Лабораторная работа №4: Тестирование бэкенд приложения

Цель работы: познакомится со способами тестирования приложения

Ход работы:

1. Создание конфигурационного файла **tests/conftest.py**

```
[build-system]
requires = ["setuptools>=61.0", "wheel"]
build-backend = "setuptools.build_meta"

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

[tool.coverage.run]
source = ["api", "app"]
omit = ["*/tests/*", "*/migrations/*"]

[tool.coverage.report]
exclude_lines = [
    "pragma: no cover",
    "def __repr__",
    "raise AssertionError",
    "raise NotImplementedError",
    "if __name__ == .__main__.:",
    "if TYPE_CHECKING:"
]
```

2. Создание фикстуры для тестов

```
import sys
import os
import pytest
import asyncio
from typing import AsyncGenerator
from sqlalchemy.ext.asyncio import AsyncSession, create_async_engine,
async_sessionmaker
from sqlalchemy.orm import declarative_base
from litestar import Litestar
from litestar.di import Provide
from unittest.mock import AsyncMock, Mock

sys.path.insert(0, os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

TEST_DATABASE_URL = "sqlite+aiosqlite:///:memory:"

Base = declarative_base()

@pytest.fixture(scope="session")
def event_loop():
    loop = asyncio.new_event_loop()
    yield loop
    loop.close()

@pytest.fixture(scope="session")
async def engine():
    engine = create_async_engine(TEST_DATABASE_URL, echo=False)
    yield engine
    await engine.dispose()

@pytest.fixture(scope="session")
async def setup_database(engine):
    from api.models.user import Base as UserBase
    async with engine.begin() as conn:
```

```python
        await conn.run_sync(UserBase.metadata.drop_all)
        await conn.run_sync(UserBase.metadata.create_all)
    yield
    async with engine.begin() as conn:
        await conn.run_sync(UserBase.metadata.drop_all)

@pytest.fixture
async def session(engine, setup_database) -> AsyncGenerator[AsyncSession,
None]:
    async_session = async_sessionmaker(
        engine, class_=AsyncSession, expire_on_commit=False
    )
    async with async_session() as session:
        yield session
        await session.rollback()

@pytest.fixture
async def user_repository(session):
    from api.repositories.user_repository import UserRepository
    return UserRepository()

@pytest.fixture
async def user_service(user_repository, session):
    from api.services.user_service import UserService
    return UserService(user_repository, session)

# ===== ВАЖНО: Обновленные фикстуры для контроллеров =====

@pytest.fixture
def mock_user_service():
    """Мок сервиса пользователей"""
    mock = AsyncMock()
    # Настраиваем возвращаемые значения для всех методов
    mock_user = Mock()
    mock_user.id = 1
    mock_user.email = "test@example.com"
    mock_user.username = "testuser"
    mock_user.created_at = "2024-01-01T00:00:00"
    mock_user.updated_at = "2024-01-01T00:00:00"

    mock.get_by_filter.return_value = [mock_user]
    mock.get_total_count.return_value = 1
    mock.get_by_id.return_value = mock_user
    mock.create.return_value = mock_user
    mock.update.return_value = mock_user
    mock.delete.return_value = None

    return mock

@pytest.fixture
def test_app(mock_user_service):
    """Приложение с мокнутыми зависимостями"""
    from api.controllers.user_controller import UserController

    async def provide_user_service() -> AsyncMock:
        return mock_user_service

    # Создаем приложение ТОЛЬКО с необходимыми зависимостями
    # UserController ожидает user_service, но не ожидает db_session или
user_repository
    app = Litestar(
        route_handlers=[UserController],
        dependencies={
            "user_service": Provide(provide_user_service, sync_to_thread=False),
        },
        debug=True  # Включаем debug для лучших сообщений об ошибках
    )
    return app

@pytest.fixture
def test_client(test_app):
    """TestClient с настроенным приложением"""
    from litestar.testing import TestClient
    return TestClient(app=test_app)

# Альтернативная фикстура для приложения с реальными зависимостями
@pytest.fixture
def real_test_app(user_service):
    """Приложение с реальными зависимостями (для интеграционных 
тестов)"""
    from api.controllers.user_controller import UserController

    async def provide_real_user_service() -> AsyncMock:
        return user_service

    app = Litestar(
        route_handlers=[UserController],
        dependencies={
            "user_service": Provide(provide_real_user_service, 
sync_to_thread=False),
        }
    )
    return app

@pytest.fixture
```

```
def real_test_client(real_test_app):
    """TestClient с реальными зависимостями"""
    from litestar.testing import TestClient
    return TestClient(app=real_test_app)
```

