Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
УРАЛЬСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ ПЕРВОГО ПРЕЗИДЕНТА РОССИИ Б.Н. ЕЛЬЦИНА
(УрФУ имени первого Президента России Б.Н. Ельцина)
Институт радиоэлектроники и информационных технологий — РТФ
Школа профессионального и академического образования

ТЕСТИРОВАНИЕ БЭКЕНД ПРИЛОЖЕНИЯ

Отчет по лабораторной работе №4

по дисциплине «Разработка приложений»

	Дата	Подпись	
Преподаватель:	_____	_____	Кузьмин Д.
Студенты:	_____	_____	Пантелеев Е.А.

Группа: РИМ-150950

Екатеринбург

2025

ЦЕЛЬ РАБОТЫ

Познакомится со способами тестирования приложения.

ХОД РАБОТЫ

1. Pytest fixture

В начале работы объявили фикстуры в conftest.py

```
GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def user_repository():
    return UserRepository()

GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def product_repository():
    return ProductRepository()

GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def order_repository():
    return OrderRepository()

GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def user_service(user_repository):
    return UserService(user_repository)

GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def order_service(order_repository, product_repository, user_repository):
    return OrderService(order_repository, product_repository, user_repository)

GigaCode: explain | explain step by step | doc | test
@pytest.fixture
def test_client(session, user_repository, user_service):
    def get_session() -> Session:
        return session

    def get_user_repository():
        return user_repository

    def get_user_service():
```

2. Тестирование создания пользователя

Для тестирования пользователя был создан файл test_user_repository.py.

```
tests > test_repositories > test_user_repository.py > TestUserRepository > test_delete_user
import asyncio
from repositories.user_repository import UserRepository
from schemas import UserCreate, UserUpdate


GigaCode: explain | explain step by step | doc | test
class TestUserRepository:
    def test_create_user(self, session, user_repository: UserRepository):
        async def _run():
            user = await user_repository.create(
                session,
                UserCreate(username="test_user_1", email="test1@example.com"),
            )
            assert user.id is not None
            assert user.username == "test_user_1"
            assert user.email == "test1@example.com"

        asyncio.run(_run())

    def test_get_by_filter_returns_list(self, session, user_repository: UserRepository):
        async def _run():
            await user_repository.create(
                session,
                UserCreate(username="test_user_2a", email="test2a@example.com"),
            )
            await user_repository.create(
                session,
                UserCreate(username="test_user_2b", email="test2b@example.com"),
            )

            users = await user_repository.get_by_filter(session, count=10, page=1)
            usernames = sorted([user.username for user in users])
            assert "test_user_2a" in usernames
            assert "test_user_2b" in usernames
```

Здесь также сделаны и тесты на удаление пользователя.

```
    def test_update_user(self, session, user_repository: UserRepository):
        async def _run():
            created = await user_repository.create(
                session,
                UserCreate(username="test_user_3", email="test3@example.com", description="old"),
            )

            updated = await user_repository.update(
                session,
                created.id,
                UserUpdate(description="new description"),
            )
            assert updated is not None
            assert updated.description == "new description"
            assert updated.username == "test_user_3"

        asyncio.run(_run())

    def test_delete_user(self, session, user_repository: UserRepository):
        async def _run():
            created = await user_repository.create(
                session,
                UserCreate(username="test_user_4", email="test4@example.com"),
            )

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

        asyncio.run(_run())
```

3. Тестируем сервисный слой

Для тестирования сервисного слоя создаем test_user_service.py.

4. Тестируем API

Для тестирования API создаем файл test_user_routes.py.

tests > test_controllers > test_user_routes.py > ...

```python
import pytest


GigaCode: explain | explain step by step | doc | test
def test_get_users_empty(test_client):
    response = test_client.get("/users")
    assert response.status_code == 200
    payload = response.json()
    assert payload == []


GigaCode: explain | explain step by step | doc | test
def test_create_and_retrieve_user(test_client):
    create_response = test_client.post(
        "/users",
        json={"username": "api_user_1", "email": "api_user1@example.com"},
    )
    assert create_response.status_code in (200, 201)
    created_user = create_response.json()
    user_id = created_user["id"]
    assert created_user["username"] == "api_user_1"

    get_response = test_client.get(f"/users/{user_id}")
    assert get_response.status_code == 200
    fetched = get_response.json()
    assert fetched["email"] == "api_user1@example.com"

