

Лабораторная работа №2: Работа с SQLAlchemy и alembic

РИМ-150950

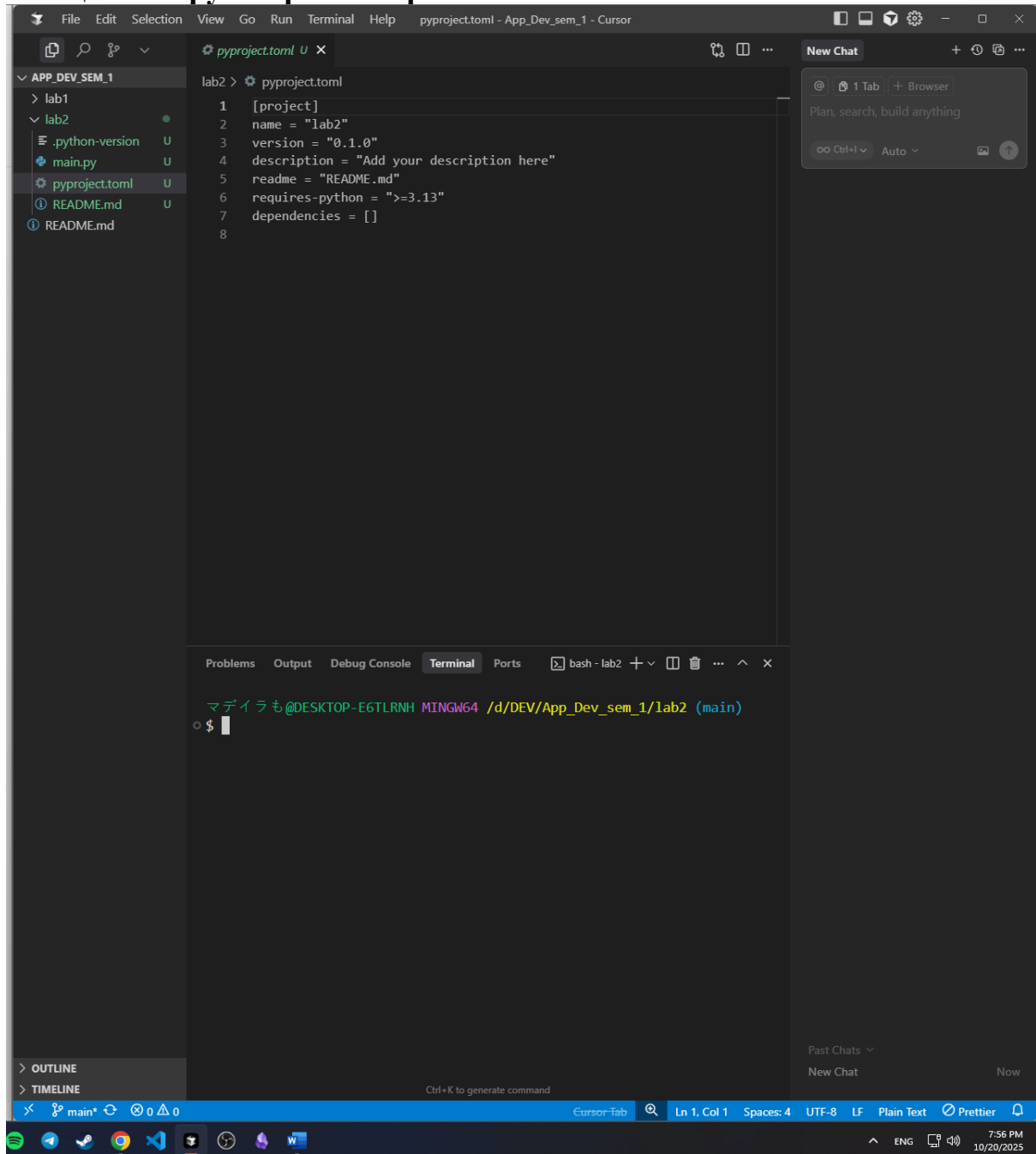
Жунёв Андрей

Цель работы

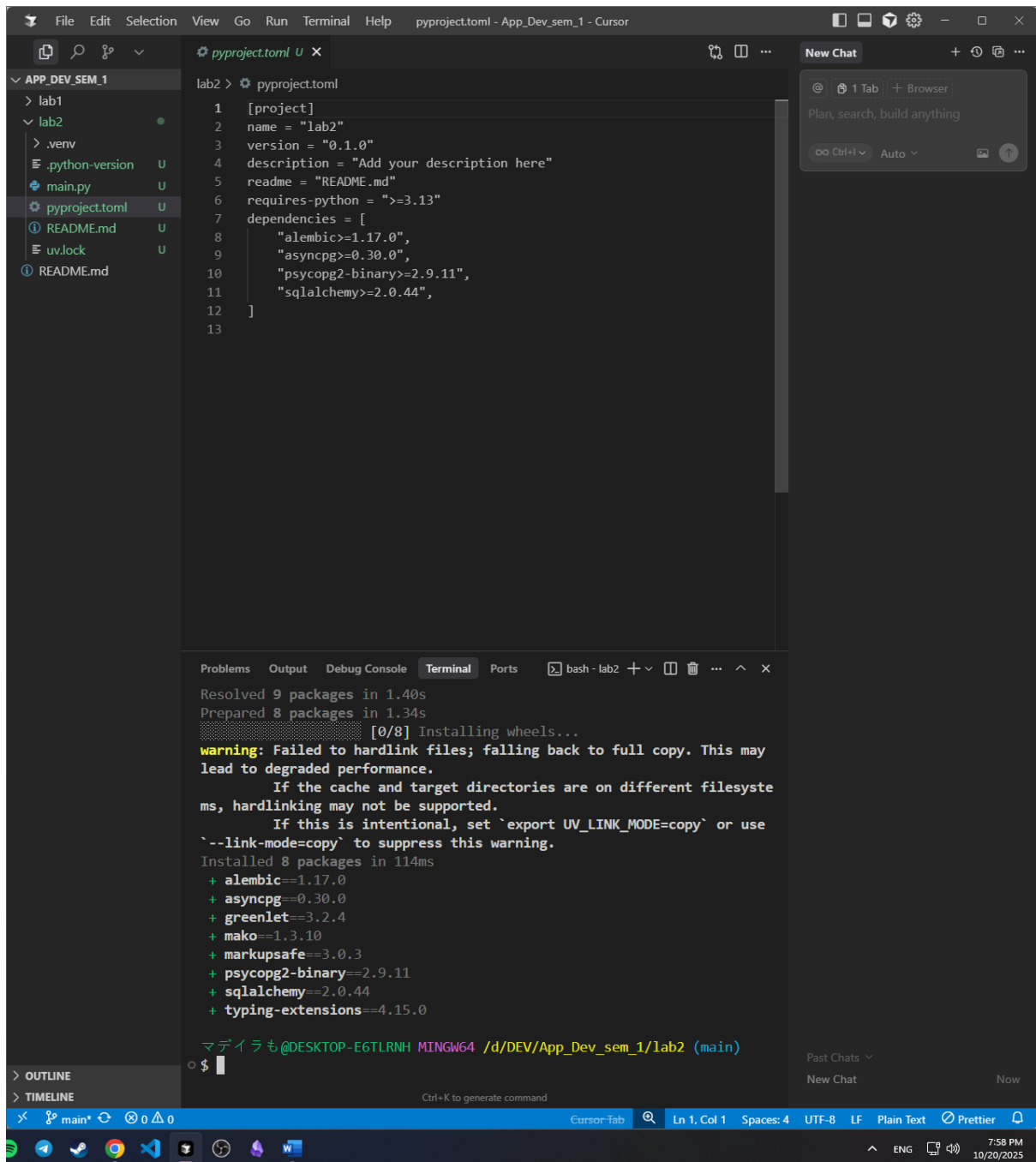
Освоить принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучить механизмы миграции базы данных.

Ход работы:

Инициализируем проект через UV



Устанавливаем необходимые библиотеки



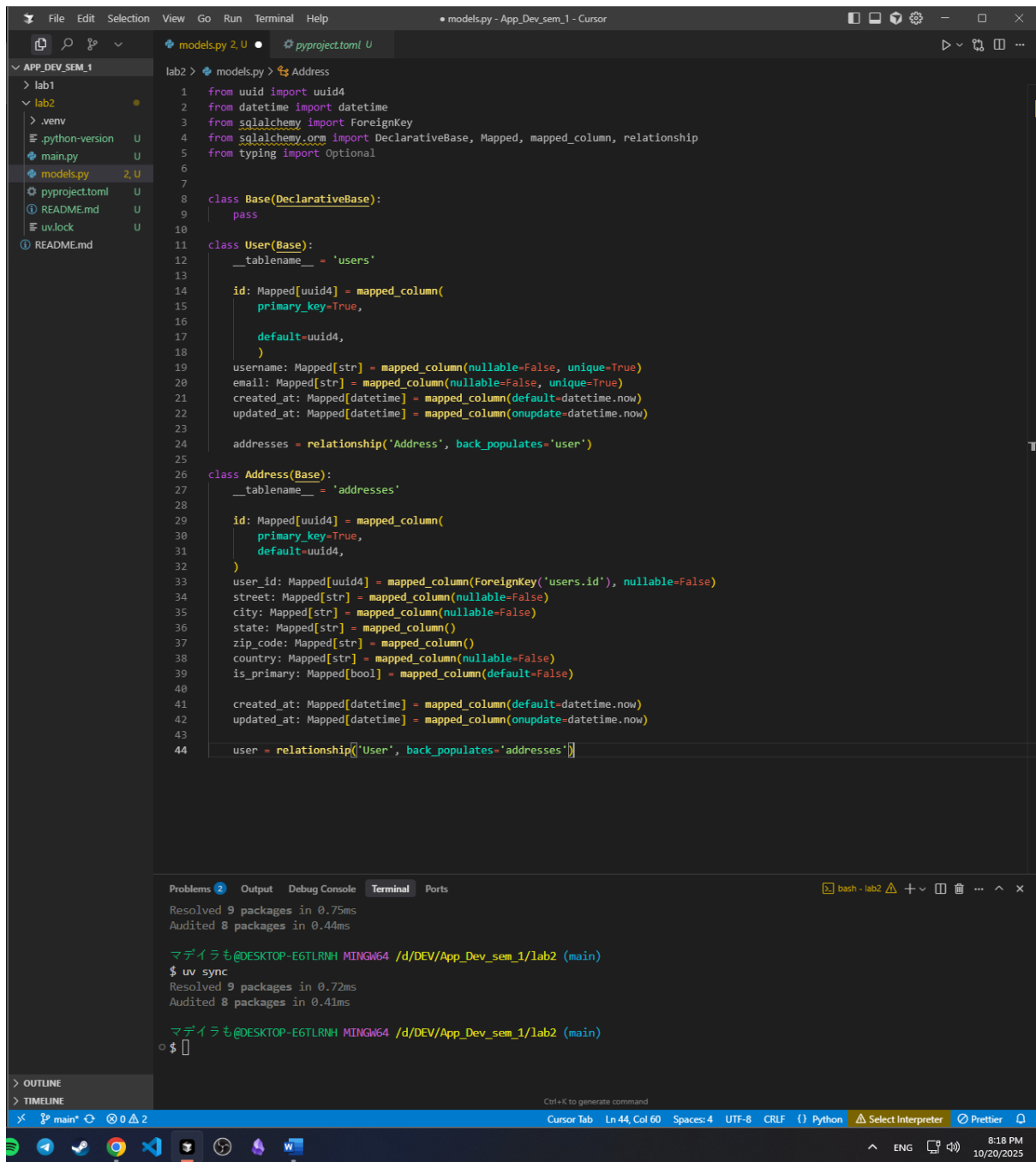
SQLAlchemy – основная библиотека для работы с базами данных через ORM