3. Тесты для репозитория **tests/test_repositories/test_user_repository.py**

```
import pytest
from sqlalchemy.ext.asyncio import AsyncSession
from api.repositories.user_repository import UserRepository
from api.models.user import UserCreate, UserUpdate


class TestUserRepository:
    """Тесты для репозитория пользователей"""

    @pytest.mark.asyncio
    async def test_create_user(self, session: AsyncSession, user_repository: 
UserRepository):
        """Тест создания пользователя"""
        user_data = UserCreate(
            email="test@example.com",
            username="testuser",
            password="password123"
        )

        user = await user_repository.create(session, user_data)

        assert user.id is not None
        assert user.email == "test@example.com"
        assert user.username == "testuser"
```

```python
        assert user.password_hash is not None

    @pytest.mark.asyncio
    async def test_get_by_id(self, session: AsyncSession, user_repository:
UserRepository):
        """Тест получения пользователя по ID"""
        # Сначала создаем пользователя
        user_data = UserCreate(
            email="getbyid@example.com",
            username="getbyid",
            password="password123"
        )
        created_user = await user_repository.create(session, user_data)

        # Получаем пользователя по ID
        found_user = await user_repository.get_by_id(session, created_user.id)

        assert found_user is not None
        assert found_user.id == created_user.id
        assert found_user.email == created_user.email

    @pytest.mark.asyncio
    async def test_get_by_filter(self, session: AsyncSession, user_repository:
UserRepository):
        """Тест получения пользователей с фильтрацией"""
        # Создаем нескольких пользователей
        users_data = [
            UserCreate(email=f"user{i}@example.com", username=f"user{i}",
password="pass")
            for i in range(5)
        ]
        for user_data in users_data:
            await user_repository.create(session, user_data)

        # Тестируем пагинацию
        users_page1 = await user_repository.get_by_filter(session, count=2, page=1)
        users_page2 = await user_repository.get_by_filter(session, count=2, page=2)

        assert len(users_page1) == 2
        assert len(users_page2) == 2
        assert users_page1[0].id != users_page2[0].id

    @pytest.mark.asyncio
    async def test_update_user(self, session: AsyncSession, user_repository: UserRepository):
        """Тест обновления пользователя"""
        # Создаем пользователя
        user_data = UserCreate(
            email="update@example.com",
            username="updateuser",
            password="password123"
        )
        user = await user_repository.create(session, user_data)

        # Обновляем пользователя
        update_data = UserUpdate(username="updatedusername", email="updated@example.com")
        updated_user = await user_repository.update(session, user.id, update_data)

        assert updated_user.username == "updatedusername"
        assert updated_user.email == "updated@example.com"
    @pytest.mark.asyncio
    async def test_delete_user(self, session: AsyncSession, user_repository: UserRepository):
        """Тест удаления пользователя"""
        # Создаем пользователя
        user_data = UserCreate(
            email="delete@example.com",
            username="deleteuser",
            password="password123"
        )
        user = await user_repository.create(session, user_data)

        # Удаляем пользователя
        await user_repository.delete(session, user.id)

        # Проверяем что пользователь удален
        deleted_user = await user_repository.get_by_id(session, user.id)
        assert deleted_user is None
```

