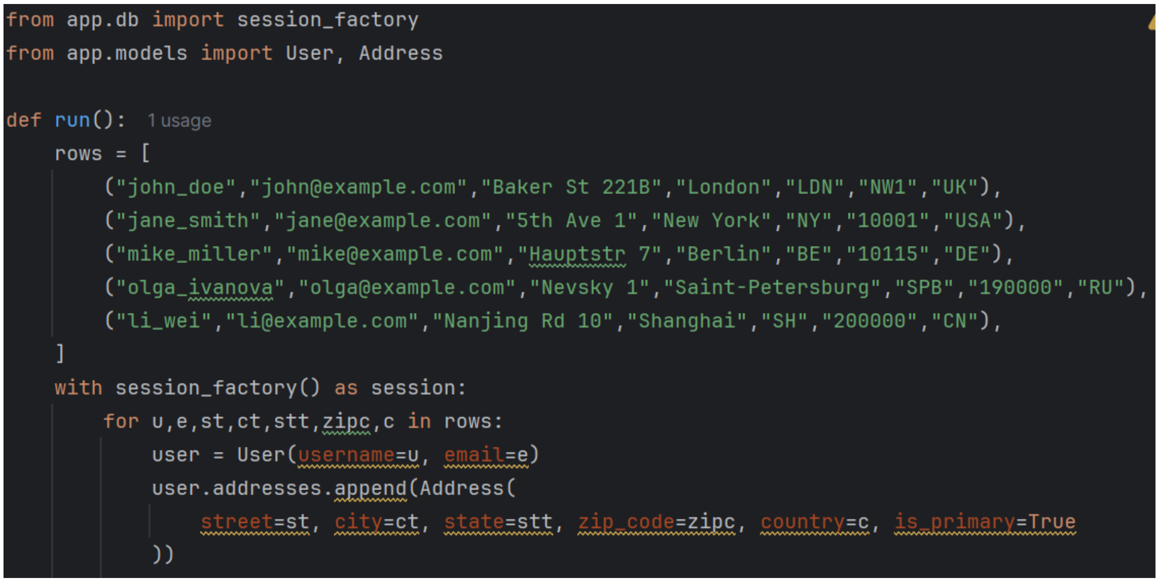


```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> alembic revision --autogenerate
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added table 'users'
INFO  [alembic.autogenerate.compare] Detected added table 'addresses'
Generating C:\Users\дом\PycharmProjects\PythonProject5\migrations\versions\748c2474a6f1_.py ...  done
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> alembic upgrade head
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 748c2474a6f1, empty message
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5>
```

5) Создание фабрики подключений, наполнение БД данными 5 пользователями и их адресами

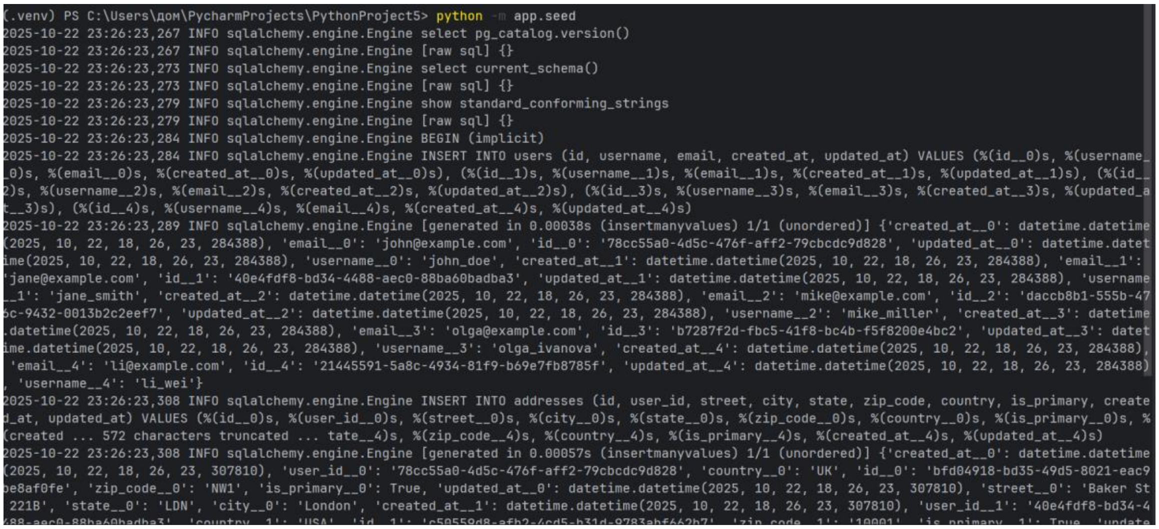


```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

engine = create_engine(
    "postgresql+psycopg2://app:app@localhost:5432/lab2",
    echo=True,
    future=True,
)

SessionLocal = sessionmaker(bind=engine, autoflush=False, autocommit=False)
```



