

# **Лабораторная работа №2: Работа с SQLAlchemy и alembic**

РИМ-150950

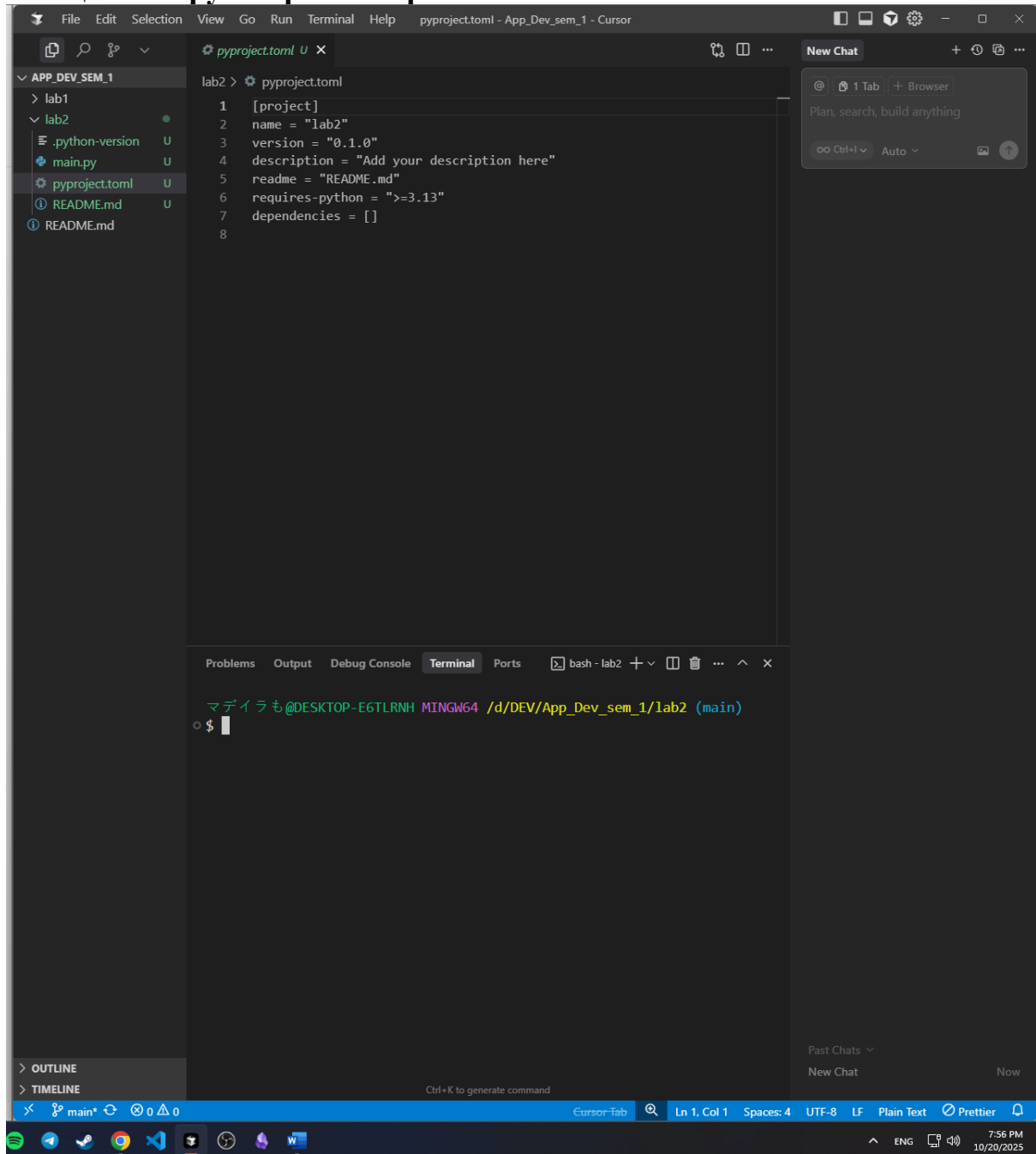
Жунёв Андрей

## **Цель работы**

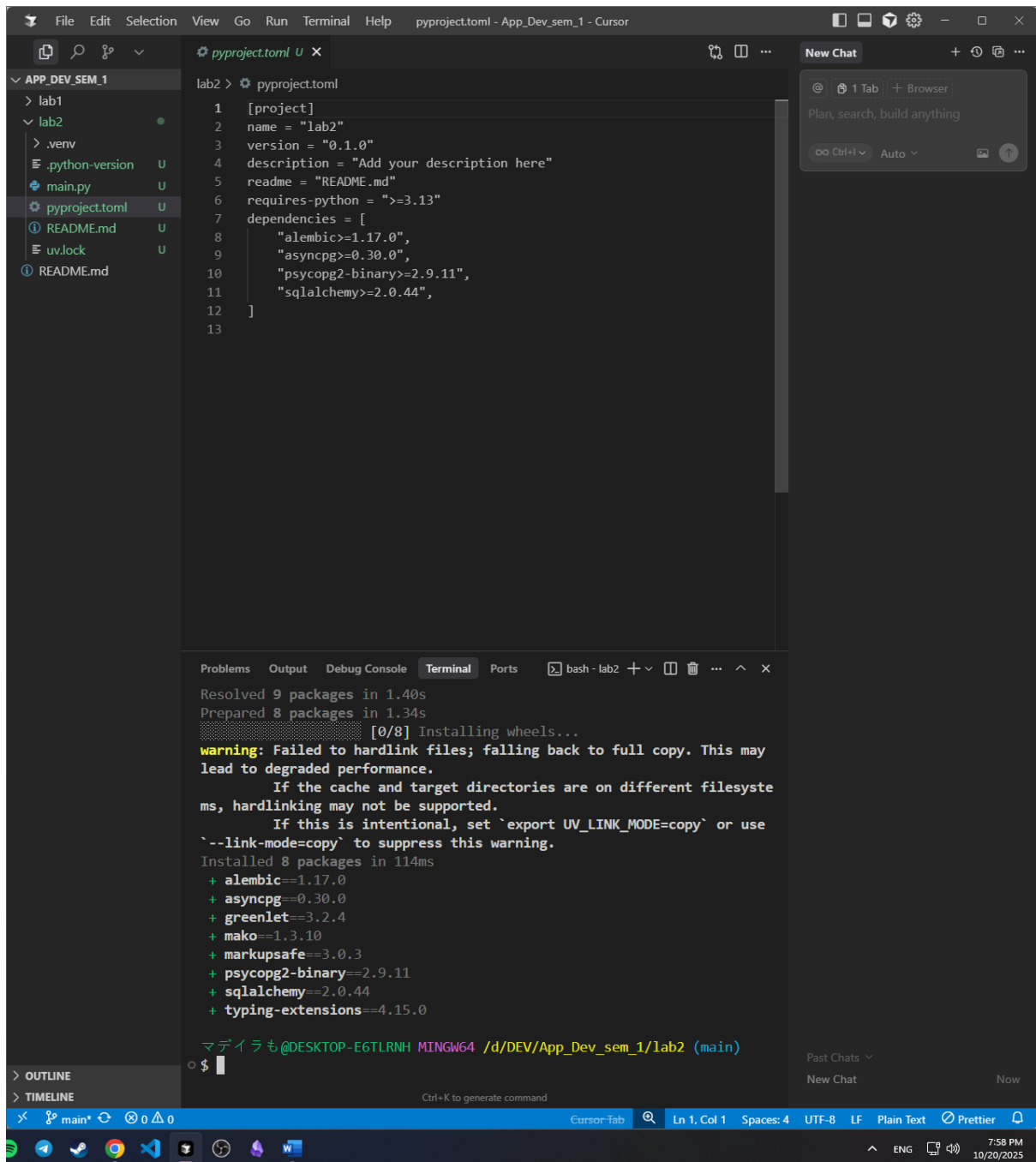
Освоить принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучить механизмы миграции базы данных.