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

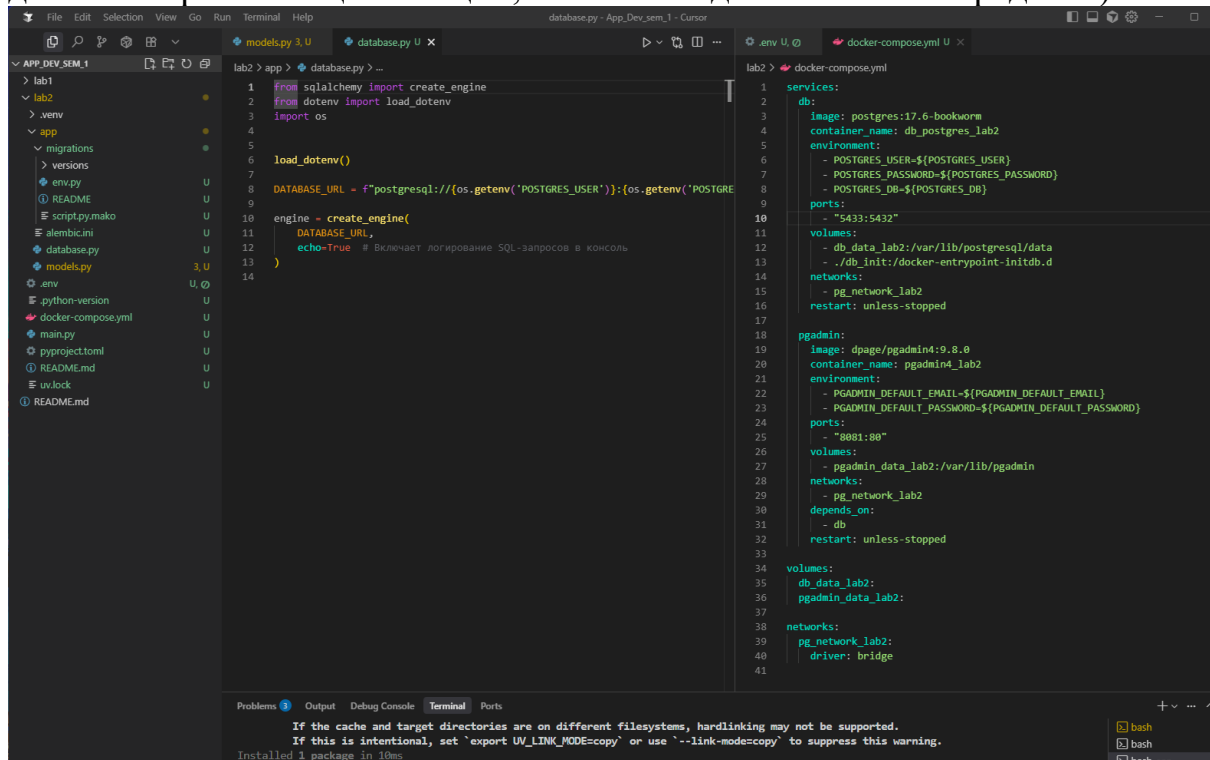


.env файл содержит следующие переменные:

POSTGRES_USER=admin
POSTGRES_PASSWORD=admin
POSTGRES_DB=lab_db2

PGADMIN_DEFAULT_EMAIL=admin@lab2.com
PGADMIN_DEFAULT_PASSWORD=admin

БД развернута. Таблицы заранее не создаются (сначала создал заранее, добавив скрипт инициализации, не понял задание – затем переделал)



The screenshot shows a code editor with two files open: `database.py` and `docker-compose.yml`. The `database.py` file contains Python code for setting up a database engine and connecting to a PostgreSQL database. The `docker-compose.yml` file defines the services for the database and pgadmin.