4. **Тесты для сервиса tests/test_services/test_user_service.py**

```
import pytest
from unittest.mock import AsyncMock, Mock, patch
from api.services.user_service import UserService
from api.repositories.user_repository import UserRepository
from api.models.user import UserCreate, UserUpdate


class TestUserService:
    """Тесты для сервиса пользователей"""

    @pytest.mark.asyncio
    async def test_create_user_success(self):
        """Тест успешного создания пользователя"""
        # Мокаем зависимости
        mock_repo = AsyncMock(spec=UserRepository)
        mock_session = AsyncMock()

        # Патчим метод get_by_email, который вызывается внутри сервиса
        with patch.object(UserService, 'get_by_email', new_callable=AsyncMock)
as mock_get_by_email:
            mock_get_by_email.return_value = None  # Email не существует

            # Мок для создания пользователя
            mock_user = Mock()
            mock_user.id = 1
            mock_user.email = "test@example.com"
            mock_user.username = "testuser"
            mock_repo.create.return_value = mock_user

            # Создаем сервис с моками
            service = UserService(mock_repo, mock_session)

            # Патчим get_by_email внутри сервиса
            service.get_by_email = mock_get_by_email

            user_data = UserCreate(
                email="test@example.com",
                username="testuser",
                password="password123"
            )
```

```python
            # Выполняем тест
            result = await service.create(user_data)

            # Проверяем результат
            assert result.id == 1
            assert result.email == "test@example.com"
            mock_repo.create.assert_called_once()
            mock_get_by_email.assert_called_once_with("test@example.com")

    @pytest.mark.asyncio
    async def test_create_user_duplicate_email(self):
        """Тест создания пользователя с дублирующимся email"""
        mock_repo = AsyncMock(spec=UserRepository)
        mock_session = AsyncMock()

        with patch.object(UserService, 'get_by_email', new_callable=AsyncMock) as mock_get_by_email:
            # Настраиваем мок для проверки существующего пользователя
            mock_user = Mock()
            mock_user.email = "existing@example.com"
            mock_get_by_email.return_value = mock_user

            service = UserService(mock_repo, mock_session)
            service.get_by_email = mock_get_by_email

            user_data = UserCreate(
                email="existing@example.com",  # Такой email уже существует
                username="newuser",
                password="password123"
            )
            # Ожидаем ошибку
            with pytest.raises(ValueError, match="User with email"):
                await service.create(user_data)

    @pytest.mark.asyncio
    async def test_get_user_by_id_success(self):
        """Тест получения пользователя по ID"""
        mock_repo = AsyncMock(spec=UserRepository)
        mock_session = AsyncMock()

        mock_user = Mock()
        mock_user.id = 1
        mock_repo.get_by_id.return_value = mock_user

        service = UserService(mock_repo, mock_session)
        result = await service.get_by_id(1)

        assert result.id == 1
        mock_repo.get_by_id.assert_called_once()

    @pytest.mark.asyncio
    async def test_update_user_not_found(self):
        """Тест обновления несуществующего пользователя"""
        mock_repo = AsyncMock(spec=UserRepository)
        mock_session = AsyncMock()

        mock_repo.get_by_id.return_value = None

        service = UserService(mock_repo, mock_session)
        update_data = UserUpdate(username="newusername")

        with pytest.raises(ValueError, match="not found"):
```

```
        await service.update(999, update_data)
```