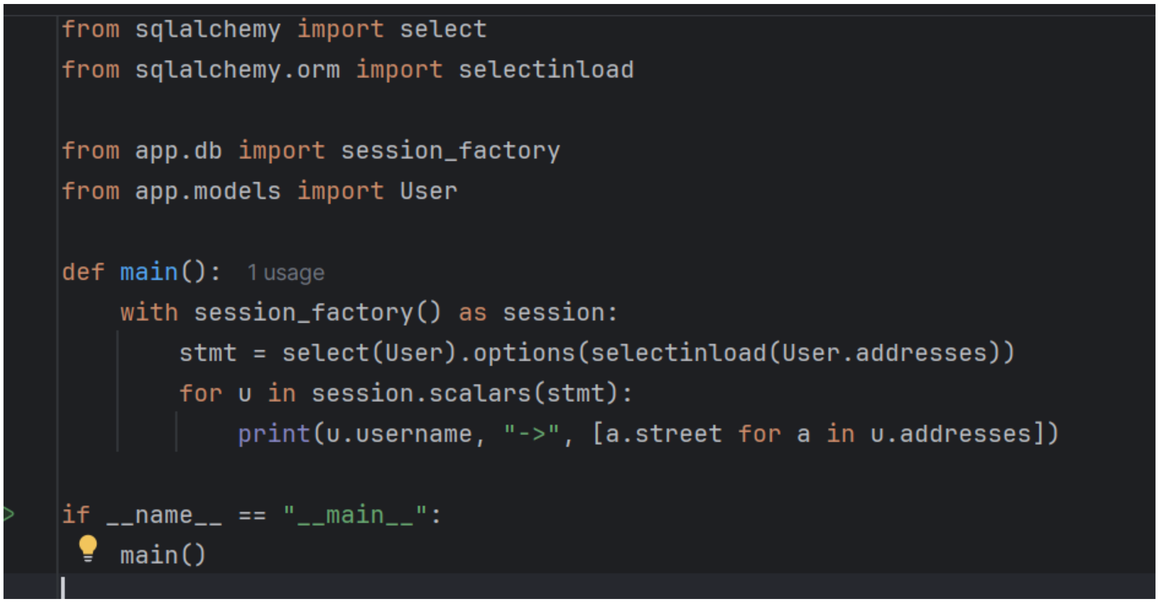
```python
from app.db import session_factory
from app.models import User, Address

def run():  1 usage
    rows = [
        ("john_doe","john@example.com","Baker St 221B","London","LDN","NW1","UK"),
        ("jane_smith","jane@example.com","5th Ave 1","New York","NY","10001","USA"),
        ("mike_miller","mike@example.com","Hauptstr 7","Berlin","BE","10115","DE"),
        ("olga_ivanova","olga@example.com","Nevsky 1","Saint-Petersburg","SPB","190000","RU"),
        ("li_wei","li@example.com","Nanjing Rd 10","Shanghai","SH","200000","CN"),
    ]
    with session_factory() as session:
        for u,e,st,ct,stt,zipc,c in rows:
            user = User(username=u, email=e)
            user.addresses.append(Address(
                street=st, city=ct, state=stt, zip_code=zipc, country=c, is_primary=True
            ))
```

