

Цель работы: Познакомиться со способами тестирования приложения.

Основа: Лабораторная работа выполняется на основе ЛРЗ, в репозитории после выполнения добавляем тэг `lab_4`.

Выполнил: Жунёв Андрей Александрович РИМ-150950

1. Подготовка моделей

Выполнен переход от связи один-к-одному к связи многие-ко-многим между заказами и продуктами. Создана промежуточная модель `OrderItem`, которая связывает заказы и продукты.

1. Изменения в моделях:

- Создана модель `OrderItem` с полями:
 - `id` (`UUID`) — первичный ключ
 - `order_id` (`UUID`) — внешний ключ к таблице `orders`
 - `product_id` (`UUID`) — внешний ключ к таблице `products`
 - `quantity` (`int`) — количество товара
 - `price_at_order` (`float`) — цена на момент заказа
 - `created_at` (`datetime`) — дата создания
- Обновлена модель `Order`:
 - Удалены поля `product_id` и `quantity` (перенесены в `OrderItem`)
 - Сохранено поле `total_price` (будет вычисляться из элементов заказа)
 - Добавлена связь `items` с каскадным удалением
 - Обновлена модель `Product`:
 - Изменена связь с `orders` на `order_items`

```

class OrderItem(Base):
    """Промежуточная таблица для связи Order и Product (many-to-many)."""
    __tablename__ = 'order_items'

    id: Mapped[UUID] = mapped_column(
        primary_key=True,
        default=uuid4,
    )
    order_id: Mapped[UUID] = mapped_column(ForeignKey('orders.id'), nullable=False)
    product_id: Mapped[UUID] = mapped_column(ForeignKey('products.id'), nullable=False)
    quantity: Mapped[int] = mapped_column(nullable=False, default=1)
    price_at_order: Mapped[float] = mapped_column(nullable=False) # Цена на момент заказа
    created_at: Mapped[datetime] = mapped_column(default=datetime.now)

    # Связи
    order = relationship("Order", back_populates="items")
    product = relationship("Product", back_populates="order_items")

class Order(Base):
    __tablename__ = 'orders'

    id: Mapped[UUID] = mapped_column(
        primary_key=True,
        default=uuid4,
    )
    user_id: Mapped[int] = mapped_column(ForeignKey('users.id'), nullable=False)
    delivery_address_id: Mapped[UUID] = mapped_column(ForeignKey('addresses.id'), nullable=False)

    total_price: Mapped[float] = mapped_column(nullable=False)
    status: Mapped[str] = mapped_column(nullable=False, default="pending")
    order_date: Mapped[datetime] = mapped_column(default=datetime.now)
    created_at: Mapped[datetime] = mapped_column(default=datetime.now)
    updated_at: Mapped[Optional[datetime]] = mapped_column(default=datetime.now, onupdate=datetime.now)

    # Связи с другими таблицами
    user = relationship("User", back_populates="orders")
    delivery_address = relationship("Address")
    items = relationship("OrderItem", back_populates="order", cascade="all, delete-orphan")

```

INFO [alembic.autogenerate.compare] Detected removed foreign key (product_id)(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders.quantity'
INFO [alembic.autogenerate.compare] Detected removed column 'orders.product_id'
Generating /Users/madeira/DEV//first_sem/App_Dev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_to_support_multiple_products.py ... done
 (lab2) + app git:(main) ✘]

2. Исправление типов данных

Приведены типы внешних ключей к пользователям в соответствие с миграцией:

- Address.user_id: UUID → int
- Order.user_id: UUID → int

3. Создание и применение миграции базы данных

Создана миграция Alembic eeb0a18a7fdf_refactor_order_to_support_multiple_products, которая:

- Создает таблицу order_items с необходимыми полями и внешними ключами
- Удаляет колонки product_id и quantity из таблицы orders
- Удаляет внешний ключ orders_product_id_fkey

Запустил контейнеры с БД и приложением

```

lab2 > app git:(main) ✘ docker-compose ps
      COMMAND           SERVICE    CREATED          STATUS    PORTS
      app              app        2 minutes ago   Up 2 minutes   0.0.0.0:8000->8000/tcp
      db_postgres      db        2 minutes ago   Up 2 minutes   0.0.0.0:5432->5432/tcp, [::]:5433->5432/tcp
      pgadmin4_lab2   pgadmin   2 minutes ago   Up 2 minutes   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp

```

Models.py content:

```

1 from uuid import UUID, 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 class Base(DeclarativeBase):
8     pass
9
10 class User(Base):
11     __tablename__ = 'users'
12
13     id: Mapped[int] = mapped_column(
14         primary_key=True,
15         autoincrement=True,
16     )
17
18     username: Mapped[str] = mapped_column(nullable=False, unique=True)
19     email: Mapped[str] = mapped_column(nullable=False, unique=True)
20     description: Mapped[Optional[str]] = mapped_column(nullable=True) # новое поле
21     created_at: Mapped[datetime] = mapped_column(default=datetime.now)
22     updated_at: Mapped[Optional[datetime]] = mapped_column(default=datetime.now, onupdate=datetime.now)
23
24     # Связь с таблицей адресов

```

Terminal output:

```

lab2 > app git:(main) ✘ uv run alembic revision --autogenerate -m "refactor_order_to_support_multiple_products"
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Config is configured to use SQLite.
INFO [alembic.autogenerate.compare] Detected added table 'order_items'.
INFO [alembic.autogenerate.compare] Detected removed foreign key 'product_id'(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders_qty' on table orders
INFO [alembic.autogenerate.compare] Generating /Users/zadniera/DEV/first_sem/appDev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_
to_support_multiple_.py ...
done

```

Выполнил миграцию после обновления моделей

```

lab2 > app git:(main) ✘ uv run alembic revision --autogenerate -m "refactor_order_to_support_multiple_products"
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Config is configured to use SQLite.
INFO [alembic.autogenerate.compare] Detected added table 'order_items'.
INFO [alembic.autogenerate.compare] Detected removed foreign key 'product_id'(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders_qty' on table orders
INFO [alembic.autogenerate.compare] Generating /Users/zadniera/DEV/first_sem/appDev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_
to_support_multiple_.py ...
done

```

Models.py content:

```

1 from uuid import UUID, 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 class Base(DeclarativeBase):
8     pass
9
10 class User(Base):
11     __tablename__ = 'users'
12
13     id: Mapped[int] = mapped_column(
14         primary_key=True,
15         autoincrement=True,
16     )
17
18     username: Mapped[str] = mapped_column(nullable=False, unique=True)
19     email: Mapped[str] = mapped_column(nullable=False, unique=True)
20     description: Mapped[Optional[str]] = mapped_column(nullable=True) # новое поле
21     created_at: Mapped[datetime] = mapped_column(default=datetime.now)
22     updated_at: Mapped[Optional[datetime]] = mapped_column(default=datetime.now, onupdate=datetime.now)
23
24     # Связь с таблицей адресов

```

Terminal output:

```

lab2 > app git:(main) ✘ uv run alembic revision --autogenerate -m "refactor_order_to_support_multiple_products"
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Config is configured to use SQLite.
INFO [alembic.autogenerate.compare] Detected added table 'order_items'.
INFO [alembic.autogenerate.compare] Detected removed foreign key 'product_id'(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders_qty' on table orders
INFO [alembic.autogenerate.compare] Generating /Users/zadniera/DEV/first_sem/appDev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_
to_support_multiple_.py ...
done

```

Миграция применена к базе данных. Структура БД соответствует обновленным моделям.

