Министерство Образования Российской Федерации
УрФУ
Образовательных информационных технологий

Лабораторная работа №4
Тема: «Тестирование бэкенд приложения»

Выполнил:
Нарсеев А.В.

Екатеринбург, 2025

Лабораторная работа №4: Тестирование бэкенд приложения

Цель работы: познакомится со способами тестирования приложения
Лабораторная работа выполняется на основе ЛР3, в репозитории после выполнения добавляем тэг lab_4.

1. Pytest fixture

/AppDev/LR4/tests/conftest.py:

```
TEST_DATABASE_URL = os.getenv("TEST_DATABASE_URL", "sqlite:///./test.db")


@pytest.fixture(scope="session")
def engine():
    engine = create_engine(
        TEST_DATABASE_URL,
        connect_args={"check_same_thread": False} if "sqlite" in TEST_DATABASE_URL e
    )
    Base.metadata.drop_all(engine)
    Base.metadata.create_all(engine)
    yield engine
    Base.metadata.drop_all(engine)
    engine.dispose()


@pytest.fixture
```

2. Тестирование создания пользователя

Пробуем создать пользователя в БД
/AppDev/LR4/tests/test_user_repository.py:

```
class TestUserRepository:
    def test_create_user(self, db_session, user_repository: UserRepository):
        async def _run():
            user = await user_repository.create(
                db_session,
                UserCreate(username="repo_user", email="repo_user@example.com"),
            )
            assert user.id is not None
            assert user.username == "repo_user"
            assert user.email == "repo_user@example.com"

        asyncio.run(_run())

    def test_get_user_by_email(self, db_session, user_repository: UserRepository):
```

Также пробуем найти пользователя в базе данных и обновить данные пользователя

```
    def test_get_user_by_email(self, db_session, user_repository: UserRepository):
        async def _run():
            created = await user_repository.create(
                db_session,
                UserCreate(username="email_user", email="email@example.com"),
            )

            found = await user_repository.get_by_email(db_session, "email@example.com"
            assert found is not None
            assert found.id == created.id

        asyncio.run(_run())
```

К этим тестам необходимо добавить удаление и получение списка пользователей. Для репозитория заказов и продукции необходимо добавить тесты создания, обновления, получения списка, получения конкретного заказа и удаления. При этом необходимо учитывать, что в заказе может находиться не один продукт, а несколько.

```
def test_get_by_filter_returns_list(self, db_session, user_repository: UserRepository):
    async def _run():
        await user_repository.create(
            db_session,
            UserCreate(username="user1", email="user1@example.com"),
        )
        await user_repository.create(
            db_session,
            UserCreate(username="user2", email="user2@example.com"),
        )

        users = await user_repository.get_by_filter(db_session, count=10, page=1)
        usernames = sorted([user.username for user in users])
        assert usernames == ["user1", "user2"]

    asyncio.run(_run())

def test_update_user(self, db_session, user_repository: UserRepository):
    async def _run():
        created = await user_repository.create(
            db_session,
            UserCreate(username="to_update", email="update@example.com", description="old"),
        )

        updated = await user_repository.update(
            db_session,
            created.id,
            UserUpdate(description="new description"),
        )
        assert updated is not None
        assert updated.description == "new description"
        assert updated.username == "to_update"

    asyncio.run(_run())
```

```python
def test_delete_user(self, db_session, user_repository: UserRepository):
    async def _run():
        created = await user_repository.create(
            db_session,
            UserCreate(username="to_delete", email="delete@example.com"),
        )

        await user_repository.delete(db_session, created.id)
        deleted = await user_repository.get_by_id(db_session, created.id)
        assert deleted is None

    asyncio.run(_run())
```

3. Тестируем сервисный слой

Mock (мок) в Python — это инструмент для тестирования кода, который заменяет реальные объекты контролируемыми имитациями.
Используя мок мы можем проверить как работает сервисный слой не взаимодействуя с БД.