## Ход работы:

### Инициализируем проект через UV



### Устанавливаем необходимые библиотеки



SQLAlchemy – основная библиотека для работы с базами данных через ORM

Alembic – инструмент для управления миграциями базы данных

psycopg2-binary – драйвер для подключения к PostgreSQL(синхронный)

asyncpg – драйвер для асинхронной работы с PostgreSQL

Создание OPM для пользователя и адреса

```
lab2 > models.py > Address
1 from uuid import uuid4
2 from datetime import datetime
3 from sqlalchemy import ForeignKey
4 from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column, relationship
5 from typing import Optional
6
7
8 class Base(DeclarativeBase):
9     pass
10
11 class User(Base):
12     __tablename__ = 'users'
13
14     id: Mapped[uuid4] = mapped_column(
15         primary_key=True,
16         default=uuid4,
17     )
18     username: Mapped[str] = mapped_column(nullable=False, unique=True)
19     email: Mapped[str] = mapped_column(nullable=False, unique=True)
20     created_at: Mapped[datetime] = mapped_column(default=datetime.now)
21     updated_at: Mapped[datetime] = mapped_column(onupdate=datetime.now)
22
23     addresses = relationship('Address', back_populates='user')
24
25
26 class Address(Base):
27     __tablename__ = 'addresses'
28
29     id: Mapped[uuid4] = mapped_column(
30         primary_key=True,
31         default=uuid4,
32     )
33     user_id: Mapped[uuid4] = mapped_column(ForeignKey('users.id'), nullable=False)
34     street: Mapped[str] = mapped_column(nullable=False)
35     city: Mapped[str] = mapped_column(nullable=False)
36     state: Mapped[str] = mapped_column()
37     zip_code: Mapped[str] = mapped_column()
38     country: Mapped[str] = mapped_column(nullable=False)
39     is_primary: Mapped[bool] = mapped_column(default=False)
40
41     created_at: Mapped[datetime] = mapped_column(default=datetime.now)
42     updated_at: Mapped[datetime] = mapped_column(onupdate=datetime.now)
43
44     user = relationship('User', back_populates='addresses')
```

Problems 2 Output Debug Console Terminal Ports

Resolved 9 packages in 0.75ms  
Audited 8 packages in 0.44ms

マデイラも@DESKTOP-E6TLRNH MINGW64 /d/DEV/App\_Dev\_sem\_1/lab2 (main)  
\$ uv sync  
Resolved 9 packages in 0.72ms  
Audited 8 packages in 0.41ms

マデイラも@DESKTOP-E6TLRNH MINGW64 /d/DEV/App\_Dev\_sem\_1/lab2 (main)  
o \$

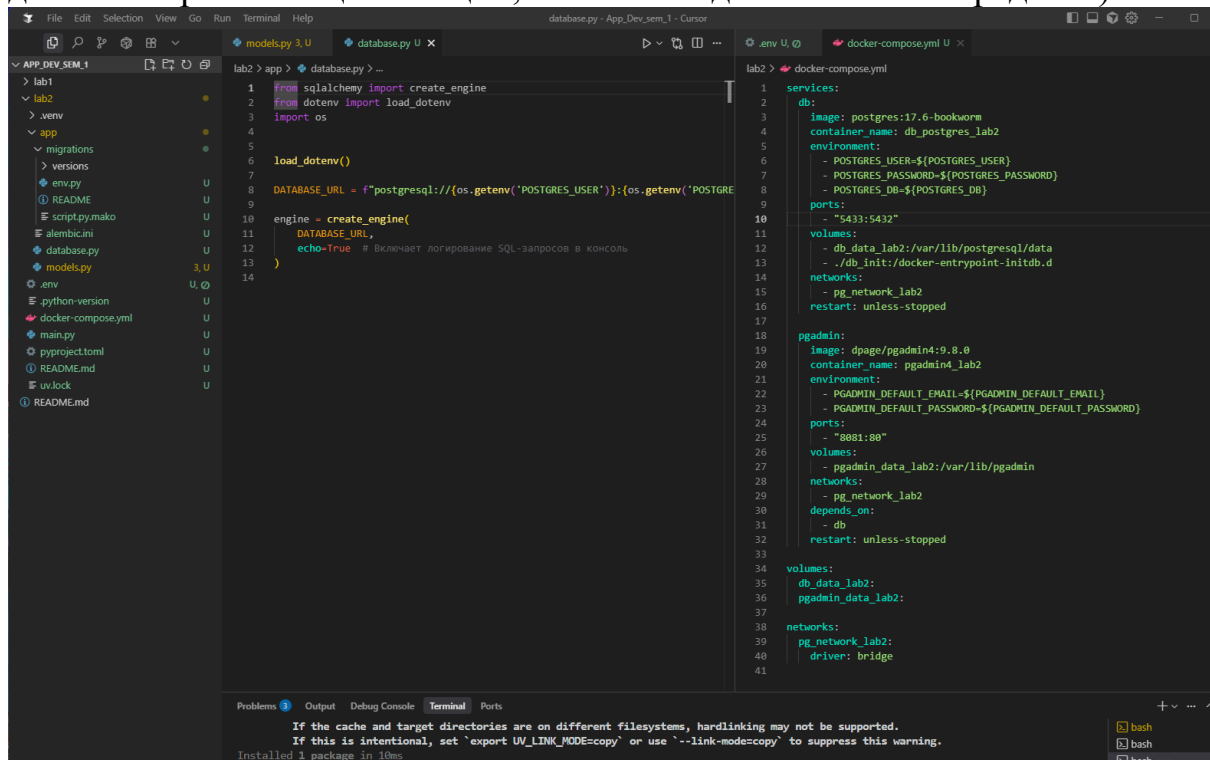
OUTLINE  
TIMELINE

Cursor Tab Ln 44, Col 60 Spaces: 4 UTF-8 CRLF Python Select Interpreter Prettier

8:18 PM  
10/20/2025

БД развернута. Таблицы заранее не создаются (сначала создал заранее,

добавив скрипт инициализации, не понял задание — затем переделал)