2. Создание недостающих компонент

Созданы недостающие компоненты для тестирования, после обновления модели данных с добавлением таблиц для продуктов и заказов

1. Схемы:

- app/schemas/product_schema.py — схемы для продуктов
- app/schemas/order_schema.py — схемы для заказов и элементов заказов

The screenshot shows two instances of the VS Code code editor. Both instances have the title bar "product_schema.py - App_Dev_sem_1" and the status bar "Пн, 24 нояб. 15:07".

Top Window (product_schema.py):

```
1 from datetime import datetime
2 from pydantic import BaseModel, Field
3
4
5 class ProductCreate(BaseModel):
6     """Схема для создания нового продукта."""
7
8     name: str = Field(..., min_length=1, max_length=200, description="Название продукта")
9     description: str | None = Field(None, max_length=1000, description="Описание продукта")
10    price: float = Field(..., gt=0, description="Цена продукта")
11    stock_quantity: int = Field(..., ge=0, description="Количество товара на складе")
12
13
14 class ProductUpdate(BaseModel):
15     """Схема для обновления продукта. Все поля опциональные."""
16
17    name: str | None = Field(None, min_length=1, max_length=200, description="Название продукта")
18    description: str | None = Field(None, max_length=1000, description="Описание продукта")
19    price: float | None = Field(None, gt=0, description="Цена продукта")
20    stock_quantity: int | None = Field(None, ge=0, description="Количество товара на складе")
21
22
23 class ProductResponse(BaseModel):
24     """Схема для ответа API с данными продукта."""
25
26    id: int = Field(..., gt=0, description="Уникальный идентификатор продукта")
27    name: str = Field(..., description="Название продукта")
28    description: str | None = Field(None, description="Описание продукта")
29    price: float = Field(..., description="Цена продукта")
30    stock_quantity: int = Field(..., description="Количество товара на складе")
31    created_at: datetime = Field(..., description="Дата и время создания")
32    updated_at: datetime | None = Field(None, description="Дата и время последнего обновления")
33
34
35 class ProductListResponse(BaseModel):
36     """Схема для ответа API со списком продуктов и общим количеством."""
37
38    products: list[ProductResponse] = Field(..., description="Список продуктов")
39    total: int = Field(..., ge=0, description="Общее количество продуктов в базе данных")
40
41
42
43
44
```

Bottom Window (order_schema.py):

```
1 from datetime import datetime
2 from pydantic import BaseModel, Field
3
4
5 class OrderItemCreate(BaseModel):
6     """Схема для создания элемента заказа."""
7
8    product_id: int = Field(..., gt=0, description="ID продукта")
9    quantity: int = Field(..., gt=0, description="Количество товара")
10
11
12 class OrderItemResponse(BaseModel):
13     """Схема для ответа API с данными элемента заказа."""
14
15    id: int = Field(..., gt=0, description="Уникальный идентификатор элемента заказа")
16    product_id: int = Field(..., gt=0, description="ID продукта")
17    quantity: int = Field(..., description="Количество товара")
18    price_at_order: float = Field(..., description="Цена на момент заказа")
19    created_at: datetime = Field(..., description="Дата и время создания")
20
21
22 class Config:
23     from_attributes = True
24
25
26
27
28 class OrderCreate(BaseModel):
29     """Схема для создания нового заказа."""
30
31    user_id: int = Field(..., gt=0, description="ID пользователя")
32    delivery_address_id: int = Field(..., gt=0, description="ID адреса доставки")
33    items: list[OrderItemCreate] = Field(..., min_length=1, description="Список товаров в заказе")
34    status: str | None = Field(None, description="Статус заказа (по умолчанию 'pending')")
35
36
37 class OrderUpdate(BaseModel):
38     """Схема для обновления заказа. Все поля опциональные."""
39
40    status: str | None = Field(None, description="Статус заказа")
41
42
43 class OrderResponse(BaseModel):
44     """Схема для ответа API с данными заказа."""
45
46    id: int = Field(..., gt=0, description="Уникальный идентификатор заказа")
47    user_id: int = Field(..., gt=0, description="ID пользователя")
48    delivery_address_id: int = Field(..., gt=0, description="ID адреса доставки")
49
50
51
52
53
54
55
56
57
58
59
59
```

2. Репозитории:

- app/repositories/product_repository.py — CRUD для продуктов
- app/repositories/order_repository.py — CRUD для заказов (с поддержкой нескольких продуктов)

3. Сервисы:

- `app/services/product_service.py` — бизнес-логика для продуктов
 - `app/services/order_service.py` — бизнес-логика для заказов (проверка `stock_quantity`, расчет `total_price`)

The image shows two side-by-side screenshots of the VS Code IDE interface. Both windows have the title bar 'Cursor' and show the same file structure for a project named 'APP_DEV_SEM_1'. The left window displays the file 'order_service.py' and the right window displays 'product_service.py'. Both files contain Python code for service classes that interact with repositories and databases. The code includes annotations for class and method descriptions, parameter types, and return values. The bottom status bar of each window shows the current file path ('order_service.py' or 'product_service.py'), the current line ('Ln 1, Col 1' or 'Ln 22, Col 12'), the number of spaces ('Spaces: 4' or 'Spaces: 4'), the character encoding ('UTF-8' or 'UTF-8'), the file type ('Python' or 'Python'), the Python version ('3.13.7' or '3.13.7'), and the environment ('(venv: venv)' or '(venv: venv)').

```

order_service.py U
lab2> app > services > order_service.py > ...
1  from sqlalchemy.ext.asyncio import AsyncSession
2  from sqlalchemy import select
3
4  from app.models import Order, Product, User, Address
5  from app.repositories.order_repository import OrderRepository
6  from app.repositories.product_repository import ProductRepository
7  from app.schemas.order_schema import OrderCreate, OrderUpdate
8
9  class OrderService:
10     """Сервис для бизнес-логики работы с заказами."""
11
12     def __init__(self, order_repository: OrderRepository, product_repository: ProductRepository):
13         ...
14
15         Инициализация сервиса.
16
17     Args:
18         order_repository: Репозиторий для работы с заказами (Dependency Injection)
19         product_repository: Репозиторий для работы с продуктами (Dependency Injection)
20
21     self.order_repository = order_repository
22     self.product_repository = product_repository
23
24     async def get_by_id(self, session: AsyncSession, order_id: int) -> Order | None:
25         ...
26
27         Получить заказ по ID.
28
29         Args:
30             session: Асинхронная сессия базы данных
31             order_id: ID заказа (int)
32
33         Returns:
34             Order объект или None, если не найден
35
36         return await self.order_repository.get_by_id(session, order_id)
37
38     async def get_by_filter(self, session: AsyncSession, count: int, page: int, **kwargs) -> list[Order]:
39
40         ...
41
42         Получить список заказов с пагинацией и фильтрацией.
43
44         Args:
45             session: Асинхронная сессия базы данных
46             count: Количество записей на странице
47             page: Номер страницы (начинается с 1)
48
49     Problems Output Debug Console Terminal Ports
50 INFO [alembic.autogenerate.compare] Detected removed foreign key (product_id)(id) on table orders
51 INFO [alembic.autogenerate.compare] Detected removed column 'orders.quantity'
52 INFO [alembic.autogenerate.compare] Detected removed column 'orders.product_id'
53 Generating /Users/madeira/DEV/first_sem/App_Dev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_
54 to_support_multiple_.py ... done
55 (lab2) > app git:(main) ✘ []