```

5. Запуск тестов

Все тесты

pytest

```
collected 25 items

tests/test_controllers/test_user_routes.py::test_get_users_empty PASSED                                                [  4%]
tests/test_controllers/test_user_routes.py::test_create_and_retrieve_user PASSED                                       [  8%]
tests/test_controllers/test_user_routes.py::test_update_and_delete_user PASSED                                         [ 12%]
tests/test_controllers/test_user_routes.py::test_get_users_with_filters PASSED                                         [ 16%]
tests/test_controllers/test_user_routes.py::test_create_user_validation PASSED                                         [ 20%]
tests/test_repositories/test_order_repository.py::TestOrderRepository::test_create_order PASSED                        [ 24%]
tests/test_repositories/test_order_repository.py::TestOrderRepository::test_list_orders PASSED                         [ 28%]
tests/test_repositories/test_order_repository.py::TestOrderRepository::test_update_order_status PASSED                 [ 32%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_create_product PASSED                  [ 36%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_get_list_products PASSED               [ 40%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_update_product PASSED                  [ 44%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_update_stock PASSED                    [ 48%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_delete_product PASSED                  [ 52%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED                           [ 56%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_filter_returns_list PASSED           [ 60%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user PASSED                           [ 64%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user PASSED                           [ 68%]
tests/test_services/test_order_service.py::TestOrderService::test_create_order PASSED                                  [ 72%]
tests/test_services/test_order_service.py::TestOrderService::test_list_orders PASSED                                   [ 76%]
tests/test_services/test_order_service.py::TestOrderService::test_update_order_status PASSED                           [ 80%]
tests/test_services/test_order_service.py::TestOrderService::test_delete_order PASSED                                  [ 84%]
tests/test_services/test_user_service.py::TestUserService::test_create_user PASSED                                     [ 88%]
tests/test_services/test_user_service.py::TestUserService::test_get_users PASSED                                       [ 92%]
tests/test_services/test_user_service.py::TestUserService::test_update_user PASSED                                     [ 96%]
tests/test_services/test_user_service.py::TestUserService::test_delete_user PASSED                                     [100%]
```

Только unit-тесты

pytest tests/test_repositories

```
============================= test session starts =============================
platform win32 -- Python 3.12.8, pytest-9.0.1, pluggy-1.6.0 -- D:\Уник\Разработка приложений (Кузьмин Д)\lab3\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Уник\Разработка приложений (Кузьмин Д)\lab3
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-37.12.0, asyncio-1.3.0, cov-7.0.0, mock-3.15.1, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 12 items

tests/test_repositories/test_order_repository.py::TestOrderRepository::test_create_order PASSED [  8%]
tests/test_repositories/test_order_repository.py::TestOrderRepository::test_list_orders PASSED [ 16%]
tests/test_repositories/test_order_repository.py::TestOrderRepository::test_update_order_status PASSED [ 25%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_create_product PASSED [ 33%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_get_list_products PASSED [ 41%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_update_product PASSED [ 50%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_update_stock PASSED [ 58%]
tests/test_repositories/test_product_repository.py::TestProductRepository::test_delete_product PASSED [ 66%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED [ 75%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_filter_returns_list PASSED [ 83%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user PASSED [ 91%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user PASSED [100%]
```

tests/test_services/

```
(.venv) D:\Уник\Разработка приложений (Кузьмин Д)\lab3>pytest tests/test_services -v
============================= test session starts =============================
platform win32 -- Python 3.12.8, pytest-9.0.1, pluggy-1.6.0 -- D:\Уник\Разработка приложений (Кузьмин Д)\lab3\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Уник\Разработка приложений (Кузьмин Д)\lab3
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-37.12.0, asyncio-1.3.0, cov-7.0.0, mock-3.15.1, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 8 items

tests/test_services/test_order_service.py::TestOrderService::test_create_order PASSED [ 12%]
tests/test_services/test_order_service.py::TestOrderService::test_list_orders PASSED [ 25%]
tests/test_services/test_order_service.py::TestOrderService::test_update_order_status PASSED [ 37%]
tests/test_services/test_order_service.py::TestOrderService::test_delete_order PASSED [ 50%]
tests/test_services/test_user_service.py::TestUserService::test_create_user PASSED [ 62%]
tests/test_services/test_user_service.py::TestUserService::test_get_users PASSED [ 75%]
tests/test_services/test_user_service.py::TestUserService::test_update_user PASSED [ 87%]
tests/test_services/test_user_service.py::TestUserService::test_delete_user PASSED [100%]
```