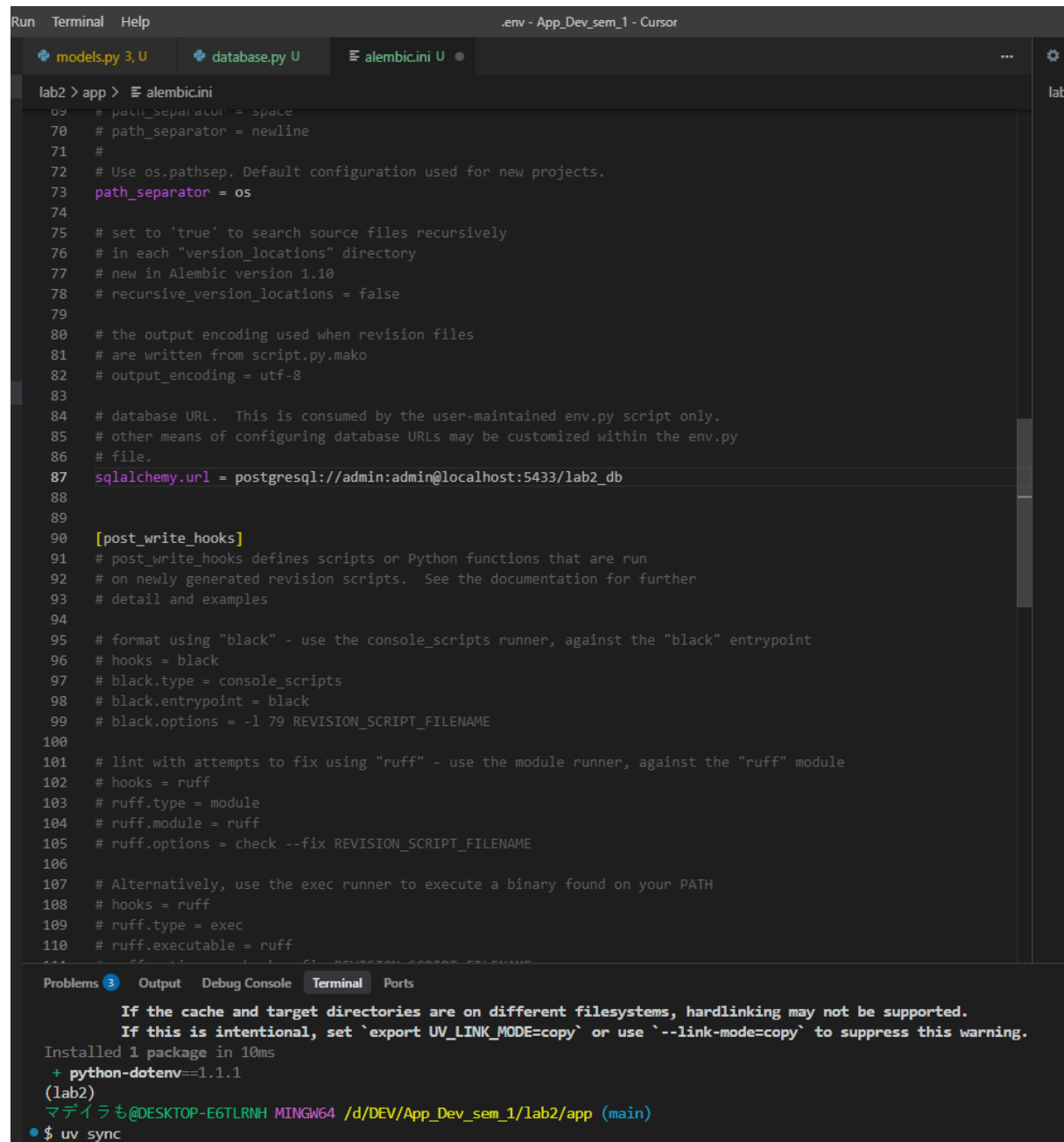
```
1 from sqlalchemy import create_engine
2 from dotenv import load_dotenv
3 import os
4
5 load_dotenv()
6
7 DATABASE_URL = f"postgresql://{os.getenv('POSTGRES_USER')}:{os.getenv('POSTGRES_PASSWORD')}@localhost:5432/{os.getenv('POSTGRES_DB')}"
8
9 engine = create_engine(
10     DATABASE_URL,
11     echo=True # Включает логирование SQL-запросов в консоль
12 )
```

```
1 services:
2   db:
3     image: postgres:17.6-bookworm
4     container_name: db_postgres_lab2
5     environment:
6       - POSTGRES_USER=${POSTGRES_USER}
7       - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
8       - POSTGRES_DB=${POSTGRES_DB}
9     ports:
10       - "5433:5432"
11     volumes:
12       - db_data_lab2:/var/lib/postgresql/data
13       - ./db_init:/docker-entrypoint-initdb.d
14     networks:
15       - pg_network_lab2
16     restart: unless-stopped
17
18   pgadmin:
19     image: dpage/pgadmin4:9.8.0
20     container_name: pgadmin4_lab2
21     environment:
22       - PGADMIN_DEFAULT_EMAIL=${PGADMIN_DEFAULT_EMAIL}
23       - PGADMIN_DEFAULT_PASSWORD=${PGADMIN_DEFAULT_PASSWORD}
24     ports:
25       - "8081:80"
26     volumes:
27       - pgadmin_data_lab2:/var/lib/pgadmin
28     networks:
29       - pg_network_lab2
30     depends_on:
31       - db
32     restart: unless-stopped
33
34 volumes:
35   db_data_lab2:
36   pgadmin_data_lab2:
37
38 networks:
39   pg_network_lab2:
40     driver: bridge
```