```



```

product_service.py U
lab2> app > services > product_service.py > ProductService > get_by_id
1  from sqlalchemy.ext.asyncio import AsyncSession
2
3  from app.models import Product
4  from app.repositories.product_repository import ProductRepository
5  from app.schemas.product_schema import ProductCreate, ProductUpdate
6
7  class ProductService:
8      """Сервис для бизнес-логики работы с продуктами."""
9
10     def __init__(self, product_repository: ProductRepository):
11         ...
12
13         Инициализация сервиса.
14
15     Args:
16         product_repository: Репозиторий для работы с продуктами (Dependency Injection)
17
18     self.product_repository = product_repository
19
20     async def get_by_id(self, session: AsyncSession, product_id: int) -> Product | None:
21         ...
22
23         Получить продукт по ID.
24
25         Args:
26             session: Асинхронная сессия базы данных
27             product_id: ID продукта (int)
28
29         Returns:
30             Product объект или None, если не найден
31
32         return await self.product_repository.get_by_id(session, product_id)
33
34     async def get_by_filter(self, session: AsyncSession, count: int, page: int, **kwargs) -> list[Product]:
35
36         ...
37
38         Получить список продуктов с пагинацией и фильтрацией.
39
40         Args:
41             session: Асинхронная сессия базы данных
42             count: Количество записей на странице
43             page: Номер страницы (начинается с 1)
44
45     Problems Output Debug Console Terminal Ports
46 INFO [alembic.autogenerate.compare] Detected removed foreign key (product_id)(id) on table orders
47 INFO [alembic.autogenerate.compare] Detected removed column 'orders.quantity'
48 INFO [alembic.autogenerate.compare] Detected removed column 'orders.product_id'
49 Generating /Users/madeira/DEV/first_sem/App_Dev_sem_1/lab2/app/migrations/versions/eeb0a18a7fdf_refactor_order_
50 to_support_multiple_.py ... done
51 (lab2) > app git:(main) ✘ []

```

4. Контроллеры:

- `app/controllers/product_controller.py` — API эндпоинты для продуктов

- `app/controllers/order_controller.py` — API эндпоинты для заказов

```

order_controller.py U
lab2 > app > controllers > order_controller.py ? ...
1  from litestar import Controller, get, post, put, delete
2  from litestar.params import Parameter
3  from sqlalchemy.ext.asyncio import AsyncSession
4
5  from app.exceptions import NotFoundException
6  from app.schemas.order_schema import (
7      OrderCreate,
8      OrderUpdate,
9      OrderResponse,
10     OrderListResponse,
11 )
12  from app.services.order_service import OrderService
13
14
15  class OrderController(Controller):
16      """Контроллер для управления заказами."""
17
18      path = "/orders"
19
20      @get("/{order_id:int}")
21      async def get_order_by_id(
22          self,
23          order_service: OrderService,
24          db_session: AsyncSession,
25          order_id: int = Parameter(gt=0, description="ID заказа"),
26      ) -> OrderResponse:
27          """
28              Получить заказ по ID.
29
30          Args:
31              order_service: Сервис для работы с заказами
32              db_session: Сессия базы данных
33              order_id: ID заказа (int)
34
35          Returns:
36              OrderResponse: Данные заказа с элементами
37
38          Raises:
39              NotFoundException: Если заказ не найден
40
41          order = await order_service.get_by_id(db_session, order_id)
42          if not order:
43              raise NotFoundException(
44                  detail=f"Order with ID {order_id} not found"
45              )
46
47          return OrderResponse.model_validate(order)
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
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
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
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
527
527
528
529
529
530
531
532
533
534
535
536
537
537
538
539
539
540
541
542
543
544
545
545
546
547
547
548
549
549
550
551
552
553
554
555
556
556
557
558
558
559
559
560
561
562
563
564
564
565
566
566
567
567
568
568
569
569
570
571
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
```

- main.py – добавлены контроллеры и зависимости

```
main.py -- App_Dev_sem_1
lab2 > main.py > ...
1 import os
2 from litestar import Litestar
3 from litestar.di import Provide
4 from litestar.openapi import OpenAPIConfig
5
6 from app.controllers.user_controller import UserController
7 from app.controllers.product_controller import ProductController
8 from app.controllers.order_controller import OrderController
9 from app.dependencies import (
10     provide_db_session,
11     provide_user_repository,
12     provide_product_repository,
13     provide_product_service,
14     provide_order_repository,
15     provide_order_service,
16 )
17
18
19 app = Litestar(
20     route_handlers=[
21         UserController,
22         ProductController,
23         OrderController,
24     ],
25     dependencies={
26         "db_session": Provide(provide_db_session),
27         "user_repository": Provide(provide_user_repository),
28         "user_service": Provide(provide_user_service),
29         "product_repository": Provide(provide_product_repository),
30         "product_service": Provide(provide_product_service),
31         "order_repository": Provide(provide_order_repository),
32         "order_service": Provide(provide_order_service),
33     },
34     openapi_config=OpenAPIConfig(
35         title="e-Commerce API",
36         version="1.0.0",
37         description="API для управления пользователями, продуктами и заказами с использованием Dependency Injection и SQLAlchemy",
38     ),
39 )
40
41
42 if __name__ == "__main__":
43     import uvicorn
44     import logging
45
46     uvicorn.run(app)
47
48
49 Problems Output Debug Console Terminal Ports
INFO [alembic.autogenerate.compare] Detected removed foreign key (product_id)(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders.order_id'
INFO [alembic.autogenerate.compare] Generating /Users/2mdeiria/DEV/first_sem/app/app_dev_sem_1/lab/app/migrations/versions/eeb0a18a7fdf_refactor_order_to_support_main_.py ...
done
(tab2) > app git:(main) ✘
```

INFO [alembic.autogenerate.compare] Detected removed foreign key (product_id)(id) on table orders
INFO [alembic.autogenerate.compare] Detected removed column 'orders.order_id'
INFO [alembic.autogenerate.compare] Generating /Users/2mdeiria/DEV/first_sem/app/app_dev_sem_1/lab/app/migrations/versions/eeb0a18a7fdf_refactor_order_to_support_main_.py ... done

3. Настройка тестового окружения

1. Обновлен `pyproject.toml`:

- Добавлены зависимости для тестирования:
 - pytest>=8.0.0
 - pytest-asyncio>=0.23.0
 - pytest-cov>=4.1.0
 - aiosqlite>=0.19.0
 - polyfactory>=2.0.0
 - Добавлена конфигурация pytest:
 - testpaths = ["tests"]
 - asyncio_mode = "auto"
 - addopts = "--verbose --color=yes"

The screenshot shows the VS Code interface with the following details:

- Project Structure:** The left sidebar shows a tree view of the project files. It includes:
 - APP_DEV_SEM_1** (marked with a green dot)
 - lab1**
 - lab2** (marked with a green dot)
 - app**
 - _pycache__**
 - controllers**
 - migrations**
 - repositories**
 - schemas**
 - services**
 - alembic.ini**
 - database.py**
 - dependencies.py**
 - exceptions.py**
 - models.py**
 - push_data.py**
 - push_products_orders.py**
 - queries.py**
 - db_init**
 - tests**
 - .dockerignore**
 - .env**
 - .python-version**
 - docker-compose.yml**
 - Dockerfile**
 - main.py**
 - pyproject.toml** (marked with a green dot)
 - rightconfig.json**
 - README.md**
 - uv.lock**
 - reports**
 - techdoc**
 - .gitignore**
 - README.md**
 - pyproject.toml Content:**