/AppDev/LR4/tests/test_order_service.py

```python
def test_create_order_success():
    async def _run():
        mock_order_repo = AsyncMock()
        mock_product_repo = AsyncMock()
        mock_user_repo = AsyncMock()

        mock_user_repo.get_by_id.return_value = Mock(id=UUID(int=1))

        product = Mock(id=UUID(int=2), price=100.0, stock_quantity=5)
        mock_product_repo.get_by_id.return_value = product

        mock_order_repo.create.return_value = Mock(total_amount=200.0)

        service = OrderService(mock_order_repo, mock_product_repo, mock_user_repo)

        order_data = OrderCreate(
            user_id=UUID(int=1),
            address_id=None,
            items=[OrderItemCreate(product_id=UUID(int=2), quantity=2)],
        )

        result = await service.create_order(Mock(), order_data)
        assert result.total_amount == 200.0
        assert product.stock_quantity == 3
        mock_order_repo.create.assert_awaited_once()

    asyncio.run(_run())
```

4. Тестируем API

Также независимо от других слоёв приложений мы можем протестировать работу наших эндпоинтов.

```python
from uuid import UUID


def test_get_users_empty(test_client):
    response = test_client.get("/users")
    assert response.status_code == 200
    payload = response.json()
    assert payload["total"] == 0
    assert payload["users"] == []


def test_create_and_retrieve_user(test_client):
    create_response = test_client.post(
        "/users",
        json={"username": "api_user", "email": "api_user@example.com"},
    )
    assert create_response.status_code in (200, 201)
    created_user = create_response.json()
    user_id = UUID(created_user["id"])
    assert created_user["username"] == "api_user"

    get_response = test_client.get(f"/users/{user_id}")
    assert get_response.status_code == 200
    fetched = get_response.json()
    assert fetched["email"] == "api_user@example.com"
```

Запуск тестов

Все тесты
Pytest

```
narseev@Mac LR4 % source venv/bin/activate
(venv) narseev@Mac LR4 % pytest
============================= test session starts ==============================
platform darwin -- Python 3.12.7, pytest-8.4.2, pluggy-1.6.0 -- /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4/venv/bin/python3
cachedir: .pytest_cache
rootdir: /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4
configfile: pyproject.toml
testpaths: tests
plugins: asyncio-1.2.0, anyio-4.11.0, Faker-37.11.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 21 items

tests/test_order_repository.py::test_create_order_with_multiple_items PASSED [  4%]
tests/test_order_repository.py::test_list_and_delete_orders PASSED       [  9%]
tests/test_order_service.py::test_create_order_success PASSED            [ 14%]
tests/test_order_service.py::test_create_order_insufficient_stock PASSED [ 19%]
tests/test_order_service.py::test_create_order_without_items PASSED      [ 23%]
tests/test_product_repository.py::TestProductRepository::test_create_product PASSED [ 28%]
tests/test_product_repository.py::TestProductRepository::test_list_products PASSED [ 33%]
tests/test_product_repository.py::TestProductRepository::test_update_product PASSED [ 38%]
tests/test_product_repository.py::TestProductRepository::test_delete_product PASSED [ 42%]
tests/test_user_repository.py::TestUserRepository::test_create_user PASSED [ 47%]
tests/test_user_repository.py::TestUserRepository::test_get_user_by_email PASSED [ 52%]
tests/test_user_repository.py::TestUserRepository::test_get_by_filter_returns_list PASSED [ 57%]
tests/test_user_repository.py::TestUserRepository::test_update_user PASSED [ 61%]
tests/test_user_repository.py::TestUserRepository::test_delete_user PASSED [ 66%]
tests/test_user_routes.py::test_get_users_empty PASSED                   [ 71%]
tests/test_user_routes.py::test_create_and_retrieve_user PASSED          [ 76%]
tests/test_user_routes.py::test_update_and_delete_user PASSED            [ 80%]
tests/test_user_service.py::TestUserService::test_create_user PASSED     [ 85%]
tests/test_user_service.py::TestUserService::test_get_users PASSED       [ 90%]
tests/test_user_service.py::TestUserService::test_update_user PASSED     [ 95%]
tests/test_user_service.py::TestUserService::test_delete_user PASSED     [100%]

============================== 21 passed in 0.17s ==============================
```