The terminal at the bottom shows a warning message: "If the cache and target directories are on different filesystems, hardlinking may not be supported. If this is intentional, set 'export UV_LINK_MODE=copy' or use '--link-mode=copy' to suppress this warning. Installed 1 package in 10ms".

Настройка миграций с Alembic

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

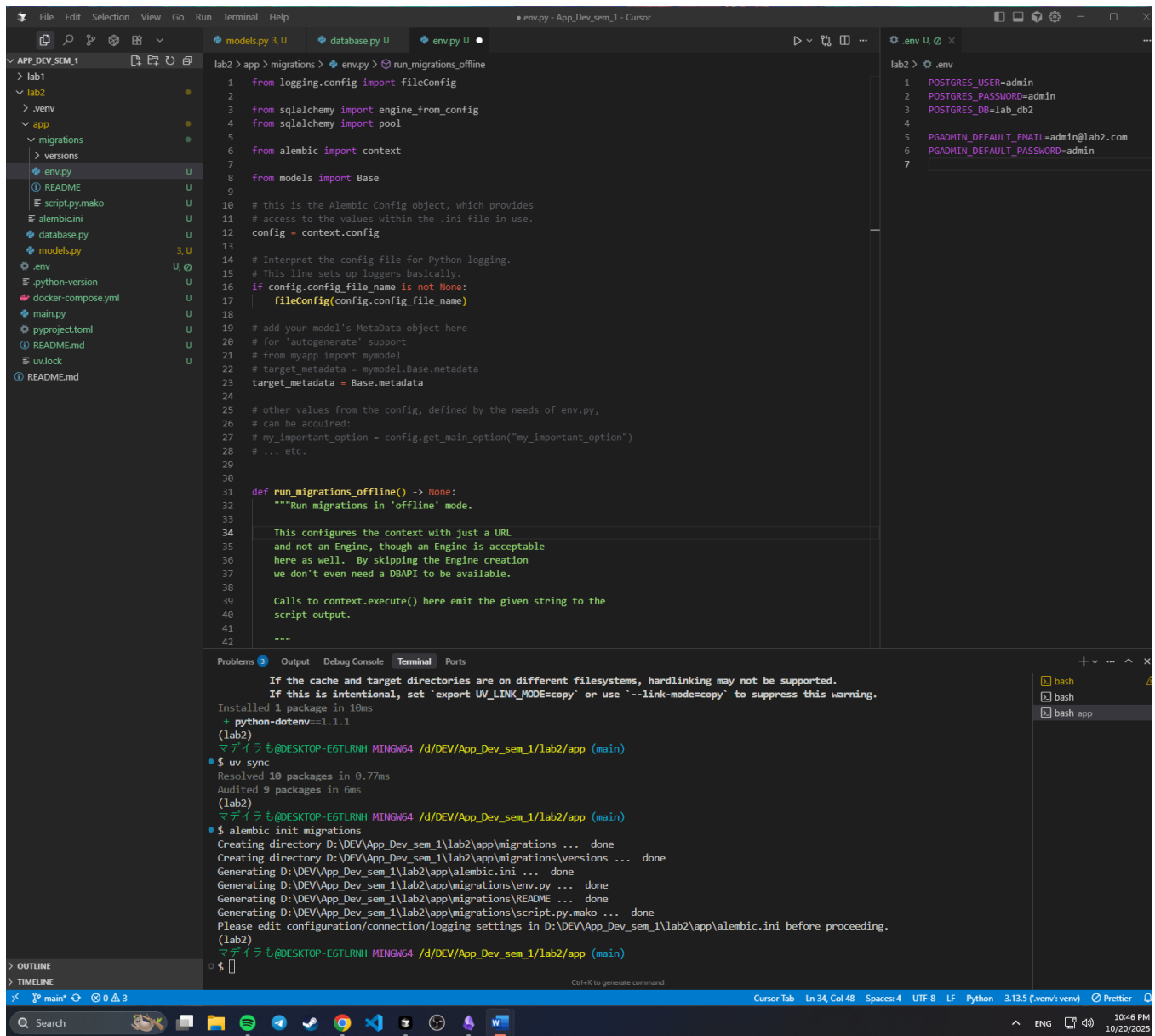


The screenshot shows a code editor with the file `alembic.ini` open. The file contains configuration for Alembic, including database URL, path separators, and hooks. The database URL is set to `postgresql://admin:admin@localhost:5433/lab2_db`. Below the code editor, a terminal window is visible, showing the output of a command. The terminal output indicates that the cache and target directories are on different filesystems, and a warning is displayed. The terminal also shows the installation of `python-dotenv==1.1.1` and the execution of `uv sync`.

```
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.options = check --fix REVISION_SCRIPT_FILENAME
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
Problems 3 Output Debug Console Terminal Ports
If the cache and target directories are on different filesystems, hardlinking may not be supported.
If this is intentional, set `export UV_LINK_MODE=copy` or use `--link-mode=copy` to suppress this warning.
Installed 1 package in 10ms
+ python-dotenv==1.1.1
(lab2)