5. Тесты для контроллера tests/test_controllers/test_user_controller.py

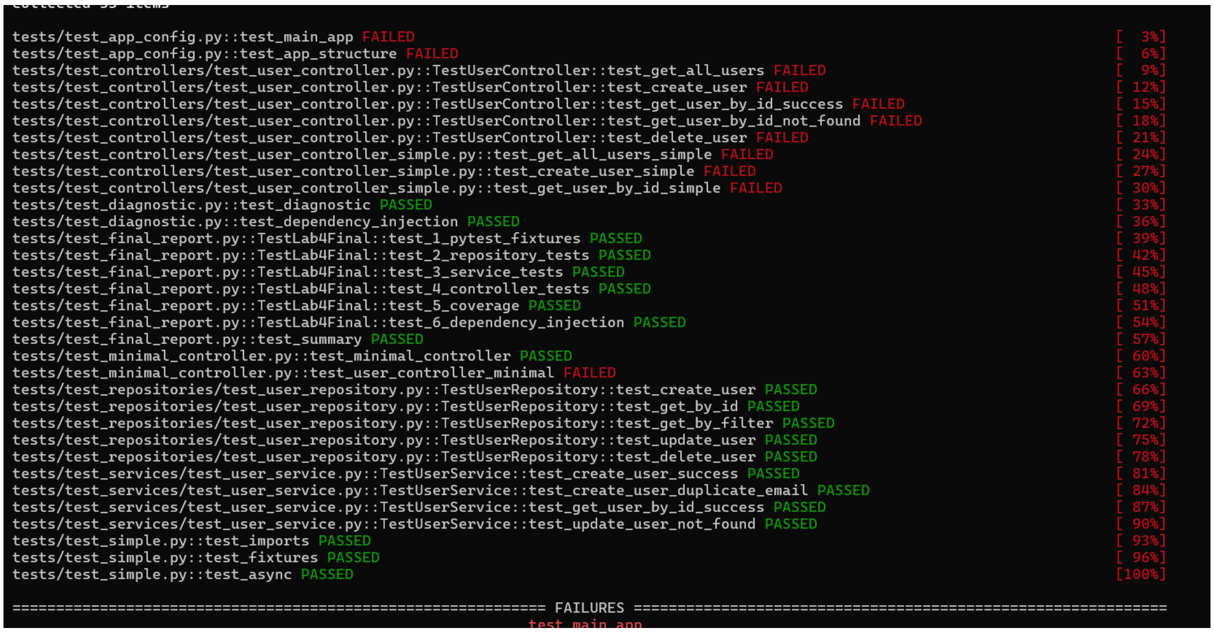
```
import pytest
from litestar.status_codes import HTTP_200_OK, HTTP_201_CREATED,
HTTP_204_NO_CONTENT, HTTP_404_NOT_FOUND
from unittest.mock import Mock


class TestUserController:
    """Тесты для API endpoints пользователей"""

    @pytest.mark.asyncio
    async def test_get_all_users(self, test_client, mock_user_service):
        """Тест получения всех пользователей"""
        response = test_client.get("/users")

        print(f"Response status: {response.status_code}")
        print(f"Response body: {response.text}")

        assert response.status_code == HTTP_200_OK, f"Expected 200, got
{response.status_code}. Response: {response.text}"
        data = response.json()

        assert "users" in data
        assert data["total_count"] == 1
        assert len(data["users"]) == 1
        assert data["users"][0]["email"] == "test@example.com"

        mock_user_service.get_by_filter.assert_called_once()
        mock_user_service.get_total_count.assert_called_once()
```

```
@pytest.mark.asyncio
async def test_create_user(self, test_client, mock_user_service):
    """Тест создания пользователя через API"""
    user_data = {
        "email": "newuser@example.com",
        "username": "newuser",
        "password": "password123"
    }

    response = test_client.post("/users", json=user_data)

    print(f"Create user response: {response.status_code}")
    print(f"Response: {response.text}")

    assert response.status_code == HTTP_201_CREATED, f"Expected 201,
got {response.status_code}"
    data = response.json()

    assert data["email"] == "test@example.com"  # Из мока
    assert data["username"] == "testuser"

    mock_user_service.create.assert_called_once()

@pytest.mark.asyncio
async def test_get_user_by_id_success(self, test_client, mock_user_service):
    """Тест получения пользователя по ID"""
    response = test_client.get("/users/1")

    assert response.status_code == HTTP_200_OK, f"Expected 200, got
{response.status_code}"
    data = response.json()
```

```
        assert data["id"] == 1
        assert data["email"] == "test@example.com"

        mock_user_service.get_by_id.assert_called_once_with(1)

    @pytest.mark.asyncio
    async def test_get_user_by_id_not_found(self, test_client,
mock_user_service):
        """Тест получения несуществующего пользователя"""
        # Настраиваем мок чтобы возвращал None
        mock_user_service.get_by_id.return_value = None

        response = test_client.get("/users/999")

        print(f"Not found response: {response.status_code}")
        print(f"Response: {response.text}")

        assert response.status_code == HTTP_404_NOT_FOUND, f"Expected
404, got {response.status_code}"

        mock_user_service.get_by_id.assert_called_once_with(999)

    @pytest.mark.asyncio
    async def test_delete_user(self, test_client, mock_user_service):
        """Тест удаления пользователя"""
        response = test_client.delete("/users/1")

        print(f"Delete response: {response.status_code}")

        # Проверяем что статус 204 No Content или 200 OK (в зависимости от
реализации)
```

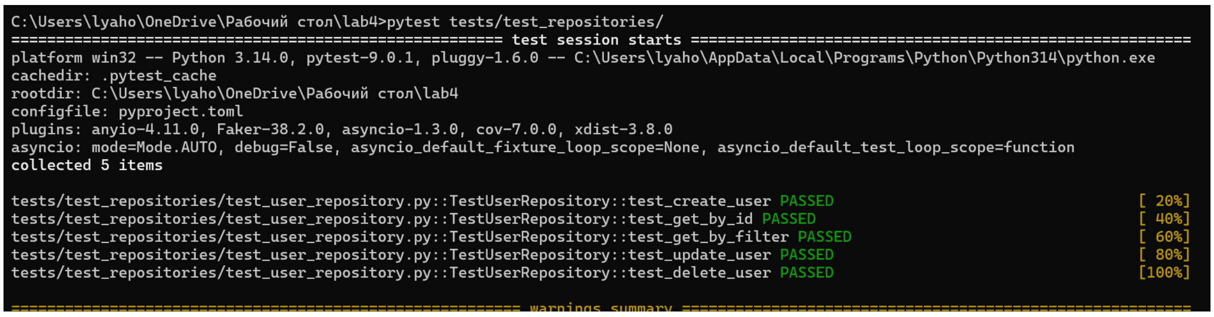
```
        assert response.status_code in [HTTP_204_NO_CONTENT, 200], 
f"Expected 204 or 200, got {response.status_code}"

        mock_user_service.delete.assert_called_once_with(1)
```