6) Создание запроса связанных данных

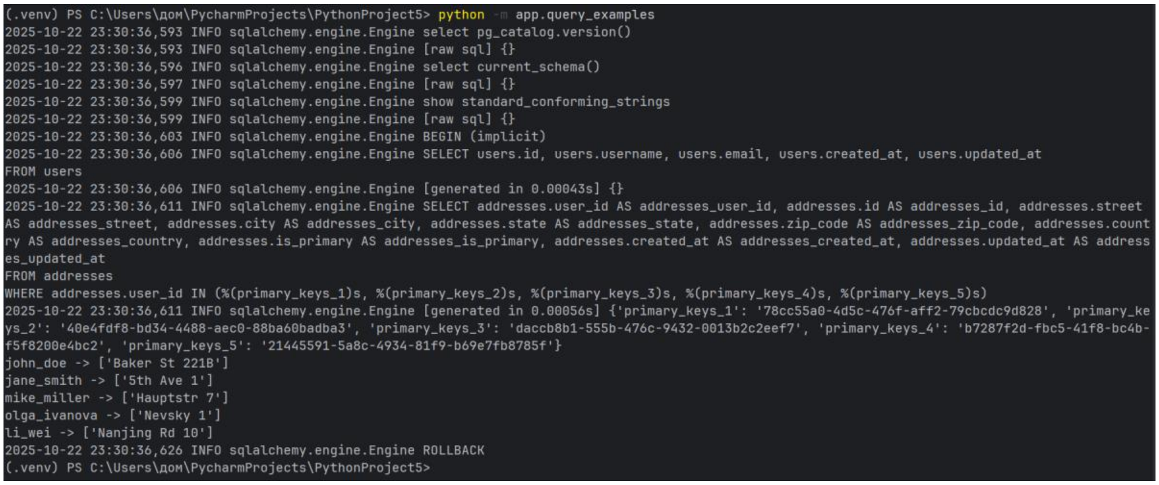


```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.seed
2025-10-22 23:26:23,267 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:26:23,267 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:26:23,273 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:26:23,273 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:26:23,279 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:26:23,279 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:26:23,284 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:26:23,284 INFO sqlalchemy.engine.Engine INSERT INTO users (id, username, email, created_at, updated_at) VALUES (%(id__0)s, %(username_
_0)s, %(email__0)s, %(created_at__0)s, %(updated_at__0)s), (%(id__1)s, %(username__1)s, %(email__1)s, %(created_at__1)s, %(updated_at__1)s), (%(id__
2)s, %(username__2)s, %(email__2)s, %(created_at__2)s, %(updated_at__2)s), (%(id__3)s, %(username__3)s, %(email__3)s, %(created_at__3)s, %(updated_a
t__3)s), (%(id__4)s, %(username__4)s, %(email__4)s, %(created_at__4)s, %(updated_at__4)s)
2025-10-22 23:26:23,289 INFO sqlalchemy.engine.Engine [generated in 0.00038s (insertmanyvalues) 1/1 (unordered)] {'created_at__0': datetime.datetime
(2025, 10, 22, 18, 26, 23, 284388), 'email__0': 'john@example.com', 'id__0': '78cc55a0-4d5c-476f-aff2-79cbcdc9d828', 'updated_at__0': datetime.datet
ime(2025, 10, 22, 18, 26, 23, 284388), 'username__0': 'john_doe', 'created_at__1': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388), 'email__1':
'jane@example.com', 'id__1': '40e4fdf8-bd34-4488-aec0-88ba60badba3', 'updated_at__1': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388), 'username
__1': 'jane_smith', 'created_at__2': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388), 'email__2': 'mike@example.com', 'id__2': 'daccb8b1-555b-47
6c-9432-0013b2c2eef7', 'updated_at__2': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388), 'username__2': 'mike_miller', 'created_at__3': datetime
.datetime(2025, 10, 22, 18, 26, 23, 284388), 'email__3': 'olga@example.com', 'id__3': 'b7287f2d-fbc5-41f8-bc4b-f5f8200e4bc2', 'updated_at__3': datet
ime.datetime(2025, 10, 22, 18, 26, 23, 284388), 'username__3': 'olga_ivanova', 'created_at__4': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388),
 'email__4': 'li@example.com', 'id__4': '21445591-5a8c-4934-81f9-b69e7fb8785f', 'updated_at__4': datetime.datetime(2025, 10, 22, 18, 26, 23, 284388)
, 'username__4': 'li_wei'}
2025-10-22 23:26:23,308 INFO sqlalchemy.engine.Engine INSERT INTO addresses (id, user_id, street, city, state, zip_code, country, is_primary, create
d_at, updated_at) VALUES (%(id__0)s, %(user_id__0)s, %(street__0)s, %(city__0)s, %(state__0)s, %(zip_code__0)s, %(country__0)s, %(is_primary__0)s, %
(created ... 572 characters truncated ... tate__4)s, %(zip_code__4)s, %(country__4)s, %(is_primary__4)s, %(created_at__4)s, %(updated_at__4)s)
2025-10-22 23:26:23,308 INFO sqlalchemy.engine.Engine [generated in 0.00057s (insertmanyvalues) 1/1 (unordered)] {'created_at__0': datetime.datetime
(2025, 10, 22, 18, 26, 23, 307810), 'user_id__0': '78cc55a0-4d5c-476f-aff2-79cbcdc9d828', 'country__0': 'UK', 'id__0': 'bfd04918-bd35-49d5-8021-eac9
be8af0fe', 'zip_code__0': 'NW1', 'is_primary__0': True, 'updated_at__0': datetime.datetime(2025, 10, 22, 18, 26, 23, 307810), 'street__0': 'Baker St
 221B', 'state__0': 'LDN', 'city__0': 'London', 'created_at__1': datetime.datetime(2025, 10, 22, 18, 26, 23, 307810), 'user_id__1': '40e4fdf8-bd34-4
488-aec0-88ba60badba3', 'country__1': 'USA', 'id__1': 'c50559d8-afb2-4cd5-b31d-9783abf662b7', 'zip_code__1': '10001', 'is_primary__1': True, 'update
```