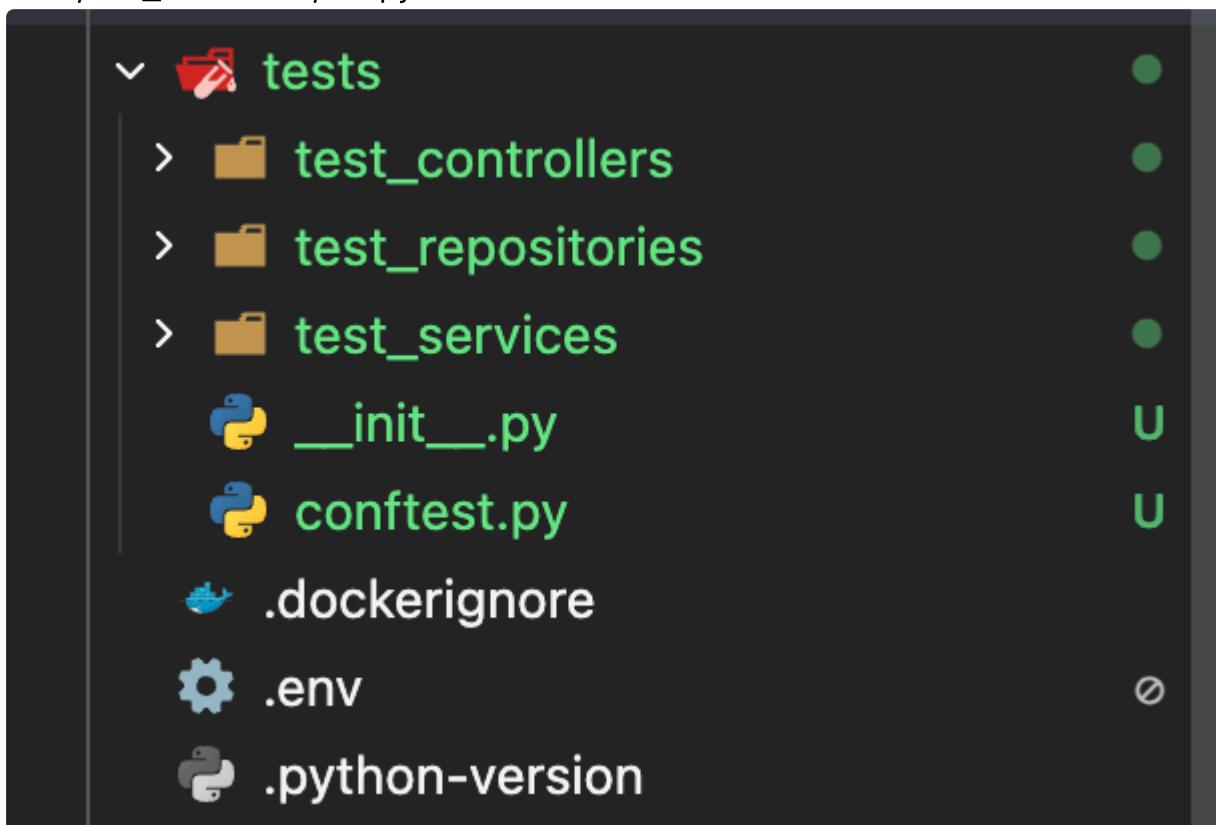
```

[project]
name = "Lab2"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.13"
dependencies = [
    "alembic>=1.17.0",
    "asyncpg>0.30.0",
    "faststar>2.18.0",
    "psycopg2-binary>2.9.11",
    "pytzicemall>2.12.4",
    "python-dotenv>1.1.1",
    "sqlalchemy>2.0.44",
    "uvicorn>0.38.0",
    "pytest>8.0.0",
    "pytest-asyncio>0.23.0",
    "pytest-cov>4.1.0",
    "aiosqlite>0.19.0",
    "polyfactory>2.0.0",
]
[tool.pytest.ini_options]
testpaths = ["tests"]
asyncio_mode = "auto"
addopts = "--verbose --color=yes"

```
 - Terminal:** The bottom right shows the terminal output for the command `docker-compose ps`. It lists services and their status.

2. Создана структура папок для тестов

- tests/**init**.py
- tests/conftest.py — общие фикстуры
- tests/test_repositories/**init**.py
- tests/test_services/**init**.py
- tests/test_controllers/**init**.py



3. Создан conftest.py с фикстурами

- engine (scope="session") — тестовая БД SQLite in-memory
- tables (scope="session") — создание/удаление таблиц
- session — сессия БД для каждого теста
- user_repository, product_repository, order_repository — репозитории
- client — TestClient для тестирования API

```
confest.py
1 import pytest
2 from litestar.testing import TestClient
3 from sqlalchemy.ext.asyncio import AsyncSession, create_async_engine, async_sessionmaker
4
5 from app.models import Base
6 from app.repositories.user_repository import UserRepository
7 from app.repositories.product_repository import ProductRepository
8 from app.repositories.order_repository import OrderRepository
9 from main import app
10
11 # Тестовая база данных (SQLite in-memory)
12 TEST_DATABASE_URL = "sqlite+aiosqlite:///memory:"
13
14
15 @pytest.fixture(scope="session")
16 def engine():
17     """Создает движок для тестовой БД SQLite."""
18     return create_async_engine(TEST_DATABASE_URL, echo=False)
19
20
21 @pytest.fixture(scope="session")
22 def tables(engine):
23     """Создает и удаляет таблицы для тестовой БД."""
24     async with engine.begin() as conn:
25         await conn.run_sync(Base.metadata.drop_all)
26         await conn.run_sync(Base.metadata.create_all)
27     yield
28     async with engine.begin() as conn:
29         await conn.run_sync(Base.metadata.drop_all)
30
31
32 @pytest.fixture
33 def session(engine, tables):
34     """Создает сессию БД для каждого теста."""
35     async_session_factory = async_sessionmaker[AsyncSession](
36         engine, class_=AsyncSession, expire_on_commit=False
37     )
38     async with async_session_factory() as session:
39         yield session
40         await session.rollback()
41
42
43 @pytest.fixture
44 def user_repository():
45     """Фикстура для репозитория пользователей."""
46     return UserRepository()
```

Выполнен uv sync - установлены добавленные в проект зависимости

```
0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
● (lab2) → lab2 git:(main) ✘ uv sync
Resolved 48 packages in 1.68s
Prepared 8 packages in 782ms
Installed 8 packages in 6ms
  + aiosqlite==0.21.0
  + coverage==7.12.0
  + configparser==2.3.0
  + packaging==25.0
  + pluggy==1.6.0
  + pytest==9.0.1
  + pytest-asyncio==1.3.0
  + pytest-cov==7.0.0
○ (lab2) → lab2 git:(main) ✘
```

4. Написание тестов репозиториев

Написаны тесты для трех репозиториев (по 11 штук для каждого). В тестах проверяется пагинация и фильтрация, проверяется каскадное удаление OrderItem при удалении Order.