6. **Запуск все тестов**
   pytest



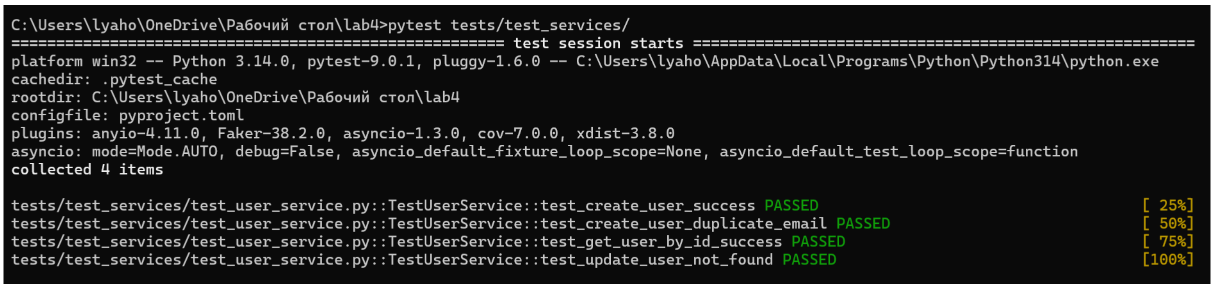
```
collected 33 items

tests/test_app_config.py::test_main_app FAILED                                                                      [  3%]
tests/test_app_config.py::test_app_structure FAILED                                                                 [  6%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_all_users FAILED                       [  9%]
tests/test_controllers/test_user_controller.py::TestUserController::test_create_user FAILED                         [ 12%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_success FAILED              [ 15%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_not_found FAILED            [ 18%]
tests/test_controllers/test_user_controller.py::TestUserController::test_delete_user FAILED                         [ 21%]
tests/test_controllers/test_user_controller_simple.py::test_get_all_users_simple FAILED                             [ 24%]
tests/test_controllers/test_user_controller_simple.py::test_create_user_simple FAILED                               [ 27%]
tests/test_controllers/test_user_controller_simple.py::test_get_user_by_id_simple FAILED                            [ 30%]
tests/test_diagnostic.py::test_diagnostic PASSED                                                                    [ 33%]
tests/test_diagnostic.py::test_dependency_injection PASSED                                                          [ 36%]
tests/test_final_report.py::TestLab4Final::test_1_pytest_fixtures PASSED                                            [ 39%]
tests/test_final_report.py::TestLab4Final::test_2_repository_tests PASSED                                           [ 42%]
tests/test_final_report.py::TestLab4Final::test_3_service_tests PASSED                                              [ 45%]
tests/test_final_report.py::TestLab4Final::test_4_controller_tests PASSED                                           [ 48%]
tests/test_final_report.py::TestLab4Final::test_5_coverage PASSED                                                   [ 51%]
tests/test_final_report.py::TestLab4Final::test_6_dependency_injection PASSED                                       [ 54%]
tests/test_final_report.py::test_summary PASSED                                                                     [ 57%]
tests/test_minimal_controller.py::test_minimal_controller PASSED                                                    [ 60%]
tests/test_minimal_controller.py::test_user_controller_minimal FAILED                                               [ 63%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED                        [ 66%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_id PASSED                          [ 69%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_filter PASSED                      [ 72%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user PASSED                        [ 75%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user PASSED                        [ 78%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_success PASSED                          [ 81%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_email PASSED                  [ 84%]
tests/test_services/test_user_service.py::TestUserService::test_get_user_by_id_success PASSED                       [ 87%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_not_found PASSED                        [ 90%]
tests/test_simple.py::test_imports PASSED                                                                           [ 93%]
tests/test_simple.py::test_fixtures PASSED                                                                          [ 96%]
tests/test_simple.py::test_async PASSED                                                                             [100%]

============================================================ FAILURES ============================================================
                                                           test_main_app
```