7) Запрос связанных данных



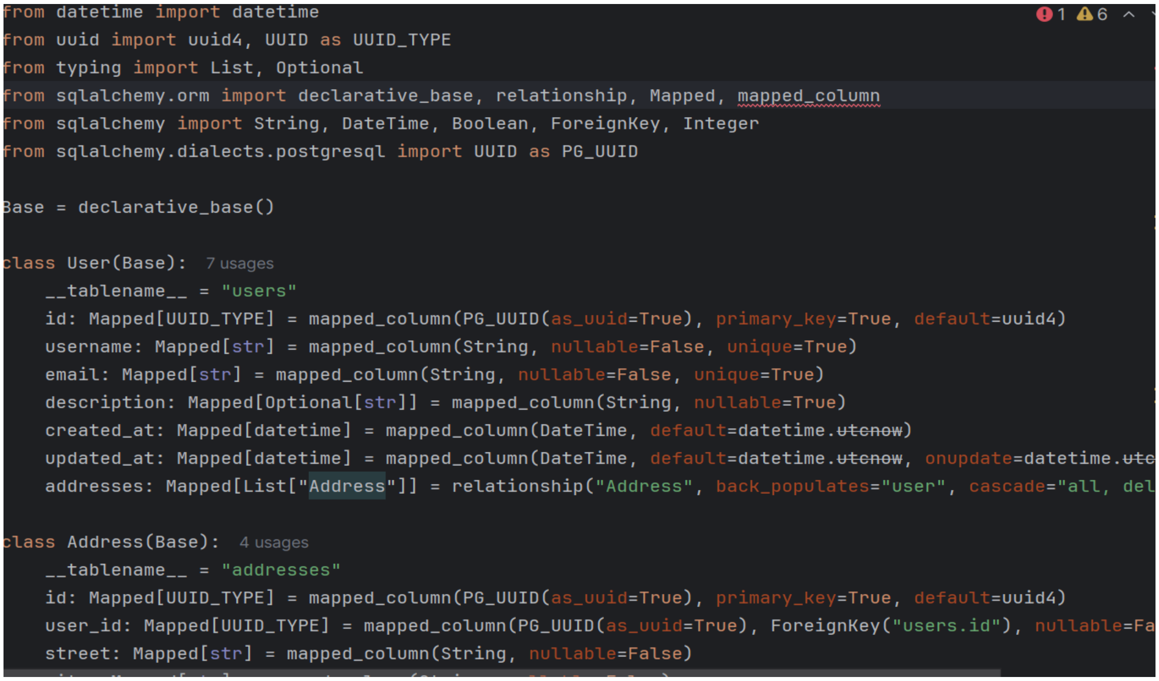
```
from sqlalchemy import select
from sqlalchemy.orm import selectinload

from app.db import session_factory
from app.models import User

def main():  1 usage
    with session_factory() as session:
        stmt = select(User).options(selectinload(User.addresses))
        for u in session.scalars(stmt):
            print(u.username, "->", [a.street for a in u.addresses])

if __name__ == "__main__":
    main()
```



```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.query_examples
2025-10-22 23:30:36,593 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:30:36,593 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:30:36,596 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:30:36,597 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:30:36,599 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:30:36,599 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:30:36,603 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:30:36,606 INFO sqlalchemy.engine.Engine SELECT users.id, users.username, users.email, users.created_at, users.updated_at
FROM users
2025-10-22 23:30:36,606 INFO sqlalchemy.engine.Engine [generated in 0.00043s] {}
2025-10-22 23:30:36,611 INFO sqlalchemy.engine.Engine SELECT addresses.user_id AS addresses_user_id, addresses.id AS addresses_id, addresses.street
AS addresses_street, addresses.city AS addresses_city, addresses.state AS addresses_state, addresses.zip_code AS addresses_zip_code, addresses.count
ry AS addresses_country, addresses.is_primary AS addresses_is_primary, addresses.created_at AS addresses_created_at, addresses.updated_at AS address
es_updated_at
FROM addresses
WHERE addresses.user_id IN (%(primary_keys_1)s, %(primary_keys_2)s, %(primary_keys_3)s, %(primary_keys_4)s, %(primary_keys_5)s)
2025-10-22 23:30:36,611 INFO sqlalchemy.engine.Engine [generated in 0.00056s] {'primary_keys_1': '78cc55a0-4d5c-476f-aff2-79cbcdc9d828', 'primary_ke
ys_2': '40e4fdf8-bd34-4488-aec0-88ba60badba3', 'primary_keys_3': 'daccb8b1-555b-476c-9432-0013b2c2eef7', 'primary_keys_4': 'b7287f2d-fbc5-41f8-bc4b-
f5f8200e4bc2', 'primary_keys_5': '21445591-5a8c-4934-81f9-b69e7fb8785f'}
john_doe -> ['Baker St 221B']
jane_smith -> ['5th Ave 1']
mike_miller -> ['Hauptstr 7']
olga_ivanova -> ['Nevsky 1']
li_wei -> ['Nanjing Rd 10']
2025-10-22 23:30:36,626 INFO sqlalchemy.engine.Engine ROLLBACK
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5>
```