## Настройка миграций с Alembic

Инициализировал миграции, заменил строку подключения к БД в alembic.ini

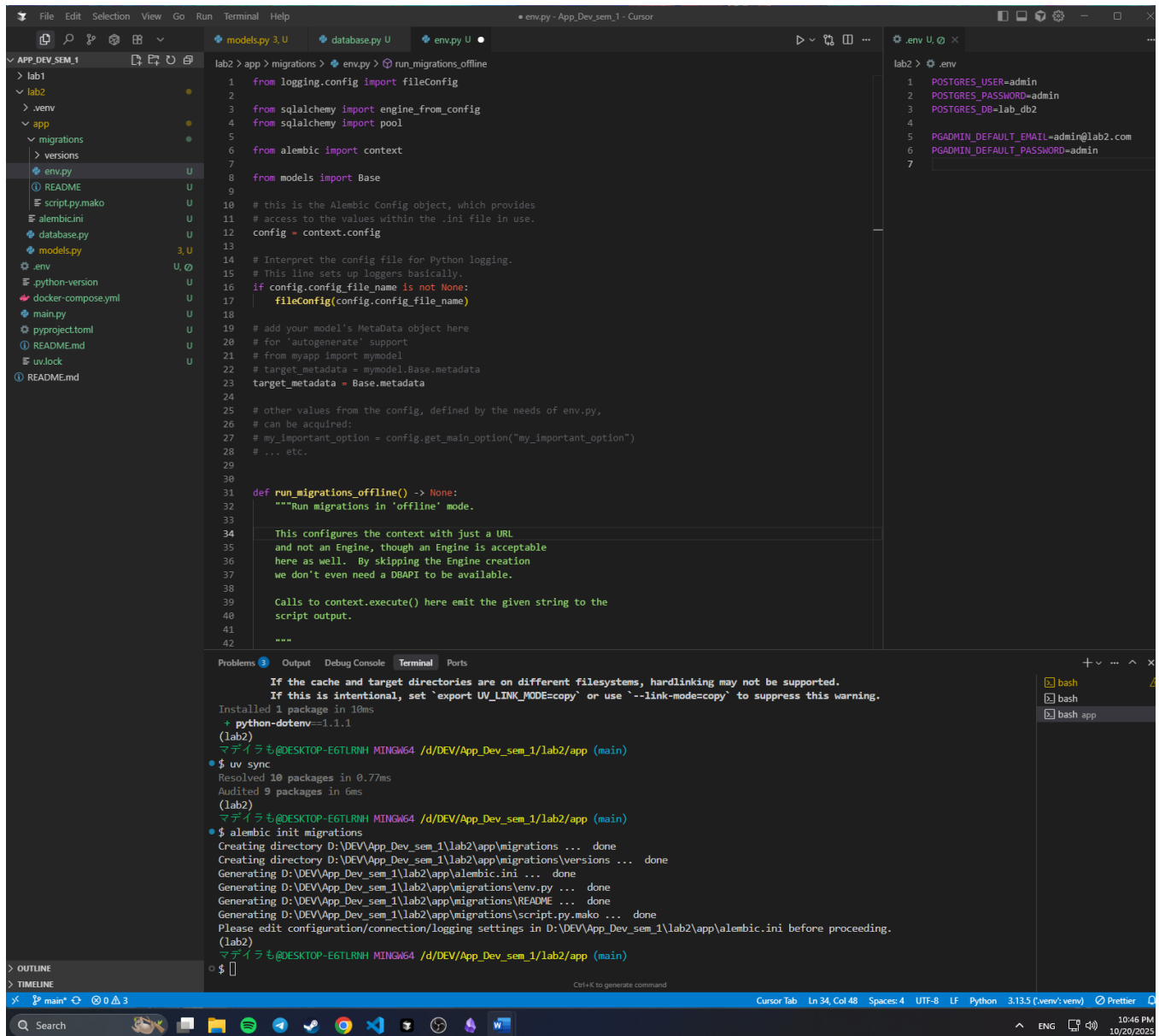
```
Run Terminal Help
.env - App_Dev_sem_1 - Cursor

models.py 3, U database.py U alembic.ini U ...

lab2 > app > alembic.ini
69 # path_separator = space
70 # path_separator = newline
71 #
72 # Use os.pathsep. Default configuration used for new projects.
73 path_separator = os
74
75 # set to 'true' to search source files recursively
76 # in each "version_locations" directory
77 # new in Alembic version 1.10
78 # recursive_version_locations = false
79
80 # the output encoding used when revision files
81 # are written from script.py.mako
82 # output_encoding = utf-8
83
84 # database URL. This is consumed by the user-maintained env.py script only.
85 # other means of configuring database URLs may be customized within the env.py
86 # file.
87 sqlalchemy.url = postgresql://admin:admin@localhost:5433/lab2_db
88
89
90 [post_write_hooks]
91 # post_write_hooks defines scripts or Python functions that are run
92 # on newly generated revision scripts. See the documentation for further
93 # detail and examples
94
95 # format using "black" - use the console_scripts runner, against the "black" entrypoint
96 # hooks = black
97 # black.type = console_scripts
98 # black.entrypoint = black
99 # black.options = -l 79 REVISION_SCRIPT_FILENAME
100
101 # lint with attempts to fix using "ruff" - use the module runner, against the "ruff" module
102 # hooks = ruff
103 # ruff.type = module
104 # ruff.module = ruff
105 # ruff.options = check --fix REVISION_SCRIPT_FILENAME
106
107 # Alternatively, use the exec runner to execute a binary found on your PATH
108 # hooks = ruff
109 # ruff.type = exec
110 # ruff.executable = ruff
111 # ruff.executable = ruff
112 # ruff.executable = ruff
113 # ruff.executable = ruff
114 # ruff.executable = ruff
115 # ruff.executable = ruff
116 # ruff.executable = ruff
117 # ruff.executable = ruff
118 # ruff.executable = ruff
119 # ruff.executable = ruff
120 # ruff.executable = ruff
121 # ruff.executable = ruff
122 # ruff.executable = ruff
123 # ruff.executable = ruff
124 # ruff.executable = ruff
125 # ruff.executable = ruff
126 # ruff.executable = ruff
127 # ruff.executable = ruff
128 # ruff.executable = ruff
129 # ruff.executable = ruff
130 # ruff.executable = ruff
131 # ruff.executable = ruff
132 # ruff.executable = ruff
133 # ruff.executable = ruff
134 # ruff.executable = ruff
135 # ruff.executable = ruff
136 # ruff.executable = ruff
137 # ruff.executable = ruff
138 # ruff.executable = ruff
139 # ruff.executable = ruff
140 # ruff.executable = ruff
141 # ruff.executable = ruff
142 # ruff.executable = ruff
143 # ruff.executable = ruff
144 # ruff.executable = ruff
145 # ruff.executable = ruff
146 # ruff.executable = ruff
147 # ruff.executable = ruff
148 # ruff.executable = ruff
149 # ruff.executable = ruff
150 # ruff.executable = ruff
151 # ruff.executable = ruff
152 # ruff.executable = ruff
153 # ruff.executable = ruff
154 # ruff.executable = ruff
155 # ruff.executable = ruff
156 # ruff.executable = ruff
157 # ruff.executable = ruff
158 # ruff.executable = ruff
159 # ruff.executable = ruff
160 # ruff.executable = ruff
161 # ruff.executable = ruff
162 # ruff.executable = ruff
163 # ruff.executable = ruff
164 # ruff.executable = ruff
165 # ruff.executable = ruff
166 # ruff.executable = ruff
167 # ruff.executable = ruff
168 # ruff.executable = ruff
169 # ruff.executable = ruff
170 # ruff.executable = ruff
171 # ruff.executable = ruff
172 # ruff.executable = ruff
173 # ruff.executable = ruff
174 # ruff.executable = ruff
175 # ruff.executable = ruff
176 # ruff.executable = ruff
177 # ruff.executable = ruff
178 # ruff.executable = ruff
179 # ruff.executable = ruff
180 # ruff.executable = ruff
181 # ruff.executable = ruff
182 # ruff.executable = ruff
183 # ruff.executable = ruff
184 # ruff.executable = ruff
185 # ruff.executable = ruff
186 # ruff.executable = ruff
187 # ruff.executable = ruff
188 # ruff.executable = ruff
189 # ruff.executable = ruff
190 # ruff.executable = ruff
191 # ruff.executable = ruff
192 # ruff.executable = ruff
193 # ruff.executable = ruff
194 # ruff.executable = ruff
195 # ruff.executable = ruff
196 # ruff.executable = ruff
197 # ruff.executable = ruff
198 # ruff.executable = ruff
199 # ruff.executable = ruff
200 # ruff.executable = ruff
201 # ruff.executable = ruff
202 # ruff.executable = ruff
203 # ruff.executable = ruff
204 # ruff.executable = ruff
205 # ruff.executable = ruff
206 # ruff.executable = ruff
207 # ruff.executable = ruff
208 # ruff.executable = ruff
209 # ruff.executable = ruff
210 # ruff.executable = ruff
211 # ruff.executable = ruff
212 # ruff.executable = ruff
213 # ruff.executable = ruff
214 # ruff.executable = ruff
215 # ruff.executable = ruff
216 # ruff.executable = ruff
217 # ruff.executable = ruff
218 # ruff.executable = ruff
219 # ruff.executable = ruff
220 # ruff.executable = ruff
221 # ruff.executable = ruff
222 # ruff.executable = ruff
223 # ruff.executable = ruff
224 # ruff.executable = ruff
225 # ruff.executable = ruff
226 # ruff.executable = ruff
227 # ruff.executable = ruff
228 # ruff.executable = ruff
229 # ruff.executable = ruff
230 # ruff.executable = ruff
231 # ruff.executable = ruff
232 # ruff.executable = ruff
233 # ruff.executable = ruff
234 # ruff.executable = ruff
235 # ruff.executable = ruff
236 # ruff.executable = ruff
237 # ruff.executable = ruff
238 # ruff.executable = ruff
239 # ruff.executable = ruff
240 # ruff.executable = ruff
241 # ruff.executable = ruff
242 # ruff.executable = ruff
243 # ruff.executable = ruff
244 # ruff.executable = ruff
245 # ruff.executable = ruff
246 # ruff.executable = ruff
247 # ruff.executable = ruff
248 # ruff.executable = ruff
249 # ruff.executable = ruff
250 # ruff.executable = ruff
251 # ruff.executable = ruff
252 # ruff.executable = ruff
253 # ruff.executable = ruff
254 # ruff.executable = ruff
255 # ruff.executable = ruff
256 # ruff.executable = ruff
257 # ruff.executable = ruff
258 # ruff.executable = ruff
259 # ruff.executable = ruff
260 # ruff.executable = ruff
261 # ruff.executable = ruff
262 # ruff.executable = ruff
263 # ruff.executable = ruff
264 # ruff.executable = ruff
265 # ruff.executable = ruff
266 # ruff.executable = ruff
267 # ruff.executable = ruff
268 # ruff.executable = ruff
269 # ruff.executable = ruff
270 # ruff.executable = ruff
271 # ruff.executable = ruff
272 # ruff.executable = ruff
273 # ruff.executable = ruff
274 # ruff.executable = ruff
275 # ruff.executable = ruff
276 # ruff.executable = ruff
277 # ruff.executable = ruff
278 # ruff.executable = ruff
279 # ruff.executable = ruff
280 # ruff.executable = ruff
281 # ruff.executable = ruff
282 # ruff.executable = ruff
283 # ruff.executable = ruff
284 # ruff.executable = ruff
285 # ruff.executable = ruff
286 # ruff.executable = ruff
287 # ruff.executable = ruff
288 # ruff.executable = ruff
289 # ruff.executable = ruff
290 # ruff.executable = ruff
291 # ruff.executable = ruff
292 # ruff.executable = ruff
293 # ruff.executable = ruff
294 # ruff.executable = ruff
295 # ruff.executable = ruff
296 # ruff.executable = ruff
297 # ruff.executable = ruff
298 # ruff.executable = ruff
299 # ruff.executable = ruff
300 # ruff.executable = ruff
301 # ruff.executable = ruff
302 # ruff.executable = ruff
303 # ruff.executable = ruff
304 # ruff.executable = ruff
305 # ruff.executable = ruff
306 # ruff.executable = ruff
307 # ruff.executable = ruff
308 # ruff.executable = ruff
309 # ruff.executable = ruff
310 # ruff.executable = ruff
311 # ruff.executable = ruff
312 # ruff.executable = ruff
313 # ruff.executable = ruff
314 # ruff.executable = ruff
315 # ruff.executable = ruff
316 # ruff.executable = ruff
317 # ruff.executable = ruff
318 # ruff.executable = ruff
319 # ruff.executable = ruff
320 # ruff.executable = ruff
321 # ruff.executable = ruff
322 # ruff.executable = ruff
323 # ruff.executable = ruff
324 # ruff.executable = ruff
325 # ruff.executable = ruff
326 # ruff.executable = ruff
327 # ruff.executable = ruff
328 # ruff.executable = ruff
329 # ruff.executable = ruff
330 # ruff.executable = ruff
331 # ruff.executable = ruff
332 # ruff.executable = ruff
333 # ruff.executable = ruff
334 # ruff.executable = ruff
335 # ruff.executable = ruff
336 # ruff.executable = ruff
337 # ruff.executable = ruff
338 # ruff.executable = ruff
339 # ruff.executable = ruff
340 # ruff.executable = ruff
341 # ruff.executable = ruff
342 # ruff.executable = ruff
343 # ruff.executable = ruff
344 # ruff.executable = ruff
345 # ruff.executable = ruff
346 # ruff.executable = ruff
347 # ruff.executable = ruff
348 # ruff.executable = ruff
349 # ruff.executable = ruff
350 # ruff.executable = ruff
351 # ruff.executable = ruff
352 # ruff.executable = ruff
353 # ruff.executable = ruff
354 # ruff.executable = ruff
355 # ruff.executable = ruff
356 # ruff.executable = ruff
357 # ruff.executable = ruff
358 # ruff.executable = ruff
359 # ruff.executable = ruff
360 # ruff.executable = ruff
361 # ruff.executable = ruff
362 # ruff.executable = ruff
363 # ruff.executable = ruff
364 # ruff.executable = ruff
365 # ruff.executable = ruff
366 # ruff.executable = ruff
367 # ruff.executable = ruff
368 # ruff.executable = ruff
369 # ruff.executable = ruff
370 # ruff.executable = ruff
371 # ruff.executable = ruff
372 # ruff.executable = ruff
373 # ruff.executable = ruff
374 # ruff.executable = ruff
375 # ruff.executable = ruff
376 # ruff.executable = ruff
377 # ruff.executable = ruff
378 # ruff.executable = ruff
379 # ruff.executable = ruff
380 # ruff.executable = ruff
381 # ruff.executable = ruff
382 # ruff.executable = ruff
383 # ruff.executable = ruff
384 # ruff.executable = ruff
385 # ruff.executable = ruff
386 # ruff.executable = ruff
387 # ruff.executable = ruff
388 # ruff.executable = ruff
389 # ruff.executable = ruff
390 # ruff.executable = ruff
391 # ruff.executable = ruff
392 # ruff.executable = ruff
393 # ruff.executable = ruff
394 # ruff.executable = ruff
395 # ruff.executable = ruff
396 # ruff.executable = ruff
397 # ruff.executable = ruff
398 # ruff.executable = ruff
399 # ruff.executable = ruff
400 # ruff.executable = ruff
401 # ruff.executable = ruff
402 # ruff.executable = ruff
403 # ruff.executable = ruff
404 # ruff.executable = ruff
405 # ruff.executable = ruff
406 # ruff.executable = ruff
407 # ruff.executable = ruff
408 # ruff.executable
```