1. test_user_repository.py

- Создание, получение по ID, получение по email
 - Обновление, удаление
 - Получение списка с пагинацией и фильтрацией
 - Обработка ошибок (несуществующие записи)

2. test_product_repository.py

- Создание, получение по ID
 - Обновление, удаление
 - Получение списка с пагинацией
 - Фильтрация по name, min_price, max_price

- Обработка ошибок

```

1 import pytest
2 from sqlalchemy.ext.asyncio import AsyncSession
3
4 from app.models import Product
5 from app.repositories.product_repository import ProductRepository
6 from app.schemas.product_schema import ProductCreate, ProductUpdate
7
8
9 class TestProductRepository:
10     """Тесты для репозитория продуктов."""
11
12     @pytest.mark.asyncio
13     async def test_create_product(
14         self, session: AsyncSession, product_repository: ProductRepository
15     ):
16         """Тест создания продукта в репозитории."""
17         product_data = ProductCreate(
18             name="Test Product",
19             description="Test description",
20             price=99.99,
21             stock_quantity=10,
22         )
23
24         product = await product_repository.create(session, product_data)
25         await session.commit()
26
27         assert product.id is not None
28         assert product.name == "Test Product"
29         assert product.description == "Test description"
30         assert product.price == 99.99
31         assert product.stock_quantity == 10
32
33     @pytest.mark.asyncio
34     async def test_get_product_by_id(
35         self, session: AsyncSession, product_repository: ProductRepository
36     ):
37         """Тест получения продукта по ID."""
38         product_data = ProductCreate(
39             name="Get By ID Product",
40             description="Product for get_by_id test",
41         )
42
43         product = await product_repository.get_by_id(session, product_data.id)
44
45         assert product is not None
46         assert product.name == "Get By ID Product"
47         assert product.description == "Product for get_by_id test"
48
    
```

Problems Output Debug Console Terminal Ports

create mode 10644 lab2/app/schemas/order_schema.py
create mode 10644 lab2/app/schemas/product_schema.py
create mode 10644 lab2/app/services/_pycache__order_service.cpython-313.pyc
create mode 10644 lab2/app/services/_pycache__product_service.cpython-313.pyc
create mode 10644 lab2/app/services/order_service.py
create mode 10644 lab2/app/services/product_service.py
create mode 10644 lab2/tests/_init_.py
create mode 10644 lab2/tests/test_controllers/_init_.py
create mode 10644 lab2/tests/test_repositories/_init_.py
create mode 10644 lab2/tests/test_services/_init_.py

(lab2) -> lab2 git:(main) >

KK to generate command

Cursor Tab Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 ('venv': venv) @ Go Live

3. test_order_repository.py

- Создание заказа с несколькими продуктами
- Получение заказа по ID с загруженными items
- Обновление статуса заказа
- Удаление заказа с каскадным удалением OrderItem
- Получение списка с фильтрацией по user_id и status

- Тест создания заказа с несколькими продуктами

```

1 import pytest
2 from sqlalchemy.ext.asyncio import AsyncSession
3
4 from app.models import Order, OrderItem, User, Address, Product
5 from app.repositories.order_repository import OrderRepository
6 from app.repositories.user_repository import UserRepository
7 from app.repositories.product_repository import ProductRepository
8 from app.schemas.order_schema import OrderCreate, OrderUpdate, OrderItemCreate
9 from app.schemas.user_schema import UserCreate
10 from app.schemas.product_schema import ProductCreate
11
12
13 class TestOrderRepository:
14     """Тести для репозитория заказов."""
15
16     @pytest.fixture
17     async def test_user(
18         self,
19         session: AsyncSession,
20         user_repository: UserRepository
21     ) -> User:
22         """Создает тестового пользователя."""
23         user_data = UserCreate(
24             email="order_test@example.com",
25             username="order_test_user",
26             description="User for order tests",
27         )
28
29         user = await user_repository.create(session, user_data)
30         await session.commit()
31         return user
32
33     @pytest.fixture
34     async def test_address(
35         self,
36         session: AsyncSession,
37         test_user: User
38     ) -> Address:
39         """Создает тестовый адрес."""
40         address = Address(
41             user_id=test_user.id,
42             street="123 Test St",
43             city="Test City",
44             state="Test State",
45             zip_code="12345",
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
254
255
255
256
257
257
258
259
259
260
261
262
263
264
264
265
266
266
267
267
268
268
269
269
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388

```

- Получение списка и подсчет

```

test_user_service.py U X
lab2 > tests > test_services > test_user_service.py ...
1 import pytest
2 from unittest.mock import AsyncMock, Mock
3 from sqlalchemy.ext.asyncio import AsyncSession
4
5 from app.models import User
6 from app.services.user_service import UserService
7 from app.repositories.user_repository import UserRepository
8 from app.schemas.user_schema import UserCreate, UserUpdate
9
10
11 class TestUserService:
12     """Тесты для сервиса пользователей с моками."""
13
14     @pytest.fixture
15     def mock_user_repository(self):
16         """Создает мок репозитория пользователей."""
17         return AsyncMock(spec=UserRepository)
18
19     @pytest.fixture
20     def mock_session(self):
21         """Создает мок сессии БД."""
22         session = AsyncMock(spec=AsyncSession)
23         session.commit = AsyncMock()
24         return session
25
26     @pytest.fixture
27     def user_service(self, mock_user_repository):
28         """Создает экземпляр сервиса с моком репозитория."""
29         return UserService(mock_user_repository)
30
31     @pytest.mark.asyncio
32     async def test_get_by_id_success(
33         self, user_service: UserService, mock_session, mock_user_repository
34     ):
35         """Тест успешного получения пользователя по ID."""
36         mock_user = Mock(spec=User)
37         mock_user.id = 1
38         mock_user.username = "test_user"
39         mock_user.email = "test@example.com"

```

2. test_product_service.py

- Получение продукта по ID
- Создание продукта (успешное, невалидная цена, отрицательное количество)
- Обновление продукта (успешное, несуществующий, невалидные данные)
- Удаление продукта
- Получение списка и подсчет

```

test_product_service.py U X
lab2 > tests > test_services > test_product_service.py ...
1 import pytest
2 from unittest.mock import AsyncMock, Mock
3 from sqlalchemy.ext.asyncio import AsyncSession
4
5 from app.models import Product
6 from app.services.product_service import ProductService
7 from app.repositories.product_repository import ProductRepository
8 from app.schemas.product_schema import ProductCreate, ProductUpdate
9
10
11 class TestProductService:
12     """Тесты для сервиса продуктов с моками."""
13
14     @pytest.fixture
15     def mock_product_repository(self):
16         """Создает мок репозитория продуктов."""
17         return AsyncMock(spec=ProductRepository)
18
19     @pytest.fixture
20     def mock_session(self):
21         """Создает мок сессии БД."""
22         session = AsyncMock(spec=AsyncSession)
23         session.commit = AsyncMock()
24         return session
25
26     @pytest.fixture
27     def product_service(self, mock_product_repository):
28         """Создает экземпляр сервиса с моком репозитория."""
29         return ProductService(mock_product_repository)
30
31     @pytest.mark.asyncio
32     async def test_get_by_id_success(
33         self, product_service: ProductService, mock_session, mock_product_repository
34     ):
35         """Тест успешного получения продукта по ID."""
36         mock_product = Mock(spec=Product)
37         mock_product.id = 1
38         mock_product.name = "Test Product"
39         mock_product.price = 99.99