Только unit-тесты
pytest tests/test_user_repository.py tests/test_product_repository.py tests/test_order_repository.py tests/test_user_service.py tests/test_order_service.py

```
(venv) narseev@Mac LR4 % pytest tests/test_user_repository.py tests/test_product_repository.py tests/test_order_repository.py tests/test_user_service.py tes
ts/test_order_service.py
============================= test session starts ==============================
platform darwin -- Python 3.12.7, pytest-8.4.2, pluggy-1.6.0 -- /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4/venv/bin/python3
cachedir: .pytest_cache
rootdir: /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4
configfile: pyproject.toml
plugins: asyncio-1.2.0, anyio-4.11.0, Faker-37.11.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 18 items

tests/test_user_repository.py::TestUserRepository::test_create_user PASSED [  5%]
tests/test_user_repository.py::TestUserRepository::test_get_user_by_email PASSED [ 11%]
tests/test_user_repository.py::TestUserRepository::test_get_by_filter_returns_list PASSED [ 16%]
tests/test_user_repository.py::TestUserRepository::test_update_user PASSED [ 22%]
tests/test_user_repository.py::TestUserRepository::test_delete_user PASSED [ 27%]
tests/test_product_repository.py::TestProductRepository::test_create_product PASSED [ 33%]
tests/test_product_repository.py::TestProductRepository::test_list_products PASSED [ 38%]
tests/test_product_repository.py::TestProductRepository::test_update_product PASSED [ 44%]
tests/test_product_repository.py::TestProductRepository::test_delete_product PASSED [ 50%]
tests/test_order_repository.py::test_create_order_with_multiple_items PASSED [ 55%]
tests/test_order_repository.py::test_list_and_delete_orders PASSED       [ 61%]
tests/test_user_service.py::TestUserService::test_create_user PASSED     [ 66%]
tests/test_user_service.py::TestUserService::test_get_users PASSED       [ 72%]
tests/test_user_service.py::TestUserService::test_update_user PASSED     [ 77%]
tests/test_user_service.py::TestUserService::test_delete_user PASSED     [ 83%]
tests/test_order_service.py::test_create_order_success PASSED            [ 88%]
tests/test_order_service.py::test_create_order_insufficient_stock PASSED [ 94%]
tests/test_order_service.py::test_create_order_without_items PASSED      [100%]

============================== 18 passed in 0.10s ==============================
```

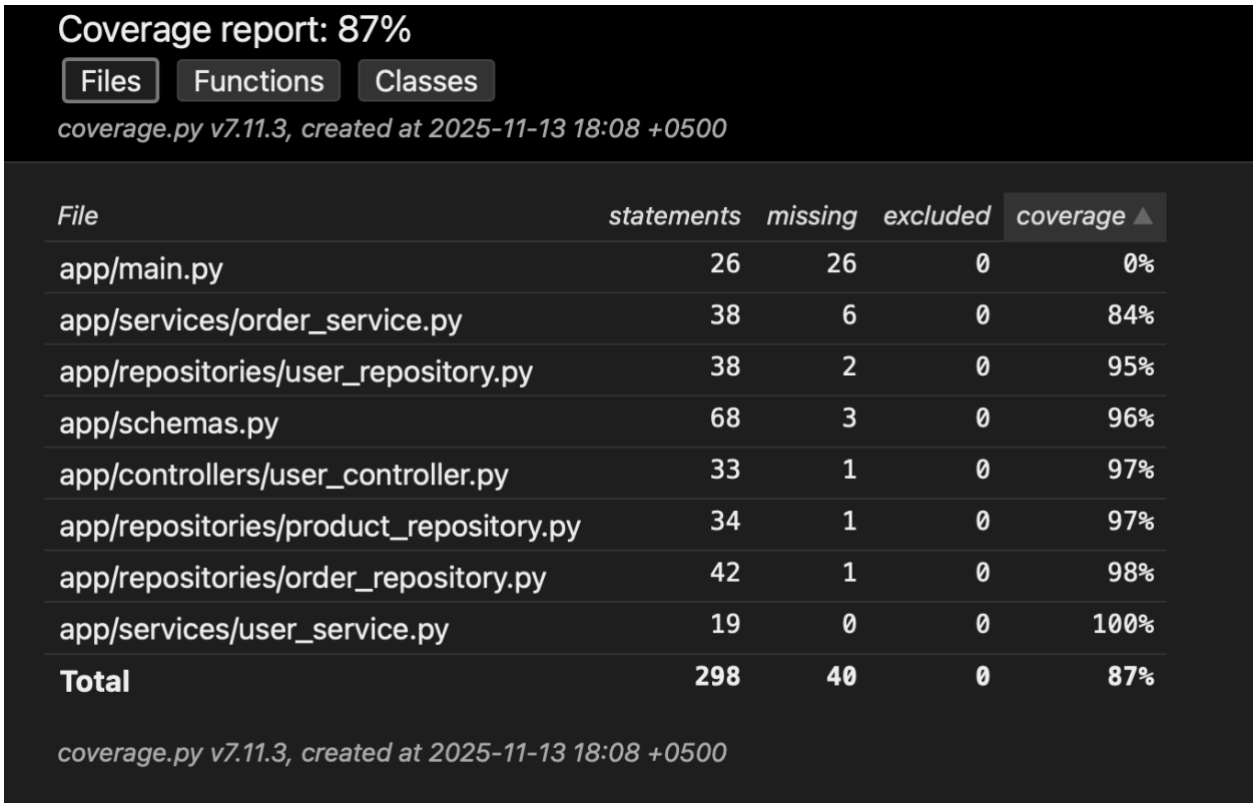
Только API тесты
pytest tests/test_user_routes.py

```
(venv) narseev@Mac LR4 % pytest tests/test_user_routes.py
============================= test session starts ==============================
platform darwin -- Python 3.12.7, pytest-8.4.2, pluggy-1.6.0 -- /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4/venv/bin/python3
cachedir: .pytest_cache
rootdir: /Users/narseev/Documents/UrFU/term_1/AppDev/LR4/AppDev/LR4
configfile: pyproject.toml
plugins: asyncio-1.2.0, anyio-4.11.0, Faker-37.11.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 3 items

tests/test_user_routes.py::test_get_users_empty PASSED                   [ 33%]
tests/test_user_routes.py::test_create_and_retrieve_user PASSED          [ 66%]
tests/test_user_routes.py::test_update_and_delete_user PASSED            [100%]

============================== 3 passed in 0.11s ===============================
```