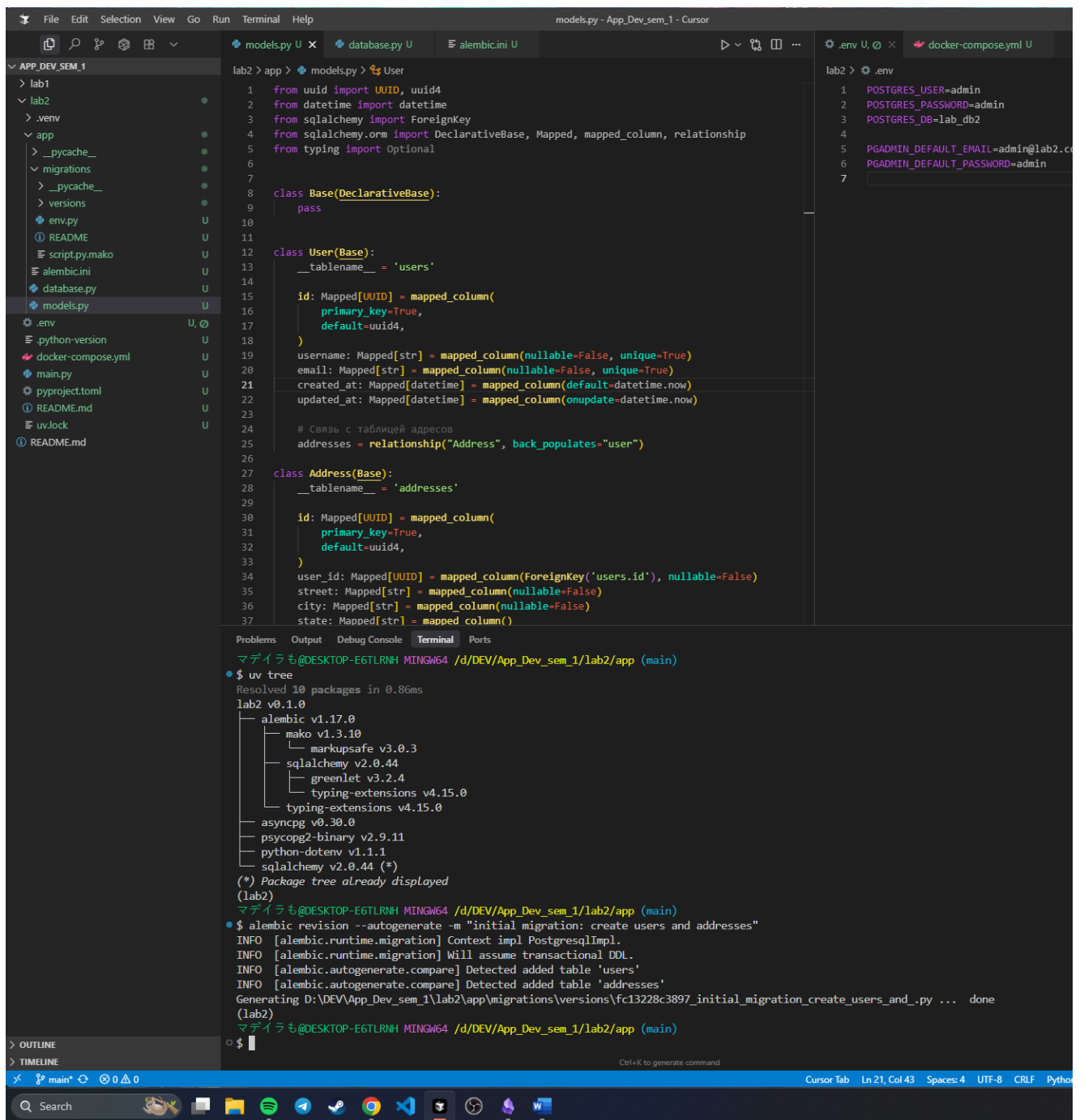
Теперь настраиваем env.py. Сначала импортируем модели, затем