マデイ ラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$ uv sync
```

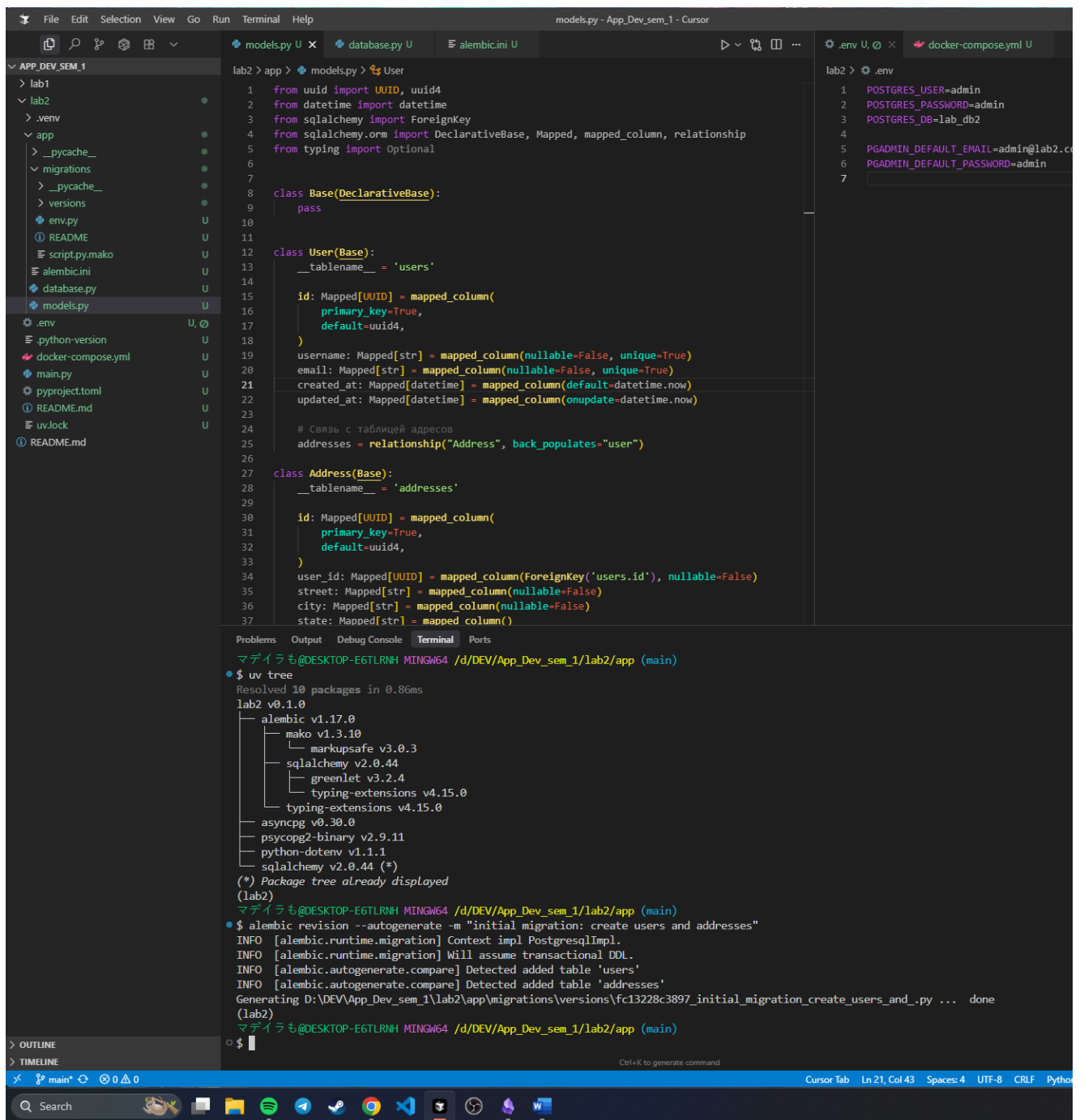
Теперь настраиваем env.py. Сначала импортируем модели, затем указываем метаданные для автогенерации миграций.



```
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     """
```

```
Problems Output Debug Console Terminal Ports
If the cache and target directories are on different filesystems, hardlinking may not be supported.
If this is intentional, set `export UV_LINK_MODE=copy` or use `--link-mode=copy` to suppress this warning.
Installed 1 package in 10ms
+ python-dotenv==1.1.1
(1ab2)
マデイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$ uv sync
Resolved 10 packages in 0.77ms
Audited 9 packages in 6ms
(1ab2)
マデイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$ alembic init migrations
Creating directory D:\DEV\App_Dev_sem_1\lab2\app\migrations ... done
Creating directory D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions ... done
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\alembic.ini ... done
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\env.py ... done
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\README ... done
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\script.py.mako ... done
Please edit configuration/connection/logging settings in D:\DEV\App_Dev_sem_1\lab2\app\alembic.ini before proceeding.
(1ab2)
マデイラも@DESKTOP-EGTLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$
```