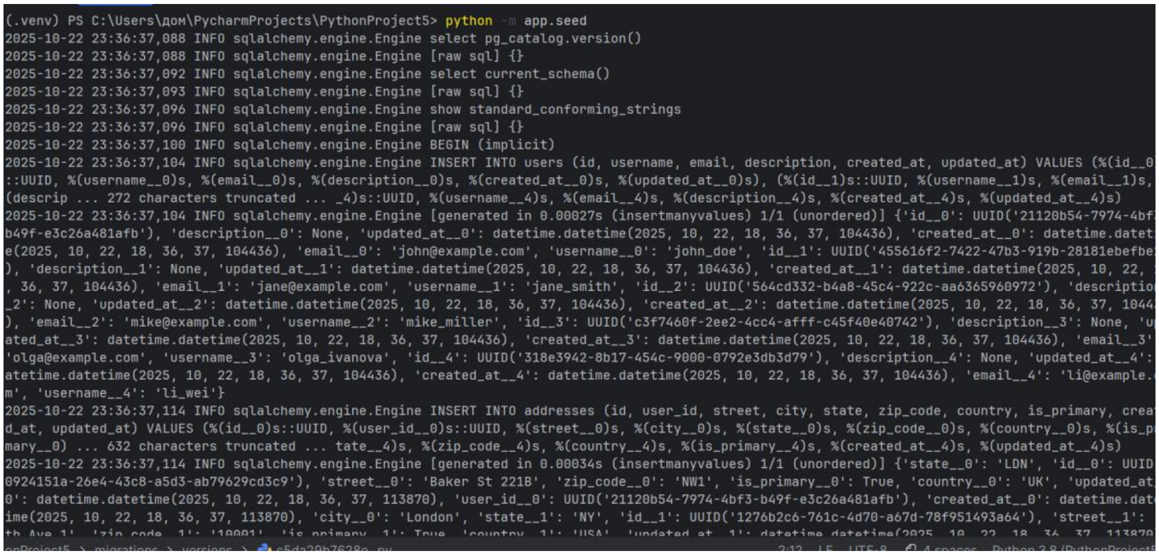
8) Последующие работы с БД и миграции

В ОРМ добавляем пользователю дополнительное строковое поле description. Добавляем дополнительную таблицу для продукции и заказов. Заказ должен в себе содержать информацию о пользователе, адресе доставки и продукции. Производим миграцию данных и добавляем 5 продукций и 5 заказов в БД.



```
from datetime import datetime
from uuid import uuid4, UUID as UUID_TYPE
from typing import List, Optional
from sqlalchemy.orm import declarative_base, relationship, Mapped, mapped_column
from sqlalchemy import String, DateTime, Boolean, ForeignKey, Integer
from sqlalchemy.dialects.postgresql import UUID as PG_UUID

Base = declarative_base()

class User(Base):  7 usages
    __tablename__ = "users"
    id: Mapped[UUID_TYPE] = mapped_column(PG_UUID(as_uuid=True), primary_key=True, default=uuid4)
    username: Mapped[str] = mapped_column(String, nullable=False, unique=True)
    email: Mapped[str] = mapped_column(String, nullable=False, unique=True)
    description: Mapped[Optional[str]] = mapped_column(String, nullable=True)
    created_at: Mapped[datetime] = mapped_column(DateTime, default=datetime.~~utcnow~~)
    updated_at: Mapped[datetime] = mapped_column(DateTime, default=datetime.~~utcnow~~, onupdate=datetime.~~utc~~
    addresses: Mapped[List["Address"]] = relationship("Address", back_populates="user", cascade="all, del

class Address(Base):  4 usages
    __tablename__ = "addresses"
    id: Mapped[UUID_TYPE] = mapped_column(PG_UUID(as_uuid=True), primary_key=True, default=uuid4)
    user_id: Mapped[UUID_TYPE] = mapped_column(PG_UUID(as_uuid=True), ForeignKey("users.id"), nullable=Fa
    street: Mapped[str] = mapped_column(String, nullable=False)
```