указываем метаданные для автогенерации миграций.



```
lab2 > app > migrations > env.py > run_migrations_offline
1 from logging.config import fileConfig
2
3 from sqlalchemy import engine_from_config
4 from sqlalchemy import pool
5
6 from alembic import context
7
8 from models import Base
9
10 # this is the Alembic Config object, which provides
11 # access to the values within the .ini file in use.
12 config = context.config
13
14 # Interpret the config file for Python logging.
15 # This line sets up loggers basically.
16 if config.config_file_name is not None:
17     fileConfig(config.config_file_name)
18
19 # add your model's Metadata object here
20 # for 'autogenerate' support
21 # from myapp import mymodel
22 # target_metadata = mymodel.Base.metadata
23 target_metadata = Base.metadata
24
25 # other values from the config, defined by the needs of env.py,
26 # can be acquired:
27 # my_important_option = config.get_main_option("my_important_option")
28 # ... etc.
29
30
31 def run_migrations_offline() -> None:
32     """Run migrations in 'offline' mode.
33
34     This configures the context with just a URL
35     and not an Engine, though an Engine is acceptable
36     here as well.  By skipping the Engine creation
37     we don't even need a DBAPI to be available.
38
39     Calls to context.execute() here emit the given string to the
40     script output.
41
42     """
43
44     If the cache and target directories are on different filesystems, hardlinking may not be supported.
45     If this is intentional, set `export UV_LINK_MODE=copy` or use `--link-mode=copy` to suppress this warning.
46
47     Installed 1 package in 10ms
48     + python-dotenv==1.1.1
49     (lab2)
50     マダイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
51     * $ uv sync
52     Resolved 10 packages in 0.77ms
53     Audited 9 packages in 6ms
54     (lab2)
55     マダイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
56     * $ alembic init migrations
57     Creating directory D:\DEV\App_Dev_sem_1\lab2\app\migrations ... done
58     Creating directory D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions ... done
59     Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\alembic.ini ... done
60     Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\env.py ... done
61     Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\README ... done
62     Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\script.py.mako ... done
63     Please edit configuration/connection/logging settings in D:\DEV\App_Dev_sem_1\lab2\app\alembic.ini before proceeding.
64     (lab2)
65     マダイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
66     o $
```

Создаем миграцию



Применяем к БД последнюю миграцию



fc13228c3897\_initial\_migration\_create\_users\_and\_py - App\_Dev\_sem\_1 - Cursor

lab2 > app > migrations > versions > fc13228c3897\_initial\_migration\_create\_users\_and\_py > ...

```
1  """initial migration: create users and addresses
2
3  Revision ID: fc13228c3897
4  Revises:
5  Create Date: 2025-10-20 22:50:33.165354
6
7  """
8  from typing import Sequence, Union
9
10 from alembic import op
11 import sqlalchemy as sa
12
13
14 # revision identifiers, used by Alembic.
15 revision: str = 'fc13228c3897'
16 down_revision: Union[str, Sequence[str], None] = None
17 branch_labels: Union[str, Sequence[str], None] = None
18 depends_on: Union[str, Sequence[str], None] = None
19
20
21 def upgrade() -> None:
22     """Upgrade schema."""
23     # ### commands auto generated by Alembic - please adjust! ###
24     op.create_table('users',
25         sa.Column('id', sa.Uuid(), nullable=False),
26         sa.Column('username', sa.String(), nullable=False),
27         sa.Column('email', sa.String(), nullable=False),
28         sa.Column('created_at', sa.DateTime(), nullable=False),
29         sa.Column('updated_at', sa.DateTime(), nullable=False),
30         sa.PrimaryKeyConstraint('id'),
31         sa.UniqueConstraint('email'),
32         sa.UniqueConstraint('username')
33     )
34     op.create_table('addresses',
35         sa.Column('id', sa.Uuid(), nullable=False),
36         sa.Column('user_id', sa.Uuid(), nullable=False),
37         sa.Column('street', sa.String(), nullable=False),
```

lab2 > .env

```
1  POSTGRES_USER=admin
2  POSTGRES_PASSWORD=admin
3  POSTGRES_DB=lab_db2
4
5  PGADMIN_DEFAULT_EMAIL=admin
6  PGADMIN_DEFAULT_PASSWORD=admin
7
```

Problems Output Debug Console Terminal Ports

```
├── markupsafe v3.0.3
├── sqlalchemy v2.0.44
├── greenlet v3.2.4
├── typing-extensions v4.15.0
├── typing-extensions v4.15.0
├── asyncpg v0.30.0
├── psycopg2-binary v2.9.11
├── python-dotenv v1.1.1
├── sqlalchemy v2.0.44 (*)
(*) Package tree already displayed
(lab2)
マデイラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
• $ alembic revision --autogenerate -m "initial migration: create users and addresses"
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 D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions\fc13228c3897_initial_migration_create_users_and_py ... done
(lab2)
マデイラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
• $ alembic upgrade head
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> fc13228c3897, initial migration: create users and addresses
(lab2)
マデイラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
o $
```

OUTLINE

TIMELINE

Cursor Tab Ln 1, Col 1 Spaces: 4 UTF-8

## Как видим, создались таблицы с верными колонками

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure for 'lab2', including 'Databases (2)', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (1)', 'public', 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (3)'.
- Query Editor:** Contains the query `select * from addresses`.
- Data Output:** Shows the results of the query, including columns: `id` (PK), `user_id`, `street`, `city`, `state`, and `zip`.
- Messages:** Displays a success message: 'Successfully run. Total query runtime: 70 msec. 0 rows affected.'
- Terminal:** Shows the output of the `alembic upgrade` command, indicating that the database schema was successfully updated.

The screenshot shows the pgAdmin 4 interface with the following components:

- Object Explorer:** Displays the database structure for 'lab2', including 'Databases (2)', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (1)', 'public', 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (3)'.
- Query Editor:** Contains the query `select * from alembic_version`.
- Data Output:** Shows the results of the query, including columns: `version_num` (PK), `character varying (32)`, and `fc13228c3897`.
- Messages:** Displays a success message: 'Successfully run. Total query runtime: 70 msec. 0 rows affected.'
- Terminal:** Shows the output of the `alembic upgrade` command, indicating that the database schema was successfully updated.

# НАПОЛНЕНИЕ БАЗЫ ДАННЫМИ

## Создадим фабрику для наполнения и добавим 5 пользователей

```
5
6
7 load_dotenv()
8
9 DATABASE_URL = f"postgresql://{os.getenv('POSTGRES_USER')}:{os.getenv('POSTGRES_PASSWORD')}@localhost:5433/{os.getenv('POSTGRES_DB')}}"
10
11 engine = create_engine(
12     DATABASE_URL,
13     echo=True # Логирование SQL-запросов в консоль
14 )
15
16 session_factory = sessionmaker(engine)
17
18 def get_session():
19     return session_factory()
20
21
22 Ctrl+L to chat, Ctrl+K to generate
```