Создаем миграцию



Применяем к БД последнюю миграцию

fc13228c3897_initial_migration_create_users_and_py - App_Dev_sem_1 - Cursor

models.py U fc13228c3897_initial_migration_create_users_and_py U database.py U

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)
$
```

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. The left sidebar displays the database structure, including the 'public' schema and the 'addresses' table. The central pane shows the query editor with the query 'select * from addresses'. The bottom pane shows the Data Output, Messages, and Notifications. The Data Output pane displays the results of the query, showing columns: id, user_id, street, city, state, zip_code. The Messages pane shows a successful execution message: 'Successfully run. Total query runtime: 70 msec. 0 rows affected. Query complete 00:00:00.070'. The Notifications pane shows the query results.

id	user_id	street	city	state	zip_code
1	fc13228c3897				

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, including the 'public' schema and the 'alembic_version' table. The central pane shows the query editor with the query 'select * from alembic_version'. The bottom pane shows the Data Output, Messages, and Notifications. The Data Output pane displays the results of the query, showing columns: version_num. The Messages pane shows a successful execution message: 'Successfully run. Total query runtime: 70 msec. 0 rows affected. Query complete 00:00:00.070'. The Notifications pane shows the query results.

version_num
1

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-E6TLRNH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
$
```

Запрос связанных данных

The screenshot shows a VS Code editor with a project named 'APP_DEV_SEM_1'. The file explorer on the left shows a directory structure with files like 'alembic.ini', 'database.py', 'models.py', 'push_data.py', and 'queries.py'. The 'queries.py' file is open in the editor, showing a Python script that uses SQLAlchemy to query a database. The script imports 'select', 'get_session', 'selectinload', and 'User', 'Address' from the respective modules. It defines a main function that gets a session, selects users with their addresses, and prints the results in a formatted string.

```
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()
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})")
```

The terminal output shows the execution of the script. It starts with a prompt to run 'uv run queries.py'. The output shows the SQLAlchemy engine logs, including the SQL query and the results of the query. The results are formatted as follows:

```
Пользователь: alice (alice@example.com)
  - Main St, 1, Springfield, USA (Primary: True)
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)
```