Только API тесты

pytest tests/test_controllers/

```
(.venv) D:\Уник\Разработка приложений (Кузьмин Д)\lab3>pytest tests/test_controllers
============================= test session starts =============================
platform win32 -- Python 3.12.8, pytest-9.0.1, pluggy-1.6.0 -- D:\Уник\Разработка приложений (Кузьмин Д)\lab3\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\Уник\Разработка приложений (Кузьмин Д)\lab3
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-37.12.0, asyncio-1.3.0, cov-7.0.0, mock-3.15.1, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 5 items

tests/test_controllers/test_user_routes.py::test_get_users_empty PASSED [ 20%]
tests/test_controllers/test_user_routes.py::test_create_and_retrieve_user PASSED [ 40%]
tests/test_controllers/test_user_routes.py::test_update_and_delete_user PASSED [ 60%]
tests/test_controllers/test_user_routes.py::test_get_users_with_filters PASSED [ 80%]
tests/test_controllers/test_user_routes.py::test_create_user_validation PASSED [100%]
```

С покрытием кода

pytest --cov=app --cov-report=html

File ▲	statements	missing	excluded	coverage
__init__.py	0	0	0	100%
controllers \ __init__.py	0	0	0	100%
controllers \ user_controller.py	37	2	0	95%
generate_tree.py	25	25	0	0%
main.py	34	34	0	0%
models.py	63	0	0	100%
repositories \ __init__.py	4	0	0	100%
repositories \ order_repository.py	54	5	0	91%
repositories \ product_repository.py	46	5	0	89%
repositories \ user_repository.py	35	1	0	97%
schemas.py	101	3	0	97%
scripts \ __init__.py	0	0	0	100%
scripts \ load_data.py	32	32	0	0%
scripts \ update_data.py	61	61	0	0%
services \ __init__.py	3	0	0	100%
services \ order_service.py	27	2	0	93%
services \ user_service.py	18	0	0	100%
tests \ __init__.py	0	0	0	100%
tests \ conftest.py	59	1	0	98%
tests \ test_controllers \ test_user_routes.py	36	0	0	100%
tests \ test_repositories \ __init__.py	0	0	0	100%
tests \ test_repositories \ test_order_repository.py	68	0	0	100%
tests \ test_repositories \ test_product_repository.py	42	0	0	100%
tests \ test_repositories \ test_user_repository.py	35	0	0	100%
tests \ test_services \ __init__.py	0	0	0	100%
tests \ test_services \ test_order_service.py	80	0	0	100%
tests \ test_services \ test_user_service.py	32	0	0	100%
Total	**892**	**171**	**0**	**81%**

6. Ответы на вопросы

1. Почему в тестах мы используем отдельную тестовую базу данных (SQLite in-memory)? Какие проблемы могут возникнуть при использовании production базы данных для тестирования?

   Избегаем повреждения продакшн-данных, обеспечиваем изоляцию

тестов. Production БД может измениться, тесты станут нестабильными, возможна потеря данных.

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

Эмулирует HTTP-запросы без запуска сервера. Быстрее реальных запросов, не требует сети, проще отладка, интеграция с DI-контейнером.

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

   - Недостаток товара на складе (stock_quantity < quantity)
   - Несуществующий продукт в заказе
   - Несуществующий пользователь
   - Пустой список items в заказе
   - Отрицательное количество товара

   Тесты для них уже есть.

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

Здесь лучше будет мокать email-сервис в OrderService. Проверяем вызов метода отправки при смене статуса на "shipped" с правильными данными заказа.

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

Проверяем: limit, offset, общее количество, корректность данных на страницах. Создаем достаточно данных для нескольких страниц.

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

Каждый тест в своей транзакции с откатом. Важно для предсказуемости, избежания побочных эффектов и параллельного запуска.

ЗАКЛЮЧЕНИЕ

В ходе работы были освоены принципы тестирования бэкенд приложения.