un Terminal Help push\_data.py - App\_Dev\_sem\_1 - Cursor

database.py models.py push\_data.py x

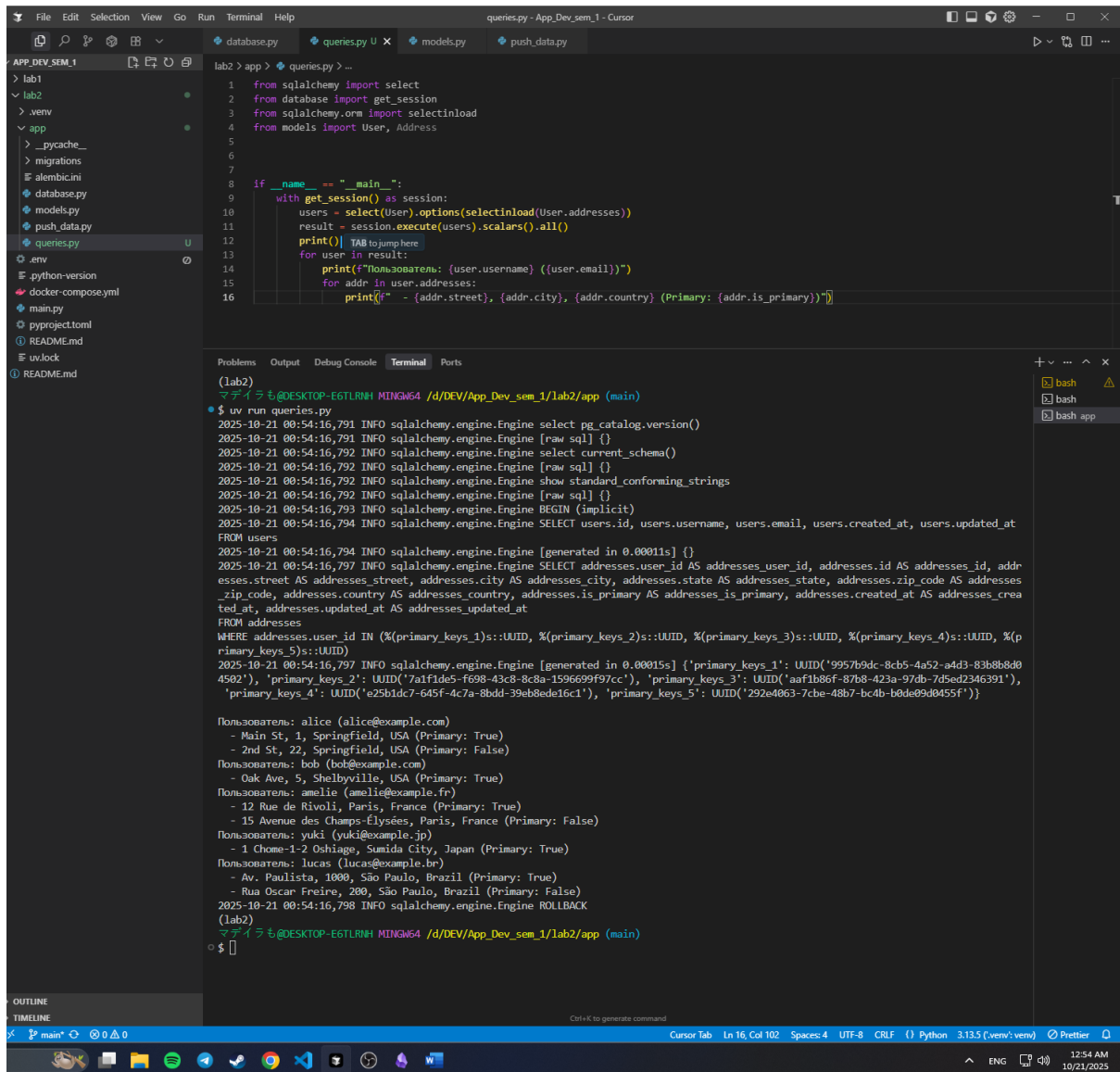
lab2 > app > push\_data.py > ...

```
12 ),
13 User(
14     username="bob",
15     email="bob@example.com",
16     addresses=[
17         Address(street="Oak Ave, 5", city="Shelbyville", state="IL", zip_code="62565", country="USA", is_primary=True),
18     ],
19 ),
20 User(
21     username="amelie",
22     email="amelie@example.fr",
23     addresses=[
24         Address(street="12 Rue de Rivoli", city="Paris", state="Île-de-France", zip_code="75001", country="France", is_primary=True),
25         Address(street="15 Avenue des Champs-Élysées", city="Paris", state="Île-de-France", zip_code="75008", country="France", is_primary=False),
26     ],
27 ),
28 User(
29     username="yuki",
30     email="yuki@example.jp",
31     addresses=[
32         Address(street="1 Chome-1-2 Oshiage", city="Sumida City", state="Tokyo", zip_code="131-0045", country="Japan", is_primary=True),
33     ],
34 ),
35 User(
36     username="lucas",
37     email="lucas@example.br",
38     addresses=[
39         Address(street="Av. Paulista, 1000", city="São Paulo", state="SP", zip_code="01310-100", country="Brazil", is_primary=True),
40         Address(street="Rua Oscar Freire, 200", city="São Paulo", state="SP", zip_code="01426-001", country="Brazil", is_primary=False),
41     ],
42 ),
43 ]
44
45
46 with get_session() as session:
47     session.add_all(users_data)
48     session.commit()
49
50 print('Data added successfully')
51
```

Problems Output Debug Console Terminal Ports

```
ted_at_3': datetime.datetime(2025, 10, 20, 23, 36, 43, 586627), 'street_3': '12 Rue de Rivoli', 'state_3': 'Île-de-France', 'zip_code_3': '75001
'), 'created_at_1': datetime.datetime(2025, 10, 20, 23, 36, 43, 586624), 'street_1': '2nd St, 22', 'state_1': 'IL', 'zip_code_1': '62702', 'is_p
5, 10, 20, 23, 36, 43, 586626), 'country_2': 'USA', 'city_2': 'Shelbyville', 'id_2': UUID('519ac6c3-d67c-44c1-8b2a-f7ce042996af')), 'created_at_2
5, 10, 20, 23, 36, 43, 586626), 'country_2': 'USA', 'city_2': 'Shelbyville', 'id_2': UUID('519ac6c3-d67c-44c1-8b2a-f7ce042996af')), 'created_at_2
62565', 'is_primary_2': True, 'user_id_2': UUID('7a1f1de5-f698-43c8-8c8a-1596699f97cc')), 'updated_at_3': datetime.datetime(2025, 10, 20, 23, 36, 4
ted_at_3': datetime.datetime(2025, 10, 20, 23, 36, 43, 586627), 'street_3': '12 Rue de Rivoli', 'state_3': 'Île-de-France', 'zip_code_3': '75001
62565', 'is_primary_2': True, 'user_id_2': UUID('7a1f1de5-f698-43c8-8c8a-1596699f97cc')), 'updated_at_3': datetime.datetime(2025, 10, 20, 23, 36, 4
ted_at_3': datetime.datetime(2025, 10, 20, 23, 36, 43, 586627), 'street_3': '12 Rue de Rivoli', 'state_3': 'Île-de-France', 'zip_code_3': '75001
ted_at_3': datetime.datetime(2025, 10, 20, 23, 36, 43, 586627), 'street_3': '12 Rue de Rivoli', 'state_3': 'Île-de-France', 'zip_code_3': '75001
ime(2025, 10, 20, 23, 36, 43, 586628), 'country_4': 'France', 'city_4': 'Paris', 'id_4': UUID('4cfb0ebb-845f-4a63-81e0-e36869487888')), 'created_at
ime(2025, 10, 20, 23, 36, 43, 586628), 'country_4': 'France', 'city_4': 'Paris', 'id_4': UUID('4cfb0ebb-845f-4a63-81e0-e36869487888')), 'created_at
Île-de-France', 'zip_code_4': '75008', 'is_primary_4': False, 'user_id_4': UUID('aaf1b86f-87b8-423a-97db-7d5ed2346391')), 'updated_at_5': dateti
ea-73a2-4461-8e71-4f2a5b738126')), 'created_at_5': datetime.datetime(2025, 10, 20, 23, 36, 43, 586630), 'street_5': '1 Chome-1-2 Oshiage', 'state_
c1', 'updated_at_6': datetime.datetime(2025, 10, 20, 23, 36, 43, 586632), 'country_6': 'Brazil', 'city_6': 'São Paulo', 'id_6': UUID('e507190f-
. Paulista, 1000', 'state_6': 'SP', 'zip_code_6': '01310-100', 'is_primary_6': True, 'user_id_6': UUID('292e4063-7cbe-48b7-bc4b-b0de09d0455f')),
'id_7': UUID('85ffd105-0955-4fc8-a8de-678cd380d0d1')), 'created_at_7': datetime.datetime(2025, 10, 20, 23, 36, 43, 586633), 'street_7': 'Rua Oscar
be-48b7-bc4b-b0de09d0455f'))}
2025-10-20 23:36:43,588 INFO sqlalchemy.engine.Engine COMMIT
Data added successfully
(lab2)
マデイラも@DESKTOP-E6TLRNIH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$
```