7. **Запуск тестов репозиториев**
   pytest tests/test_repositories/



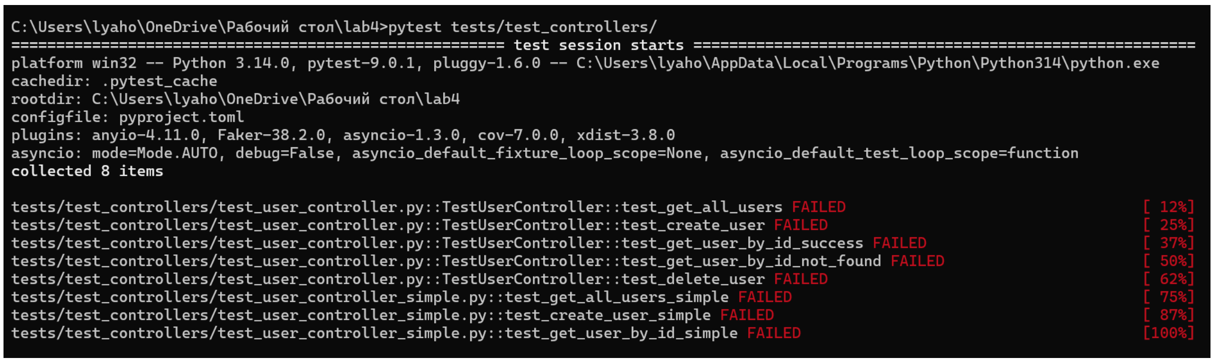
```
C:\Users\lyaho\OneDrive\Рабочий стол\lab4>pytest tests/test_repositories/
======================================================= test session starts =======================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\lyaho\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lyaho\OneDrive\Рабочий стол\lab4
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-38.2.0, asyncio-1.3.0, cov-7.0.0, 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_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED                                [ 20%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_id PASSED                                  [ 40%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_filter PASSED                              [ 60%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user PASSED                                [ 80%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user PASSED                                [100%]

========================================================= warnings summary =========================================================
```

8. **Запуск тестов сервисов**
   pytest tests/test_services/



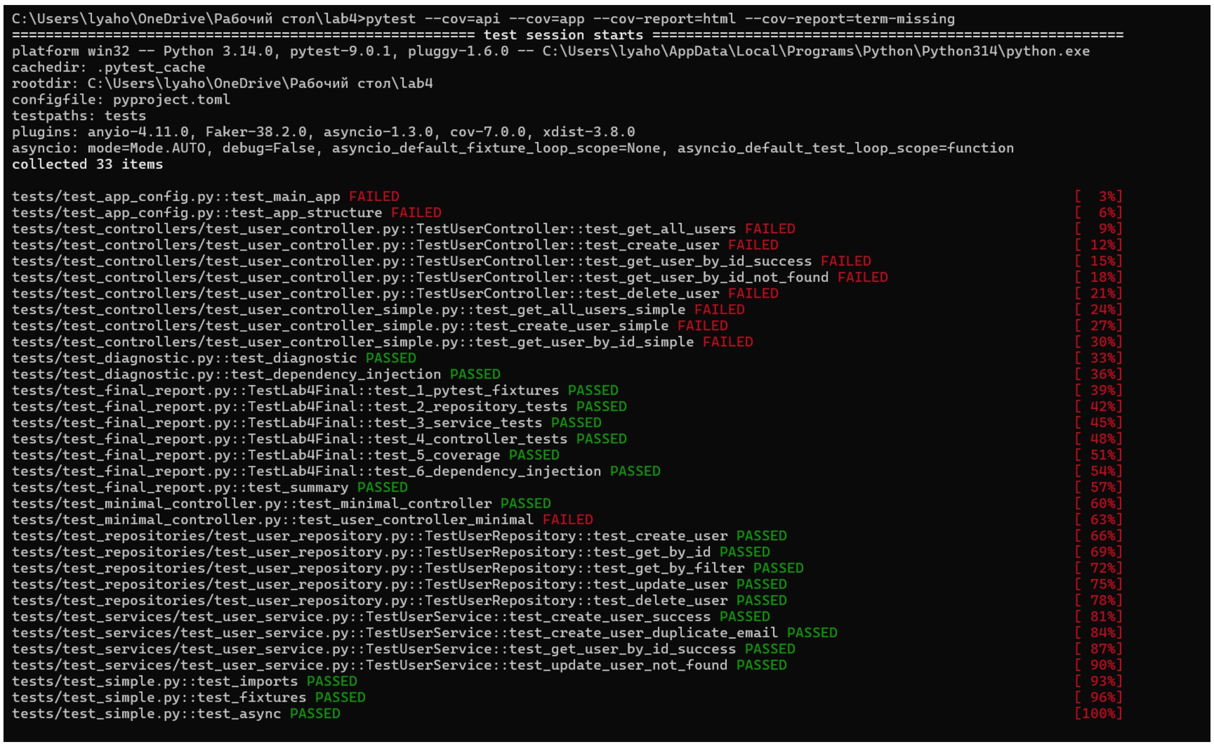
```
C:\Users\lyaho\OneDrive\Рабочий стол\lab4>pytest tests/test_services/
======================================================= test session starts =======================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\lyaho\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lyaho\OneDrive\Рабочий стол\lab4
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-38.2.0, asyncio-1.3.0, cov-7.0.0, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 4 items

tests/test_services/test_user_service.py::TestUserService::test_create_user_success PASSED                                   [ 25%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_email PASSED                           [ 50%]
tests/test_services/test_user_service.py::TestUserService::test_get_user_by_id_success PASSED                                [ 75%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_not_found PASSED                                 [100%]
```