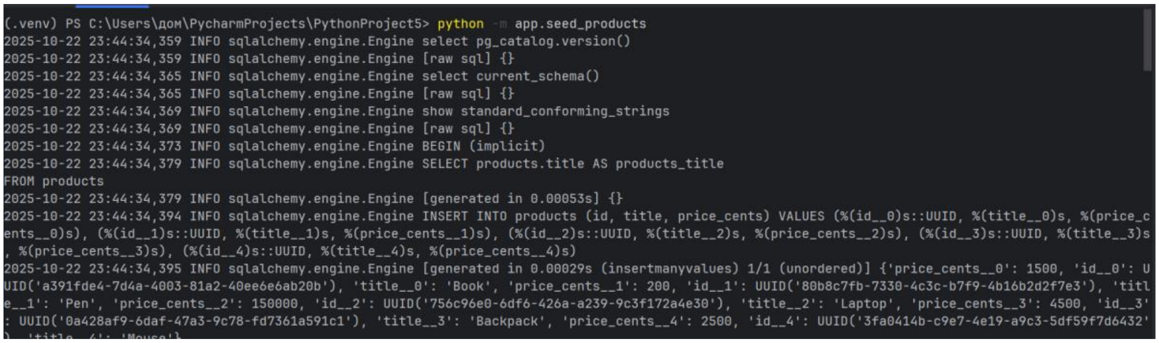
Наполнение БД: добавлены 5 пользователей и их адреса



```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.seed
2025-10-22 23:36:37,088 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:36:37,088 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:36:37,092 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:36:37,093 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:36:37,096 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:36:37,096 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:36:37,100 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:36:37,104 INFO sqlalchemy.engine.Engine INSERT INTO users (id, username, email, description, created_at, updated_at) VALUES (%(id__0
::UUID, %(username__0)s, %(email__0)s, %(description__0)s, %(created_at__0)s, %(updated_at__0)s), (%(id__1)s::UUID, %(username__1)s, %(email__1)s,
(descrip ... 272 characters truncated ... _4)s::UUID, %(username__4)s, %(email__4)s, %(description__4)s, %(created_at__4)s, %(updated_at__4)s)
2025-10-22 23:36:37,104 INFO sqlalchemy.engine.Engine [generated in 0.00027s (insertmanyvalues) 1/1 (unordered)] {'id__0': UUID('21120b54-7974-4bf
b49f-e3c26a481afb'), 'description__0': None, 'updated_at__0': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'created_at__0': datetime.datet
e(2025, 10, 22, 18, 36, 37, 104436), 'email__0': 'john@example.com', 'username__0': 'john_doe', 'id__1': UUID('455616f2-7422-47b3-919b-28181ebefbe
), 'description__1': None, 'updated_at__1': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'created_at__1': datetime.datetime(2025, 10, 22,
, 36, 37, 104436), 'email__1': 'jane@example.com', 'username__1': 'jane_smith', 'id__2': UUID('564cd332-b4a8-45c4-922c-aa6365960972'), 'descriptio
_2': None, 'updated_at__2': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'created_at__2': datetime.datetime(2025, 10, 22, 18, 36, 37, 1044
), 'email__2': 'mike@example.com', 'username__2': 'mike_miller', 'id__3': UUID('c3f7460f-2ee2-4cc4-afff-c45f40e40742'), 'description__3': None, 'u
ated_at__3': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'created_at__3': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'email__3'
'olga@example.com', 'username__3': 'olga_ivanova', 'id__4': UUID('318e3942-8b17-454c-9000-0792e3db3d79'), 'description__4': None, 'updated_at__4':
atetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'created_at__4': datetime.datetime(2025, 10, 22, 18, 36, 37, 104436), 'email__4': 'li@example.
m', 'username__4': 'li_wei'}
2025-10-22 23:36:37,114 INFO sqlalchemy.engine.Engine INSERT INTO addresses (id, user_id, street, city, state, zip_code, country, is_primary, crea
d_at, updated_at) VALUES (%(id__0)s::UUID, %(user_id__0)s::UUID, %(street__0)s, %(city__0)s, %(state__0)s, %(zip_code__0)s, %(country__0)s, %(is_p
mary__0) ... 632 characters truncated ... tate__4)s, %(zip_code__4)s, %(country__4)s, %(is_primary__4)s, %(created_at__4)s, %(updated_at__4)s)
2025-10-22 23:36:37,114 INFO sqlalchemy.engine.Engine [generated in 0.00034s (insertmanyvalues) 1/1 (unordered)] {'state__0': 'LDN', 'id__0': UUID
0924151a-26e4-43c8-a5d3-ab79629cd3c9'), 'street__0': 'Baker St 221B', 'zip_code__0': 'NW1', 'is_primary__0': True, 'country__0': 'UK', 'updated_at
0': datetime.datetime(2025, 10, 22, 18, 36, 37, 113870), 'user_id__0': UUID('21120b54-7974-4bf3-b49f-e3c26a481afb'), 'created_at__0': datetime.dat
ime(2025, 10, 22, 18, 36, 37, 113870), 'city__0': 'London', 'state__1': 'NY', 'id__1': UUID('1276b2c6-761c-4d70-a67d-78f951493a64'), 'street__1':
th Ave 1', 'zip_code__1': '10001', 'is_primary__1': True, 'country__1': 'USA', 'updated_at__1': datetime.datetime(2025, 10, 22, 18, 36, 37, 113870
```