```

3. test_order_service.py

- Создание заказа:
 - Успешное создание с несколькими продуктами
 - Недостаточное количество товара на складе
 - Несуществующий продукт
 - Несуществующий пользователь
 - Несуществующий адрес
 - Адрес принадлежит другому пользователю
 - Расчет общей стоимости
- Получение заказа по ID
- Обновление заказа (успешное и несуществующий)
- Удаление заказа

```
test_order_service.py
lab2 tests > test_services > test_order_service.py > ...
1 import pytest
2 from unittest.mock import AsyncMock, Mock
3 from sqlalchemy.ext.asyncio import AsyncSession
4
5 from app.models import Order, Product, User, Address, OrderItem
6 from app.services.order_service import OrderService
7 from app.repositories.order_repository import OrderRepository
8 from app.repositories.product_repository import ProductRepository
9 from app.schemas.order_schema import OrderCreate, OrderUpdate, OrderItemCreate
10
11 class TestOrderService:
12     """Тесты для сервиса заказов с моками."""
13
14     @pytest.fixture
15     def mock_order_repository(self):
16         """Создает мок репозитория заказов."""
17         return AsyncMock(spec=OrderRepository)
18
19     @pytest.fixture
20     def mock_product_repository(self):
21         """Создает мок репозитория продуктов."""
22         return AsyncMock(spec=ProductRepository)
23
24     @pytest.fixture
25     def mock_session(self):
26         """Создает мок сессии БД."""
27         session = AsyncMock(spec=AsyncSession)
28         session.commit = AsyncMock()
29         session.add = Mock()
30         session.refresh = AsyncMock()
31         return session
32
33     @pytest.fixture
34     def order_service(self, mock_order_repository, mock_product_repository):
35         """Создает экземпляр сервиса с моками репозиториев."""
36         return OrderService(mock_order_repository, mock_product_repository)
37
38     @pytest.mark.asyncio
39
lab2 git:(main) ✘
```

6. Тесты для API эндпоинтов

Тесты используют `TestClient` из `litestar`. В качестве тестовой БД используются `sqlite in-memory`. Проверяются `http` статус коды. Проверяется структура `json` ответов.

1. test_user_controller.py

- `GET /users/{user_id}` — получение пользователя по ID
- `GET /users` — получение списка с пагинацией
- `POST /users` — создание пользователя (включая проверку дублирующихся email)
- `PUT /users/{user_id}` — обновление пользователя

- DELETE /users/{user_id} — удаление пользователя
 - Обработка ошибок (404, 400)

The screenshot shows a macOS desktop environment with a terminal window open. The terminal title is "test_user_controller.py – App_Dev_sem_1". The window contains the following code:

```
test_user_controller.py

1 import pytest
2 from litestar.status_codes import HTTP_200_OK, HTTP_201_CREATED, HTTP_204_NO_CONTENT, HTTP_404_NOT_FOUND, HTTP_400_BAD_REQUEST
3 from litestar.testing import TestClient
4
5 from app.models import User
6 from app.schemas.user_schema import UserCreate
7 from app.repositories.user_repository import UserRepository
8
9
10 class TestUserController:
11     """Тесты для API эндпоинтов пользователей."""
12
13     @pytest.mark.asyncio
14     async def test_get_user_by_id(self, client: TestClient, session, user_repository: UserRepository):
15         """Тест GET /users/{user_id} – получение пользователя по ID."""
16         # Создаем пользователя в БД
17         user_data = UserCreate(
18             email="test@example.com",
19             username="test_user",
20             description="Test user",
21         )
22         created_user = await user_repository.create(session, user_data)
23         await session.commit()
24
25         # Делаем запрос к API
26         response = client.get(f"/users/{created_user.id}")
27
28         assert response.status_code == HTTP_200_OK
29         data = response.json()
30         assert data["id"] == created_user.id
31         assert data["email"] == "test@example.com"
32         assert data["username"] == "test_user"
33
34
35         @pytest.mark.asyncio
36     async def test_get_user_by_id_not_found(self, client: TestClient):
37         """Тест GET /users/{user_id} – несуществующий пользователь."""
38         response = client.get("users/99999")
39
39
```

The terminal also displays the following status bar information:

File Edit Selection View Go Run Terminal Window Help
Πт, 24 нояб. 15:53
Agents Editor
Cursor Tab Ln Col 1 Spaces: 4 UTF-8 LF Python 3.13.7 (venv: venv) Go Live

2. test_product_controller.py

- GET /products/{product_id} — получение продукта по ID
 - GET /products — получение списка с пагинацией и фильтрацией
 - POST /products — создание продукта (включая валидацию цены)
 - PUT /products/{product_id} — обновление продукта
 - DELETE /products/{product_id} — удаление продукта
 - Фильтрация по name, min_price, max_price

- Обработка ошибок

```

1 import pytest
2 from litestar.status_codes import HTTP_200_OK, HTTP_201_CREATED, HTTP_204_NO_CONTENT, HTTP_404_NOT_FOUND, HTTP_400_BAD_REQUEST
3 from litestar.testing import TestClient
4
5 from app.models import Product
6 from app.schemas.product_schema import ProductCreate
7 from app.repositories.product_repository import ProductRepository
8
9
10 class TestProductController:
11     """Тесты для API эндпоинтов продуктов."""
12
13     @pytest.mark.asyncio
14     async def test_get_product_by_id(self, client: TestClient, session, product_repository: ProductRepository):
15         """
16             Тест GET /products/{product_id} – получение продукта по ID.
17         """
18         # Создаем продукт в БД
19         product_data = ProductCreate(
20             name="Test Product",
21             description="Test description",
22             price=99.99,
23             stock_quantity=10,
24         )
25         created_product = await product_repository.create(session, product_data)
26         await session.commit()
27
28         # Делаем запрос к API
29         response = client.get(f"/products/{created_product.id}")
30
31         assert response.status_code == HTTP_200_OK
32         data = response.json()
33         assert data["id"] == created_product.id
34         assert data["name"] == "Test Product"
35         assert data["price"] == 99.99
36         assert data["stock_quantity"] == 10
37
38     @pytest.mark.asyncio
39     async def test_get_product_by_id_not_found(self, client: TestClient):
40
41         response = client.get("/products/1")
42
43         assert response.status_code == HTTP_404_NOT_FOUND
44         data = response.json()
45         assert data["detail"] == "Not Found"
46
47

```

3. test_order_controller.py

- GET /orders/{order_id} — получение заказа по ID
- GET /orders — получение списка с фильтрацией
- POST /orders — создание заказа с несколькими продуктами
- POST /orders — обработка ошибки недостаточного количества товара
- PUT /orders/{order_id} — обновление заказа
- DELETE /orders/{order_id} — удаление заказа
- Фильтрация по user_id и status

- Обработка ошибок

```

import pytest
from litestar.status_codes import HTTP_200_OK, HTTP_201_CREATED, HTTP_204_NO_CONTENT, HTTP_404_NOT_FOUND, HTTP_400_BAD_REQUEST
from app.models import User, Address, Product, Order
from app.schemas.user_schema import UserCreate
from app.repositories.user_repository import UserRepository
from app.repositories.product_repository import ProductRepository
from app.repositories.order_repository import OrderRepository

class TestOrderController:
    """Тесты для API эндпоинтов заказов."""

    @pytest.fixture
    async def test_user(self, session, user_repository: UserRepository):
        """Создает тестового пользователя."""
        user_data = UserCreate(
            email="order_test@example.com",
            username="order_test_user",
        )
        user = await user_repository.create(session, user_data)
        await session.commit()
        return user

    @pytest.fixture
    async def test_address(self, session, test_user: User) -> Address:
        """Создает тестовый адрес."""
        from app.models import Address
        address = Address(
            user_id=test_user.id,
            street="123 Test St",
            city="Test City",
            state="Test State",
        )
        session.add(address)
        await session.commit()
        return address

```

7. Запуск тестов

Запускаем тесты. Команды для запусков:

```

cd lab2

# Все тесты
uv run pytest -v

# Только тесты репозиториев
uv run pytest tests/test_repositories/ -v

# Только тесты сервисов
uv run pytest tests/test_services/ -v

# Только тесты API (контроллеров)
uv run pytest tests/test_controllers/ -v