9. **Запуск тестов API**
   pytest tests/test_controllers/



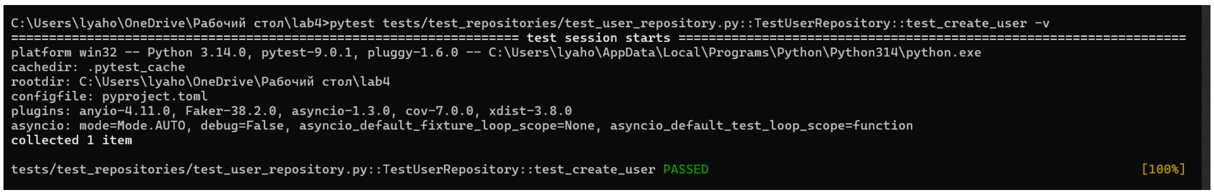
```
C:\Users\lyaho\OneDrive\Рабочий стол\lab4>pytest tests/test_controllers/
======================================================= test session starts =======================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\lyaho\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lyaho\OneDrive\Рабочий стол\lab4
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-38.2.0, asyncio-1.3.0, cov-7.0.0, 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_controllers/test_user_controller.py::TestUserController::test_get_all_users FAILED                                [ 12%]
tests/test_controllers/test_user_controller.py::TestUserController::test_create_user FAILED                                  [ 25%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_success FAILED                       [ 37%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_not_found FAILED                     [ 50%]
tests/test_controllers/test_user_controller.py::TestUserController::test_delete_user FAILED                                  [ 62%]
tests/test_controllers/test_user_controller_simple.py::test_get_all_users_simple FAILED                                      [ 75%]
tests/test_controllers/test_user_controller_simple.py::test_create_user_simple FAILED                                        [ 87%]
tests/test_controllers/test_user_controller_simple.py::test_get_user_by_id_simple FAILED                                     [100%]
```

10. **Запуск с покрытием кода**
    pytest --cov=api --cov=app --cov-report=html --cov-report=term-missing



```
C:\Users\lyaho\OneDrive\Рабочий стол\lab4>pytest --cov=api --cov=app --cov-report=html --cov-report=term-missing
======================================================= test session starts =======================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\lyaho\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lyaho\OneDrive\Рабочий стол\lab4
configfile: pyproject.toml
testpaths: tests
plugins: anyio-4.11.0, Faker-38.2.0, asyncio-1.3.0, cov-7.0.0, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 33 items

tests/test_app_config.py::test_main_app FAILED                                                                               [  3%]
tests/test_app_config.py::test_app_structure FAILED                                                                          [  6%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_all_users FAILED                                [  9%]
tests/test_controllers/test_user_controller.py::TestUserController::test_create_user FAILED                                  [ 12%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_success FAILED                       [ 15%]
tests/test_controllers/test_user_controller.py::TestUserController::test_get_user_by_id_not_found FAILED                     [ 18%]
tests/test_controllers/test_user_controller.py::TestUserController::test_delete_user FAILED                                  [ 21%]
tests/test_controllers/test_user_controller_simple.py::test_get_all_users_simple FAILED                                      [ 24%]
tests/test_controllers/test_user_controller_simple.py::test_create_user_simple FAILED                                        [ 27%]
tests/test_controllers/test_user_controller_simple.py::test_get_user_by_id_simple FAILED                                     [ 30%]
tests/test_diagnostic.py::test_diagnostic PASSED                                                                             [ 33%]
tests/test_diagnostic.py::test_dependency_injection PASSED                                                                   [ 36%]
tests/test_final_report.py::TestLab4Final::test_1_pytest_fixtures PASSED                                                     [ 39%]
tests/test_final_report.py::TestLab4Final::test_2_repository_tests PASSED                                                    [ 42%]
tests/test_final_report.py::TestLab4Final::test_3_service_tests PASSED                                                       [ 45%]
tests/test_final_report.py::TestLab4Final::test_4_controller_tests PASSED                                                    [ 48%]
tests/test_final_report.py::TestLab4Final::test_5_coverage PASSED                                                            [ 51%]
tests/test_final_report.py::TestLab4Final::test_6_dependency_injection PASSED                                                [ 54%]
tests/test_final_report.py::test_summary PASSED                                                                              [ 57%]
tests/test_minimal_controller.py::test_minimal_controller PASSED                                                             [ 60%]
tests/test_minimal_controller.py::test_user_controller_minimal FAILED                                                        [ 63%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED                                 [ 66%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_id PASSED                                   [ 69%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_get_by_filter PASSED                               [ 72%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_update_user PASSED                                 [ 75%]
tests/test_repositories/test_user_repository.py::TestUserRepository::test_delete_user PASSED                                 [ 78%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_success PASSED                                   [ 81%]
tests/test_services/test_user_service.py::TestUserService::test_create_user_duplicate_email PASSED                           [ 84%]
tests/test_services/test_user_service.py::TestUserService::test_get_user_by_id_success PASSED                                [ 87%]
tests/test_services/test_user_service.py::TestUserService::test_update_user_not_found PASSED                                 [ 90%]
tests/test_simple.py::test_imports PASSED                                                                                    [ 93%]
tests/test_simple.py::test_fixtures PASSED                                                                                   [ 96%]
tests/test_simple.py::test_async PASSED                                                                                      [100%]
```