С покрытием кода
pytest --cov=app --cov-report=html



Параллельный запуск
pytest -n auto

pyproject.toml

[tool.pytest.ini_options]
testpaths = ["tests"]
asyncio_mode = "auto"
addopts = "--verbose --color=yes"

Вопросы

1. Почему в тестах мы используем отдельную тестовую базу данных (SQLite in-memory)?
   Быстро поднимается, не требует внешнего сервиса и каждый прогон начинается с чистого состояния

   Какие проблемы могут возникнуть при использовании production базы данных для тестирования?
   Можно случайно удалить реальные данные. Засорение боевой БД тестовыми объектами

2. Как работает TestClient в Litestar?
   Клиент оборачивает ASGI-приложение в объект, который исполняет запросы в памяти, без поднятия реального HTTP-сервера. Запросы идут напрямую через event loop; мы получаем полноценные ответы (status_code, json()) и можем подменять зависимости

   Какие преимущества он дает по сравнению с обычными HTTP-запросами?
   Быстрый запуск, нет сетевых задержек, не нужен отдельный процесс, можно контролировать состояние приложения

3. При тестировании сервиса заказов, какие edge cases (граничные случаи) нужно учитывать? Напишите к ним тесты
   В проекте проверяем три случая: успешное создание, нехватка товара, пустой список позиций
   Реализовано тут: /AppDev/LR4/tests/test_order_service.py

4. Как бы вы протестировали метод, который должен отправлять email при смене статуса заказа на "shipped"?
   Выносим отправку почты в отдельный сервис и внедряем его через зависимость. В тесте подменяем его мок-объектом и проверяем, что он вызван ровно один раз при переводе заказа в shipped и не вызывается для других статусов

5. Напишите тест для проверки пагинации товаров.
   /AppDev/LR4/tests/test_product_pagination.py

   Какие параметры должны проверяться?
   Тест проверяет:
   count - размер страницы (по 2 товара)
   page - номер страницы: первая, вторая и третья
   Содержимое каждой страницы и что последняя возвращает остаток == 1

6. Как обеспечить изоляцию тестов друг от друга?
   Наш db_session открывает соединение, начинает транзакцию и откатывает её после теста; база создаётся заново для сессии (SQLite файл), а контейнеры не участвуют

   Почему это важно?
   Изоляция обеспечивает независимость тестов. Результат одного не влияет на другой, ошибки воспроизводимы. Без неё тесты становятся нестабильными и трудно диагностировать причины падений

Ссылки

GitHub: https://github.com/gh-u14/AppDev/