```

Первый запуск тестов показал наличие проблем

```
greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user_not_found - ValueError
: the greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user - ValueError: the greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user_not_found - ValueError
: the greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_all_users - ValueError: the greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_all_users_with_pagination - ValueError: the greenlet library is required to use this function. No module named 'greenlet'
ERROR tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_users_with_filter - ValueError
: the greenlet library is required to use this function. No module named 'greenlet'
===== 17 failed, 28 passed, 4 warnings, 50 errors in 1.75s =====
o (lab2) → lab2 git:(main) ✘
```

⌘K to generate command

0 △ 0

1. Проблемы с тестами контроллеров - в order_service.py локальный импорт select из sqlalchemy конфликтовал с использованием select ранее в коде.
2. В conftest.py фикстура client требовала controller_session для всех тестов, что приводило к багам в тестах на 404 например. Теперь, если controller_session не используется, то создается новая сессия для запроса.
3. Решена проблема с моками при teste создания заказа
4. Отрефакторины все взаимодействия с БД через ORM

```
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_email PASSED [ 92%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_username PASSED [ 93%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_success PASSED [ 94%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_not_found PASSED [ 95%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_duplicate_email PASSED [ 96%]
tests/test_services/test_user_service.py::TestUserService::test_delete_user_success PASSED [ 97%]
tests/test_services/test_user_service.py::TestUserService::test_get_by_filter PASSED [ 98%]
tests/test_services/test_user_service.py::TestUserService::test_count PASSED [100%]

===== 95 passed in 0.66s =====
o (lab2) → lab2 git:(main) ✘
```

⌘K to generate command

Все тесты успешно выполняются.

Вопросы

1. Почему в тестах мы используем отдельную тестовую базу данных (SQLite in-memory)?

Отдельная тестовая база данных используется для изоляции тестов от production-окружения. SQLite in-memory обеспечивает:

- **Быстроту выполнения:** данные хранятся в памяти, нет дисковых операций
- **Автоматическую очистку:** база удаляется после завершения тестов
- **Изоляцию:** тесты не влияют на реальные данные
- **Простота настройки:** не требуется отдельный сервер БД

2. Какие проблемы могут возникнуть при использовании production базы данных для тестирования?

- **Потеря данных:** тесты могут удалить или изменить реальные данные
- **Нарушение целостности:** тестовые данные смешиваются с production
- **Безопасность:** риск утечки конфиденциальной информации
- **Производительность:** нагрузка на production-систему
- **Непредсказуемость:** результаты зависят от текущего состояния БД
- **Параллельное выполнение:** конфликты при одновременном запуске тестов

3. Как работает TestClient в Litestar? Какие преимущества он дает по сравнению с обычными HTTP-запросами?

TestClient в Litestar создает тестовое приложение с переопределенными зависимостями (например, тестовая БД). Преимущества:

- **Без HTTP-слоя:** тесты выполняются напрямую, без сетевых запросов
- **Скорость:** отсутствие накладных расходов на сеть
- **Изоляция зависимостей:** можно подменить любые зависимости (БД, внешние сервисы)
- **Удобство отладки:** прямой доступ к исключениям и стеку вызовов
- **Контроль окружения:** полный контроль над конфигурацией приложения

4. При тестировании сервиса заказов, какие edge cases (границные случаи) нужно учитывать? Напишите к ним тесты.

Основные edge cases:

- Недостаточное количество товара на складе
- Несуществующий продукт
- Несуществующий пользователь
- Несуществующий адрес доставки
- Адрес принадлежит другому пользователю
- Пустой список товаров в заказе
- Нулевое или отрицательное количество товара
- Обновление несуществующего заказа
- Удаление несуществующего заказа

Примеры

```
@pytest.mark.asyncio
async def test_create_order_insufficient_stock(
```

```
    self, order_service, mock_session, mock_order_repository,
mock_product_repository
):
    """Тест создания заказа с недостаточным количеством товара."""
    mock_product = Mock(spec=Product)
    mock_product.stock_quantity = 5
    order_data = OrderCreate(
        user_id=1,
        delivery_address_id=1,
        items=[OrderItemCreate(product_id=1, quantity=10)],
    )

    with pytest.raises(ValueError, match="Insufficient stock"):
        await order_service.create(mock_session, order_data)

@pytest.mark.asyncio
async def test_create_order_invalid_product(
    self, order_service, mock_session, mock_order_repository,
mock_product_repository
):
    """Тест создания заказа с несуществующим продуктом."""
    mock_product_repository.get_by_id.return_value = None
    order_data = OrderCreate(
        user_id=1,
        delivery_address_id=1,
        items=[OrderItemCreate(product_id=999, quantity=1)],
    )

    with pytest.raises(ValueError, match="Product with ID 999 not found"):
        await order_service.create(mock_session, order_data)

@pytest.mark.asyncio
async def test_create_order_address_belongs_to_different_user(
    self, order_service, mock_session, mock_order_repository,
mock_product_repository
):
    """Тест создания заказа, когда адрес принадлежит другому
пользователю."""
    mock_user = Mock(spec=User)
    mock_user.id = 1
    mock_address = Mock(spec=Address)
    mock_address.user_id = 2 # Адрес принадлежит другому пользователю

    order_data = OrderCreate(
        user_id=1,
        delivery_address_id=1,
        items=[OrderItemCreate(product_id=1, quantity=1)],
    )
```

```
    with pytest.raises(ValueError, match="Delivery address does not belong to the user"):
        await order_service.create(mock_session, order_data)
```

5. Как бы вы протестирували метод, который должен отправлять email при смене статуса заказа на "shipped"?

Использовать моки для email-сервиса:

```
@pytest.mark.asyncio
async def test_update_order_sends_email_when_shipped(
    self, order_service, mock_session, mock_order_repository,
    mock_email_service
):
    """Тест отправки email при смене статуса на 'shipped'."""
    existing_order = Mock(spec=Order)
    existing_order.id = 1
    existing_order.status = "pending"
    existing_order.user_id = 1
    updated_order = Mock(spec=Order)
    updated_order.id = 1
    updated_order.status = "shipped"
    updated_order.user_id = 1
    mock_order_repository.get_by_id.return_value = existing_order
    mock_order_repository.update.return_value = updated_order
    update_data = OrderUpdate(status="shipped")
    result = await order_service.update(mock_session, 1, update_data)

    # Проверяем, что email был отправлен
    mock_email_service.send_order_shipped_email.assert_called_once_with(
        user_id=1, order_id=1
    )

    assert result.status == "shipped"

@pytest.mark.asyncio
async def test_update_order_no_email_when_not_shipped(
    self, order_service, mock_session, mock_order_repository,
    mock_email_service
):
    """Тест, что email не отправляется при других статусах."""
    existing_order = Mock(spec=Order)
    existing_order.id = 1
    existing_order.status = "pending"
```

```

updated_order = Mock(spec=Order)
updated_order.id = 1
updated_order.status = "completed"

mock_order_repository.get_by_id.return_value = existing_order
mock_order_repository.update.return_value = updated_order
update_data = OrderUpdate(status="completed")

await order_service.update(mock_session, 1, update_data)
# Проверяем, что email не был отправлен

mock_email_service.send_order_shipped_email.assert_not_called()

```

6. Напишите тест для проверки пагинации товаров. Какие параметры должны проверяться?