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

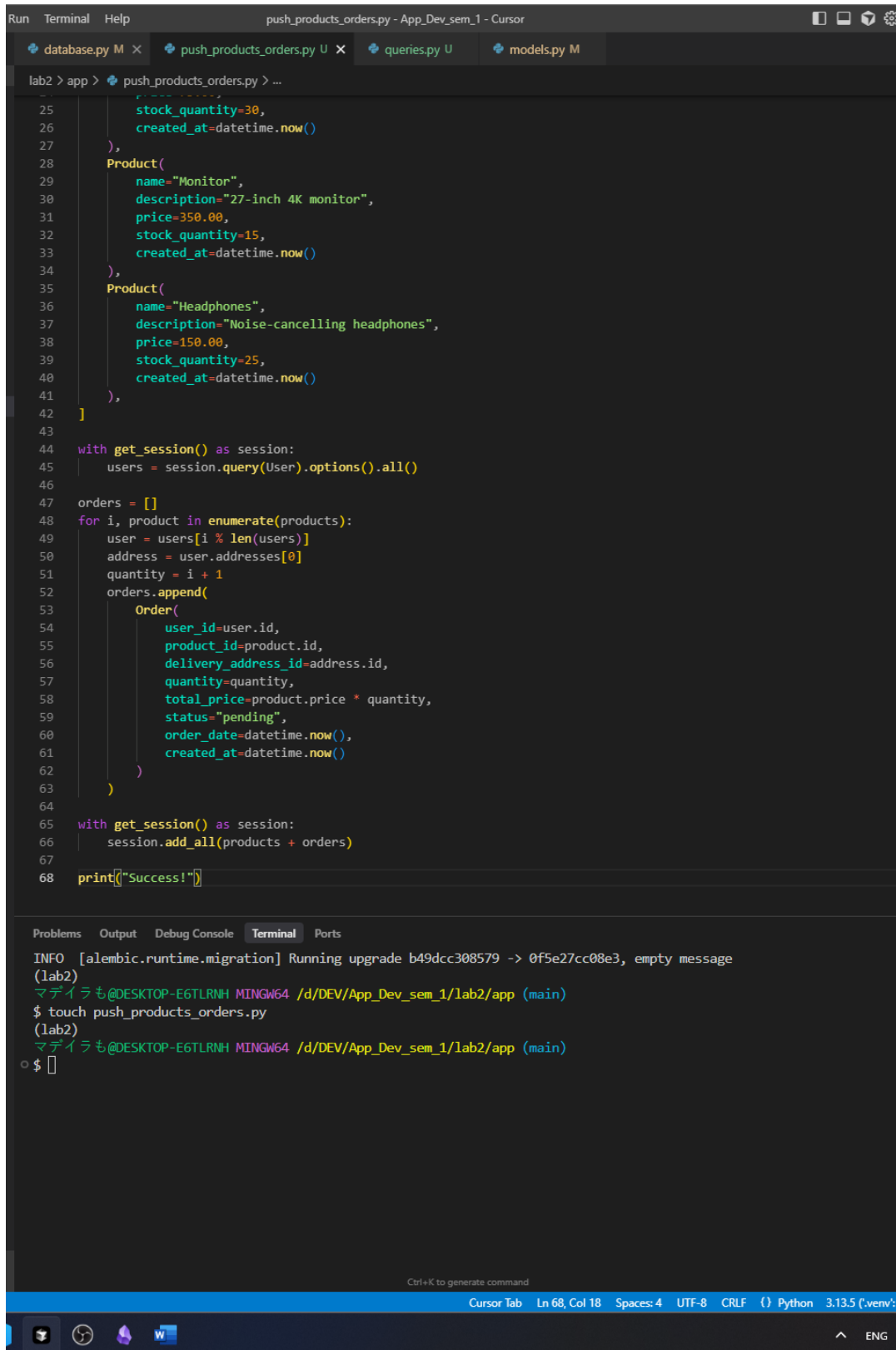
```
INFO [alembic.runtime.migration] Detected added column 'users.description'
Generating D:\DEV\App_Dev_sem_1\lab2\app\migrations\versions\0f5e27cc08e3_... done
(lab2)
マデイ ラも@DESKTOP-E6TLRHH MINGW64 /d/DEV/App_Dev_sem_1/lab2/app (main)
• $ alembic upgrade head
INFO [alembic.runtime.migration] Context 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 is designed to populate a database with products and orders. It uses SQLAlchemy for database operations. The script defines two product models: `Monitor` and `Headphones`, and an `Order` model. It then creates a session, queries for users, and iterates over the products to create orders for each user. The script also includes a final session commit and a success message.

```
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 following commands and messages:

```
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 file is `push_products_orders.py`, line 68, column 18, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.5, and a virtual 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
```


Успешно добавлены 5 заказов и 5 продуктов

The screenshot displays a web application interface. On the left, a sidebar contains navigation links for various sections. The main content area shows a table with 5 rows of data, representing orders and products. The right-hand panel features a code editor with a script that appears to be a REST client or a data manipulation tool, showing a successful response for a POST request.

ID	name	description	category	price	stock_quantity	created_at
1	High-performance laptop	High-performance laptop	Laptop	1200	10	2025-10-21 01:46:17.439060
2	Wireless mouse	Wireless mouse	Mouse	25.0	20	2025-10-21 01:46:17.439060
3	Mechanical keyboard	Mechanical keyboard	Keyboard	75	10	2025-10-21 01:46:17.439060
4	27-inch 4K monitor	27-inch 4K monitor	Monitor	750	10	2025-10-21 01:46:17.439060
5	Noise-cancelling headphones	Noise-cancelling headphones	Headphones	150	25	2025-10-21 01:46:17.439060

This screenshot shows the same web application interface as the previous one, but with a different set of data in the table. The table now contains 5 rows of data, representing orders and products. The right-hand panel shows a code editor with a script that appears to be a REST client or a data manipulation tool, showing a successful response for a POST request.

ID	name	description	category	price	stock_quantity	created_at
1	High-performance laptop	High-performance laptop	Laptop	1200	10	2025-10-21 01:46:17.439060
2	Wireless mouse	Wireless mouse	Mouse	25.0	20	2025-10-21 01:46:17.439060
3	Mechanical keyboard	Mechanical keyboard	Keyboard	75	10	2025-10-21 01:46:17.439060
4	27-inch 4K monitor	27-inch 4K monitor	Monitor	750	10	2025-10-21 01:46:17.439060
5	Noise-cancelling headphones	Noise-cancelling headphones	Headphones	150	25	2025-10-21 01:46:17.439060

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 версионированы 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.