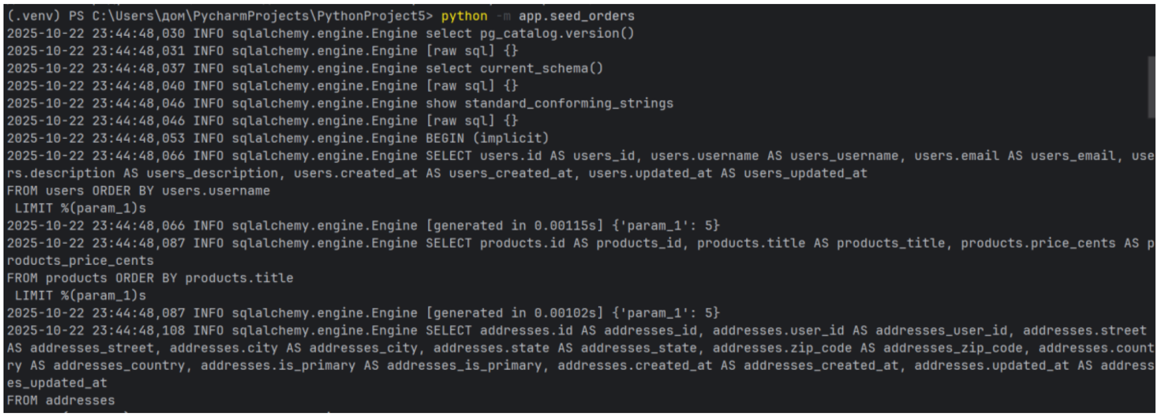
onProject5 > migrations > versions > c5da29b7628e.py
2:12 LF UTF-8 4 spaces Python 3.8 (PythonProject5

Наполнение БД товарами



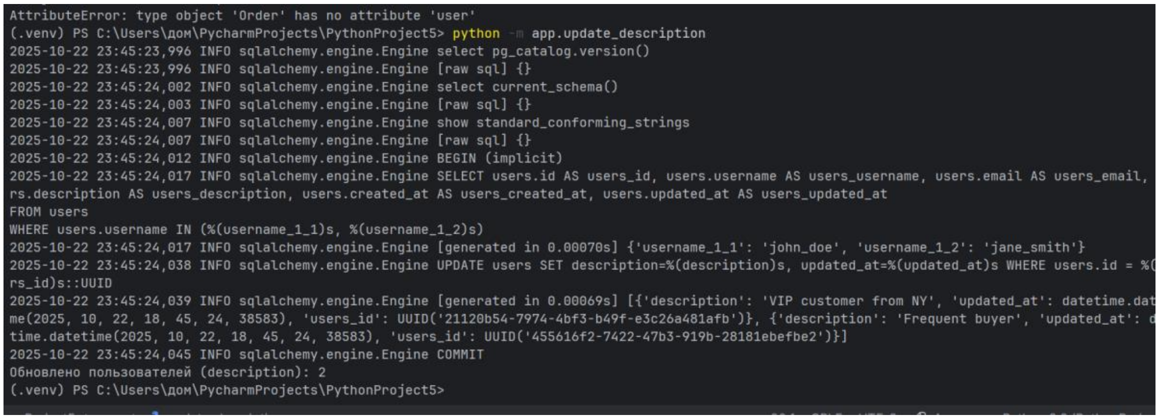
```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.seed_products
2025-10-22 23:44:34,359 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:44:34,359 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:34,365 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:44:34,365 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:34,369 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:44:34,369 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:34,373 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:44:34,379 INFO sqlalchemy.engine.Engine SELECT products.title AS products_title
FROM products
2025-10-22 23:44:34,379 INFO sqlalchemy.engine.Engine [generated in 0.00053s] {}
2025-10-22 23:44:34,394 INFO sqlalchemy.engine.Engine INSERT INTO products (id, title, price_cents) VALUES (%(id__0)s::UUID, %(title__0)s, %(price_c
ents__0)s), (%(id__1)s::UUID, %(title__1)s, %(price_cents__1)s), (%(id__2)s::UUID, %(title__2)s, %(price_cents__2)s), (%(id__3)s::UUID, %(title__3)s
, %(price_cents__3)s), (%(id__4)s::UUID, %(title__4)s, %(price_cents__4)s)
2025-10-22 23:44:34,395 INFO sqlalchemy.engine.Engine [generated in 0.00029s (insertmanyvalues) 1/1 (unordered)] {'price_cents__0': 1500, 'id__0': U
UID('a391fde4-7d4a-4003-81a2-40ee6e6ab20b'), 'title__0': 'Book', 'price_cents__1': 200, 'id__1': UUID('80b8c7fb-7330-4c3c-b7f9-4b16b2d2f7e3'), 'titl
e__1': 'Pen', 'price_cents__2': 150000, 'id__2': UUID('756c96e0-6df6-426a-a239-9c3f172a4e30'), 'title__2': 'Laptop', 'price_cents__3': 4500, 'id__3'
: UUID('0a428af9-6daf-47a3-9c78-fd7361a591c1'), 'title__3': 'Backpack', 'price_cents__4': 2500, 'id__4': UUID('3fa0414b-c9e7-4e19-a9c3-5df59f7d6432'
), 'title__4': 'Mouse'}
```