## Запрос связанных данных



```
File Edit Selection View Go Run Terminal Help
queries.py - App_Dev_seм_1 - Cursor

APP_DEV_SEM_1
├── lab1
├── lab2
├── .venv
├── app
├── __pycache__
├── migrations
├── alembic.ini
├── database.py
├── models.py
├── push_data.py
├── queries.py
├── .env
├── .python-version
├── docker-compose.yml
├── main.py
├── pyproject.toml
├── README.md
└── uv.lock
  README.md

lab2 > app > queries.py > ...
1 from sqlalchemy import select
2 from database import get_session
3 from sqlalchemy.orm import selectinload
4 from models import User, Address
5
6
7
8 if __name__ == "__main__":
9     with get_session() as session:
10         users = select(User).options(selectinload(User.addresses))
11         result = session.execute(users).scalars().all()
12         print() TAB to jump here
13         for user in result:
14             print(f"Пользователь: {user.username} ({user.email})")
15             for addr in user.addresses:
16                 print(f"    - {addr.street}, {addr.city}, {addr.country} (Primary: {addr.is_primary})")

Problems Output Debug Console Terminal Ports
(Lab2)
マディラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_seм_1/lab2/app (main)
$ uv run queries.py
2025-10-21 00:54:16,791 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2025-10-21 00:54:16,791 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-21 00:54:16,792 INFO sqlalchemy.engine.Engine select current_schema()
2025-10-21 00:54:16,792 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-21 00:54:16,792 INFO sqlalchemy.engine.Engine show standard_conforming_strings
2025-10-21 00:54:16,792 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-10-21 00:54:16,793 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-10-21 00:54:16,794 INFO sqlalchemy.engine.Engine SELECT users.id, users.username, users.email, users.created_at, users.updated_at
FROM users
2025-10-21 00:54:16,794 INFO sqlalchemy.engine.Engine [generated in 0.00011s] {}
2025-10-21 00:54:16,797 INFO sqlalchemy.engine.Engine SELECT addresses.user_id AS addresses_user_id, addresses.id AS addresses_id, add
resses.street AS addresses_street, addresses.city AS addresses_city, addresses.state AS addresses_state, addresses.zip_code AS addresses
_zip_code, addresses.country AS addresses_country, addresses.is_primary AS addresses_is_primary, addresses.created_at AS addresses_crea
ted_at, addresses.updated_at AS addresses_updated_at
FROM addresses
WHERE addresses.user_id IN %(primary_keys_1)s::UUID, %(primary_keys_2)s::UUID, %(primary_keys_3)s::UUID, %(primary_keys_4)s::UUID, %(p
rimary_keys_5)s::UUID)
2025-10-21 00:54:16,797 INFO sqlalchemy.engine.Engine [generated in 0.00015s] {'primary_keys_1': UUID('9957b9dc-8cb5-4a52-a4d3-83b8b8d0
4502'), 'primary_keys_2': UUID('7a1f1de5-f698-43c8-8c8a-1596699f97cc'), 'primary_keys_3': UUID('aaf1b86f-87b8-423a-97db-7d5ed2346391'),
'primary_keys_4': UUID('e25b1dc7-645f-4c7a-8bdd-39eb8ede16c1'), 'primary_keys_5': UUID('292e4063-7cbe-48b7-bc4b-b0de09d0455f')}
Пользователь: alice (alice@example.com)
  - Main St, 1, Springfield, USA (Primary: True)
  - 2nd St, 22, Springfield, USA (Primary: False)
Пользователь: bob (bob@example.com)
  - Oak Ave, 5, Shelbyville, USA (Primary: True)
Пользователь: amelie (amelie@example.fr)
  - 12 Rue de Rivoli, Paris, France (Primary: True)
  - 15 Avenue des Champs-Élysées, Paris, France (Primary: False)
Пользователь: yuki (yuki@example.jp)
  - 1 Chome-1-2 Oshiage, Sumida City, Japan (Primary: True)
Пользователь: lucas (lucas@example.br)
  - Av. Paulista, 1000, São Paulo, Brazil (Primary: True)
  - Rua Oscar Freire, 200, São Paulo, Brazil (Primary: False)
2025-10-21 00:54:16,798 INFO sqlalchemy.engine.Engine ROLLBACK
(Lab2)
マディラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_seм_1/lab2/app (main)
$
```

## Последующие работы с БД и миграции

Добавил классы Product и Order, добавил поле description для класса User, добавил недостающие поля связей

```

47     # Обратная связь с таблицей пользователей
48     user = relationship("User", back_populates="addresses")
49
50     class Product(Base):
51         __tablename__ = 'products'
52
53         id: Mapped[UUUID] = mapped_column(
54             primary_key=True,
55             default=uuid4,
56         )
57         name: Mapped[str] = mapped_column(nullable=False)
58         description: Mapped[str] = mapped_column(nullable=True)
59         price: Mapped[float] = mapped_column(nullable=False)
60         stock_quantity: Mapped[int] = mapped_column(nullable=False, default=0)
61         created_at: Mapped[datetime] = mapped_column(default=datetime.now)
62         updated_at: Mapped[Optional[datetime]] = mapped_column(default=datetime.now, onupdate=datetime.now)
63
64         # связь с заказами
65         orders = relationship("Order", back_populates="products")
66
67     class Order(Base):
68         __tablename__ = 'orders'
69
70         id: Mapped[UUUID] = mapped_column(
71             primary_key=True,
72             default=uuid4,
73         )
74         user_id: Mapped[UUUID] = mapped_column(ForeignKey('users.id'), nullable=False)
75         product_id: Mapped[UUUID] = mapped_column(ForeignKey('products.id'), nullable=False)
76         delivery_address_id: Mapped[UUUID] = mapped_column(ForeignKey('addresses.id'), nullable=False)
77
78         quantity: Mapped[int] = mapped_column(nullable=False, default=1)
79         total_price: Mapped[float] = mapped_column(nullable=False)
80         status: Mapped[str] = mapped_column(nullable=False, default="pending")
81         order_date: Mapped[datetime] = mapped_column(default=datetime.now)
82         created_at: Mapped[datetime] = mapped_column(default=datetime.now)
83         updated_at: Mapped[Optional[datetime]] = mapped_column(default=datetime.now, onupdate=datetime.now)
84
85         # Связи с другими таблицами
86         user = relationship("User", back_populates="orders")
87         product = relationship("Product", back_populates="orders")
88         delivery_address = relationship("Address")
89

```

## Создание новой миграции

```

- Av. Paulista, 1000, São Paulo, Brazil (Primary: True)
- Rua Oscar Freire, 200, São Paulo, Brazil (Primary: False)
2025-10-21 00:54:16,798 INFO sqlalchemy.engine.Engine ROLLBACK
(lab2)
マデイラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$ alembic revision --autogenerate -m "create Product, Order. add new fields and relations"
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.autogenerate.compare] Detected NULL on column 'addresses.updated_at'
INFO [alembic.autogenerate.compare] Detected NULL on column 'users.updated_at'
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions\b49dcc308579_create_product_order_add_new_fields_and_.py ... done
(lab2)
マデイラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$

```

## Применяем миграцию

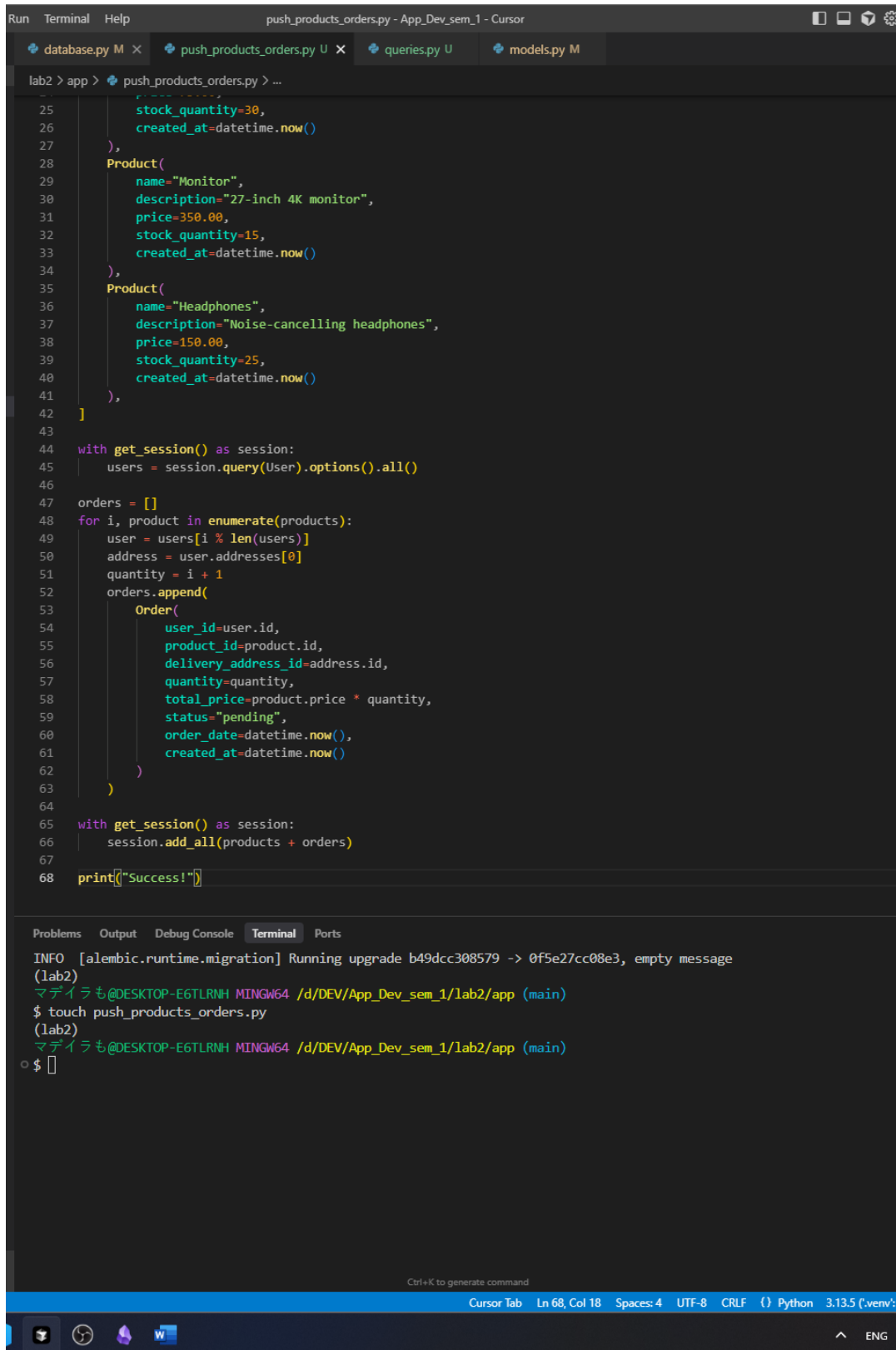
```
INFO [alembic.runtime.migration] Detected added column users.description
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions\0f5e27cc08e3_.py ... done
(lab2)
マデイ ラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
• $ alembic upgrade head
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade b49dcc308579 -> 0f5e27cc08e3, empty message
(lab2)
マデイ ラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
○ $
```