Параметры для проверки:

- Корректность количества записей на странице (`count`)
- Корректность номера страницы (`page`)
- Общее количество записей (`total`)
- Корректность данных на разных страницах
- Границные случаи (первая/последняя страница, пустые страницы)

Пример

```

@pytest.mark.asyncio
async def test_get_products_pagination(
    self, client: TestClient, controller_session, product_repository:
ProductRepository
):
    """Тест GET /products – проверка пагинации."""
    # Создаем 5 продуктов
    for i in range(5):
        product_data = ProductCreate(
            name=f"Paginated Product {i}",
            price=10.0 * (i + 1),
            stock_quantity=10,
        )
        await product_repository.create(controller_session, product_data)
    await controller_session.commit()

    # Первая страница (2 записи)
    response = client.get("/products?count=2&page=1")
    assert response.status_code == HTTP_200_OK

```

```

data = response.json()

assert len(data["products"]) == 2
assert data["total"] >= 5
assert "products" in data
assert "total" in data

# Вторая страница (2 записи)
response = client.get("/products?count=2&page=2")
assert response.status_code == HTTP_200_OK

data = response.json()

assert len(data["products"]) == 2

# Третья страница (оставшиеся записи)
response = client.get("/products?count=2&page=3")
assert response.status_code == HTTP_200_OK

data = response.json()

assert len(data["products"]) >= 1

# Проверка граничных случаев
response = client.get("/products?count=10&page=1")

assert response.status_code == HTTP_200_OK

data = response.json()

assert len(data["products"]) <= 10
assert data["total"] >= 5

```

7. Как обеспечить изоляцию тестов друг от друга? Почему это важно?

Изоляция обеспечивается через:

- **Отдельные транзакции**: каждый тест выполняется в своей транзакции с откатом
- **Очистка данных**: удаление данных перед/после каждого теста
- **Отдельные фикстуры**: каждый тест получает чистые экземпляры зависимостей
- **In-memory БД**: каждый тестовый прогон использует новую БД

Пример из `conftest.py` для тестов репозиториев:

```

@ pytest.fixture
async def session(engine, tables):

```

```
"""Создает сессию БД для каждого теста с изоляцией через
транзакцию."""
async with _test_session_factory() as session:
    # Очищаем данные перед каждым тестом
    async with session.begin():
        await session.execute(text("DELETE FROM order_items"))
        await session.execute(text("DELETE FROM orders"))
    # ... очистка других таблиц
    # Начинаем транзакцию для изоляции
    trans = await session.begin()
    try:
        yield session
    finally:
        # Всегда откатываем транзакцию для изоляции тестов
        await trans.rollback()
```

Для тестов контроллеров используется очистка данных перед и после теста, что обеспечивает базовую изоляцию, но для полной изоляции рекомендуется использовать savepoint или откат транзакции.

Почему это важно:

- **Предсказуемость:** тесты дают одинаковые результаты независимо от порядка выполнения
- **Надежность:** один тест не влияет на другой
- **Параллелизация:** тесты можно запускать параллельно
- **Отладка:** легче найти причину падения теста
- **Чистота данных:** каждый тест начинается с чистого состояния

Отвечая на этот вопрос заметил, что у меня не реализована полная изоляция: тесты репозиториев имеют корректную полную изоляцию, тогда как тесты контроллеров частично изолированы. Исправлю эту проблему.

```

90
91     # Создаем savepoint для изоляции
92     # Savepoint позволяет коммитить данные внутри теста (для TestClient),
93     # но откатывать их после теста для полной изоляции
94     # begin_nested() автоматически создаст транзакцию, если ее нет
95     savepoint = await session.begin_nested()
96     _current_test_session = session
97     try:
98         yield session
99     finally:
100        _current_test_session = None
101        # Всегда откатываем savepoint для изоляции тестов
102        # Это откатит все изменения, сделанные в тесте
103        # Если savepoint уже закрыт, очищаем данные вручную
104        try:
105            await savepoint.rollback()
106        except Exception:
107            # Savepoint уже закрыт - очищаем данные вручную для гарантии изоляции
108            try:
109                async with session.begin():
110                    from sqlalchemy import text
111                    await session.execute(text("DELETE FROM order_items"))
112                    await session.execute(text("DELETE FROM orders"))
113                    await session.execute(text("DELETE FROM addresses"))
114                    await session.execute(text("DELETE FROM products"))
115                    await session.execute(text("DELETE FROM users"))
116            except Exception:
117                pass
118
119

```

Была обновлена фикстура controller_session:

- Используется savepoint вместо простой очистки данных
- Savepoint позволяет коммитить данные внутри теста (для TestClient), но откатывает их после теста
- При невозможности откатить savepoint выполняется очистка данных вручную

```

71
72     @pytest.fixture
73     async def controller_session(engine, tables):
74         """Создает сессию БД для тестов контроллеров с изоляцией через savepoint."""
75         global _test_session_factory, _current_test_session
76         if _test_session_factory is None:
77             _test_session_factory = async_sessionmaker[AsyncSession](
78                 engine, class_=AsyncSession, expire_on_commit=False
79             )
80

```

Обновлена функция provide_test_session:

- Для тестов с controller_session используется та же сессия (без дополнительного коммита)
- Для тестов без controller_session создается простая сессия без явной транзакции

```

71
72     # Используем ту же сессию, что и controller_session через глобальную переменную
73     # Если controller_session не используется, создаем новую сессию для этого запроса
74     async def provide_test_session() -> AsyncSession:
75         global _current_test_session, _test_session_factory
76         if _current_test_session is not None:
77             # Используем существующую сессию из controller_session
78             # Не коммитим здесь, так как savepoint в controller_session обеспечит изоляцию
79             yield _current_test_session
80         else:
81             # Создаем новую сессию для тестов, которые не используют controller_session
82             # Для таких тестов изоляция не критична, так как они обычно не создают данные
83             if _test_session_factory is None:
84                 from sqlalchemy.ext.asyncio import async_sessionmaker
85                 _test_session_factory = async_sessionmaker[AsyncSession](
86                     engine, class_=AsyncSession, expire_on_commit=False
87                 )
88             async with _test_session_factory() as session:
89                 try:
90                     yield session
91                 except Exception:
92                     await session.rollback()
93                     raise
94

```

Все тесты успешно пройдены

```

tests/test_services/test_product_service.py::testProductService::test_get_by_ritter PASSED [ 87%]
tests/test_services/test_product_service.py::TestProductService::test_count PASSED [ 88%]
tests/test_services/test_user_service.py::TestUserService::test_get_by_id_success PASSED [ 89%]
tests/test_services/test_user_service.py::TestUserService::test_get_by_id_not_found PASSED [ 90%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_success PASSED [ 91%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_email PASSED [ 92%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_username PASSED [ 93%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_success PASSED [ 94%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_not_found PASSED [ 95%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_duplicate_email PASSED [ 96%]
tests/test_services/test_user_service.py::TestUserService::test_delete_user_success PASSED [ 97%]
tests/test_services/test_user_service.py::TestUserService::test_get_by_filter PASSED [ 98%]
tests/test_services/test_user_service.py::TestUserService::test_count PASSED [100%]

===== 95 passed in 0.66s =====

```

○ (lab2) → lab2 git:(main) ✘

⌘K to generate command

Cursor Tab ~o- Andrey Zhunev (1 hour ago) Ln 174, C

Выходы

В ходе работы изучены и применены подходы к тестированию бэкенд-приложения: созданы тесты для репозиториев (интеграционные с БД), сервисов (с моками) и контроллеров (API через TestClient). Настроено тестовое окружение на SQLite in-memory, реализована изоляция тестов через откат транзакций и savepoint, что обеспечивает независимость и предсказуемость. Все 95 тестов проходят успешно, что подтверждает корректность реализации бизнес-логики, валидации данных и обработки граничных случаев. Применены принципы SOLID и KISS, использована Dependency Injection для упрощения тестирования и поддержки кода.