Создание заказов



```
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.seed_orders
2025-10-22 23:44:48,030 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:44:48,031 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:48,037 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:44:48,040 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:48,046 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:44:48,046 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:44:48,053 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:44:48,066 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.username AS users_username, users.email AS users_email, use
rs.description AS users_description, users.created_at AS users_created_at, users.updated_at AS users_updated_at
FROM users ORDER BY users.username
 LIMIT %(param_1)s
2025-10-22 23:44:48,066 INFO sqlalchemy.engine.Engine [generated in 0.00115s] {'param_1': 5}
2025-10-22 23:44:48,087 INFO sqlalchemy.engine.Engine SELECT products.id AS products_id, products.title AS products_title, products.price_cents AS p
roducts_price_cents
FROM products ORDER BY products.title
 LIMIT %(param_1)s
2025-10-22 23:44:48,087 INFO sqlalchemy.engine.Engine [generated in 0.00102s] {'param_1': 5}
2025-10-22 23:44:48,108 INFO sqlalchemy.engine.Engine SELECT addresses.id AS addresses_id, addresses.user_id AS addresses_user_id, addresses.street
AS addresses_street, addresses.city AS addresses_city, addresses.state AS addresses_state, addresses.zip_code AS addresses_zip_code, addresses.count
ry AS addresses_country, addresses.is_primary AS addresses_is_primary, addresses.created_at AS addresses_created_at, addresses.updated_at AS address
es_updated_at
FROM addresses
```

Обновление описаний пользователей



```
AttributeError: type object 'Order' has no attribute 'user'
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5> python -m app.update_description
2025-10-22 23:45:23,996 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-22 23:45:23,996 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:45:24,002 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-22 23:45:24,003 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:45:24,007 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-22 23:45:24,007 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-22 23:45:24,012 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-22 23:45:24,017 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.username AS users_username, users.email AS users_email,
rs.description AS users_description, users.created_at AS users_created_at, users.updated_at AS users_updated_at
FROM users
WHERE users.username IN (%(username_1_1)s, %(username_1_2)s)
2025-10-22 23:45:24,017 INFO sqlalchemy.engine.Engine [generated in 0.00070s] {'username_1_1': 'john_doe', 'username_1_2': 'jane_smith'}
2025-10-22 23:45:24,038 INFO sqlalchemy.engine.Engine UPDATE users SET description=%(description)s, updated_at=%(updated_at)s WHERE users.id = %(
rs_id)s::UUID
2025-10-22 23:45:24,039 INFO sqlalchemy.engine.Engine [generated in 0.00069s] [{'description': 'VIP customer from NY', 'updated_at': datetime.dat
me(2025, 10, 22, 18, 45, 24, 38583), 'users_id': UUID('21120b54-7974-4bf3-b49f-e3c26a481afb')}, {'description': 'Frequent buyer', 'updated_at': d
time.datetime(2025, 10, 22, 18, 45, 24, 38583), 'users_id': UUID('455616f2-7422-47b3-919b-28181ebefbe2')}]
2025-10-22 23:45:24,045 INFO sqlalchemy.engine.Engine COMMIT
Обновлено пользователей (description): 2
(.venv) PS C:\Users\дом\PycharmProjects\PythonProject5>
```

Ответы на вопросы

1. Подходы маппинга в SQLAlchemy: Declarative (классы), Classical (ручное сопоставление таблиц и классов) и Automap (автогенерация классов по существующей схеме). Declarative — основной и наиболее удобный в новых проектах.

2. Alembic хранит текущую версию БД в служебной таблице alembic_version, а история описывается ревизиями в каталоге versions.

3. В работе реализованы связи: один-ко-многим (User → Address) и многие-ко-многим (Order ↔ Product через таблицу order_products).

4. Миграция БД — контролируемое изменение схемы (создание/изменение таблиц, полей, индексов) без потери данных; позволяет последовательно развивать структуру.

5. Отношения многие-ко-многим в SQLAlchemy реализуются через вспомогательную таблицу (association table), указанную в параметре secondary.

6. При конфликте версий Alembic либо выполняют merge-ревизию (объединение веток), либо вручную переносят изменения и создают новую ревизию; затем приводят БД к единому head.

Вывод

Освоены принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучены механизмы миграции базы данных.