## Видим, что миграция прошла успешно

The screenshot shows a database management interface. On the left, a tree view displays the database schema. The 'users' table is selected, showing its columns: id, username, email, created\_at, updated\_at, and description. On the right, a table view shows the first five rows of data for the 'users' table. The table has columns for an index, email, and creation time.

	mail	creat
	character varying	times
1	lice@example.c...	2025
2	ob@example.com	2025
3	melie@example.fr	2025
4	uki@example.jp	2025
5	icas@exams	2025

## Создал скрипт для заполнения новых таблиц данными



The screenshot shows a code editor with a Python script named `push_products_orders.py`. The script defines two product models, `Monitor` and `Headphones`, and creates a list of orders based on these products. It uses SQLAlchemy to interact with a database session.

```
25         stock_quantity=30,
26         created_at=datetime.now()
27     ),
28     Product(
29         name="Monitor",
30         description="27-inch 4K monitor",
31         price=350.00,
32         stock_quantity=15,
33         created_at=datetime.now()
34     ),
35     Product(
36         name="Headphones",
37         description="Noise-cancelling headphones",
38         price=150.00,
39         stock_quantity=25,
40         created_at=datetime.now()
41     ),
42 ]
43
44 with get_session() as session:
45     users = session.query(User).options().all()
46
47 orders = []
48 for i, product in enumerate(products):
49     user = users[i % len(users)]
50     address = user.addresses[0]
51     quantity = i + 1
52     orders.append(
53         Order(
54             user_id=user.id,
55             product_id=product.id,
56             delivery_address_id=address.id,
57             quantity=quantity,
58             total_price=product.price * quantity,
59             status="pending",
60             order_date=datetime.now(),
61             created_at=datetime.now()
62         )
63     )
64
65 with get_session() as session:
66     session.add_all(products + orders)
67
68 print("Success!")
```

The terminal output shows the execution of the script:

```
INFO [alembic.runtime.migration] Running upgrade b49dcc308579 -> 0f5e27cc08e3, empty message
(lab2)
マデイラも@DESKTOP-E6TLRINH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$ touch push_products_orders.py
(lab2)
マデイラも@DESKTOP-E6TLRINH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
o $
```

The status bar at the bottom indicates the cursor is at line 68, column 18, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python 3.13.5 environment.

Доработал `get_session()` обернув в декоратор контекстного менеджера

```
12 engine = create_engine(  
13     DATABASE_URL,  
14     echo=True # Логирование SQL-запросов в консоль  
15 )  
16  
17 session_factory = sessionmaker(engine)  
18  
19 @contextmanager  
20 def get_session():  
21     session = session_factory()  
22     try:  
23         yield session  
24         session.commit()  
25     except:  
26         session.rollback()  
27         raise  
28     finally:  
29         session.close()  
30  
31  
32 | Ctrl+L to chat, Ctrl+Shift+K to toggle
```



The screenshot displays a development environment with two main windows. The left window shows the DBeaver interface with a project named 'admin@lab2.com (internal)' and a database named 'lab\_db2'. A query is executed in the 'Query History' tab, showing a SQL statement that selects from a table. The right window shows the 'Data Output' tab, displaying a table with 5 rows and 5 columns. The columns are 'id', 'user\_id', 'product\_id', 'delivery\_address\_id', and 'status'. The rows contain numerical data for each column. The bottom status bar indicates 'Total rows: 5' and 'Query complete 00:00:03.035'.

The screenshot displays a web application interface for an 'Admin' dashboard. The left sidebar contains a navigation menu with categories like 'Users', 'Products', 'Orders', and 'Reports'. The main content area shows a table of products with columns for ID, Name, Description, Price, Stock, and Status. A right-hand panel is open, showing a SQL query editor and a data output table. The query is a complex JOIN statement involving tables like 'products', 'users', 'orders', and 'inventory'. The data output table shows the results of this query, with columns for product details and user information. The interface is modern and clean, with a dark theme.

Ответы на вопросы:

**1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?**

Ответ: В SQLAlchemy есть два основных подхода к маппингу:

Декларативный описывает структуру таблиц и связей прямо в теле класса, строится на наследовании от базового класса, удобен для новых проектов и быстрого прототипирования.

Императивный (классический) использует функцию `mapper()` для привязки классов к отдельным объектам `Table`, подходит для сложных или импортируемых схем.

Для новых проектов чаще всего рекомендуют декларативный подход – он наглядней и выглядит лаконичней (большинство современных библиотек рекомендует его). Классический стиль же, когда требуется сложная и тонкая настройка маппинга.

**2. Как Alembic отслеживает текущую версию базы данных?**

Ответ: Alembic сохраняет информацию о текущей схеме в служебной таблице `alembic_version`. При применении миграций Alembic сравнивает метаданные моделей (`target_metadata`) с фактической структурой базы и генерирует скрипты миграций. Каждая миграция получает уникальный идентификатор (`revision ID`).

Таблица `alembic_version` содержит этот ID последней успешно выполненной миграции. При выполнении `alembic upgrade Alembic` находит неприменённые миграции по цепочке `down_revision—revision`, выполняет их последовательно и обновляет запись в `alembic_version`.

**3. Какие типы связей между таблицами вы реализовали в данной работе?**

Ответ: Один ко многим (пользователь, адрес или пользователь заказ – многие к одному обратная связь этой), Один к одному (между заказом и адресом доставки, реализована через внешний ключ)

#### **4. Что такое миграция базы данных и почему она важна?**

Ответ: миграция базы данных - это скрипт, который описывает изменения в структуре таблиц (создание, удаление, изменение столбцов, связей и индексов). Миграции гарантируют, что изменения схемы версионизируются и могут быть автоматически применены или откатаны.

#### **5. Как обрабатываются отношения многие-ко-многим в SQLAlchemy?**

Ответ: отношение «многие ко многим» реализуется через вспомогательную (association) таблицу, содержащую два внешних ключа. В SQLAlchemy её можно определить либо как отдельную ORM-модель, либо как экземпляр Table (я мог так реализовать связь между заказами и продуктами, но не стал)

#### **6. Каков порядок действий при возникновении конфликта версий в Alembic?**

Ответ: Для диагностики миграций Alembic необходимо проверить текущий `revision_id` в таблице `alembic_version` и сравнить его с миграциями в папке `migrations/versions`. В случае необходимости следует выполнить откат до нужного состояния базы с помощью команды `alembic downgrade <revision_id>` или до базового состояния через `alembic downgrade base`. После этого стоит проверить и при необходимости исправить зависимости между миграциями, объединив или переименовав конфликтующие файлы. При значительных изменениях может потребоваться удалить проблемные миграции, сгенерировать новые с помощью `alembic revision --autogenerate` и применить их повторно командой `alembic upgrade head`. Важно убедиться, что таблица `alembic_version` обновилась, и структура базы данных соответствует моделям. Если в проекте появились несколько веток миграций, их следует объединить с помощью `alembic merge heads`, создав новую миграцию для разрешения конфликта.