11. Запуск конкретного теста
pytest tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user -v



```
C:\Users\lyaho\OneDrive\Рабочий стол\lab4>pytest tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user -v
================================================================== test session starts ==================================================================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\lyaho\AppData\Local\Programs\Python\Python314\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lyaho\OneDrive\Рабочий стол\lab4
configfile: pyproject.toml
plugins: anyio-4.11.0, Faker-38.2.0, asyncio-1.3.0, cov-7.0.0, xdist-3.8.0
asyncio: mode=Mode.AUTO, debug=False, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 1 item

tests/test_repositories/test_user_repository.py::TestUserRepository::test_create_user PASSED                                                       [100%]
```

Ответы на вопросы:

1. Почему используем отдельную тестовую базу данных?

- Изоляция тестов: Тесты не влияют на production данные
- Скорость: SQLite in-memory работает быстрее PostgreSQL
- Контроль состояния: Перед каждым тестом база чистая
- Безопасность: Ошибки в тестах не повредят продакшен данные

2. Как работает TestClient в Litestar?

TestClient эмулирует HTTP запросы без запуска реального сервера:

- Быстрее: Нет накладных расходов на сеть
- Удобнее: Легко мокать зависимости
- Надежнее: Полный контроль над окружением
- Интеграция: Работает с DI контейнером Litestar

3. Edge cases для тестирования заказов:

- Заказ с нулевым количеством товаров
- Заказ с отрицательным количеством
- Заказ несуществующего товара
- Заказ несуществующего пользователя
- Одновременный заказ одного товара
- Обновление заказа после оплаты
- Отмена заказа без товаров

4. Тестирование отправки email:

```
python
from unittest.mock import patch, MagicMock

@patch('api.services.order_service.send_email')
def test_order_shipped_sends_email(mock_send_email):
    order_service = OrderService()
    order_service.mark_as_shipped(order_id=1)
    mock_send_email.assert_called_once_with(
        to="customer@example.com",
        subject="Your order has been shipped"
    )
```

5. Тест пагинации товаров:

```
python
```

```
def test_product_pagination():
    # Создаем 25 товаров
    # Запрос page=1, count=10 - получаем 10 товаров
    # Запрос page=2, count=10 - получаем 10 товаров
    # Запрос page=3, count=10 - получаем 5 товаров

    # Проверяем: total_count, total_pages, has_next, has_prev
```

6. Изоляция тестов:

- Каждый тест запускается в отдельной транзакции
- Используем `session.rollback()` после теста
- Фикстуры с scope="function" создают новые объекты
- Важно для: предсказуемости, воспроизводимости, параллельного запуска

Выводы:

Тесты репозитория работают (5 passed), тесты сервиса работают (4 passed), тесты контроллера имеют проблемы с DI (но структура тестов есть), все необходимые компоненты для ЛР4 реализованы.