

Лабораторная работа №3

Фикстуры

```
# Фикстуры для базы данных
@pytest.fixture 2 usages
def db_connection():
    """Фикстура для создания временной базы данных в памяти"""
    connection = sqlite3.connect(":memory:")
    yield connection
    connection.close()

@pytest.fixture
def auth_manager(db_connection):
    """Фикстура для создания экземпляра AuthManager"""
    return AuthManager(db_connection)

@pytest.fixture 19 usages
def populated_auth_manager(auth_manager):
    """Фикстура с предзаполненными данными пользователей"""
    # Добавляем тестовых пользователей
    test_users = [
        ("user1", "pass1", "USA", 1000.0),
        ("user2", "pass2", "USA", 2000.0),
        ("user3", "pass3", "Canada", 1500.0),
        ("user4", "pass4", "UK", 3000.0),
        ("user5", "pass5", "Germany", 2500.0)
    ]
```

Базовые тесты:

1) Тест создания таблиц

```
170     for username, password, country, balance in test_users:
171         auth_manager.register_user(username, password, country, balance)
172
173     return auth_manager
174
175
176 # Базовые тесты (3 штуки)
177 def test_create_tables(auth_manager):
178     """Тест создания таблиц"""
179     cursor = auth_manager.connection.cursor()
180     cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='users'")
181     result = cursor.fetchone()
182     assert result is not None
183     assert result[0] == 'users'
```

lab3.test_create_tables x

0ms ✓ Tests passed: 1 of 1 test - 0ms

Launching pytest with arguments lab3.py::test_create_tables --no-header --no-summary -q in C:\Users\city...

===== test session starts =====

collecting ... collected 1 item

lab3.py::test_create_tables PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

2) Тест регистрации пользователей

```
> def test_register_user(auth_manager):
    """Тест регистрации пользователя"""
    auth_manager.register_user(username='testuser', password='testpass', country='TestCountry', balance=1000.0)

    cursor = auth_manager.connection.cursor()
    cursor.execute("SELECT * FROM users WHERE username = 'testuser'")
    user = cursor.fetchone()

    assert user is not None
    assert user[1] == 'testuser'
    assert user[2] == 'testpass'
    assert user[3] == 'TestCountry'
    assert user[4] == 1000.0

> def test_authenticate_user(auth_manager):
    """Тест аутентификации пользователя"""
    auth_manager.register_user(username='authuser', password='authpass', country='AuthCountry', balance=500.0)
```

test_register_user x

0ms ✓ Tests passed: 1 of 1 test - 0ms

===== test session starts =====

collecting ... collected 1 item

lab3.py::test_register_user PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

3) Тест аутентификации пользователей

```
199
200
201 > def test_authenticate_user(auth_manager):
202     """Тест аутентификации пользователя"""
203     auth_manager.register_user( username: "authuser", password: "authpass", country: "AuthCountry", balance: 500.0)
204
205     user = auth_manager.authenticate_user( username: "authuser", password: "authpass")
206     assert user is not None
207     assert user[1] == "authuser"
208
```

for lab3.test_authenticate_user x

0ms ✓ Tests passed: 1 of 1 test - 0ms

===== test session starts =====
collecting ... collected 1 item

lab3.py::test_authenticate_user PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

Параметрические тесты

1) Тест регистрации нескольких пользователей

```
1)
> def test_register_multiple_users(auth_manager, username, password, country, balance):
    """Параметризованный тест регистрации нескольких пользователей"""
    auth_manager.register_user(username, password, country, balance)

    user = auth_manager.authenticate_user(username, password)
    assert user is not None
    assert user[1] == username
    assert user[3] == country
    assert user[4] == balance
```

test_register_multiple_users x

0ms ✓ Tests passed: 3 of 3 tests - 0ms

collecting ... collected 3 items

lab3.py::test_register_multiple_users[param1-pass1-Country1-1000.0] PASSED [33%]
lab3.py::test_register_multiple_users[param2-pass2-Country2-2000.0] PASSED [66%]
lab3.py::test_register_multiple_users[param3-pass3-Country3-3000.0] PASSED [100%]

===== 3 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

2) Тест подсчета пользователей по странам

```
232     ("France", 0) # Несоответствующая страна
233 })
234 def test_count_users_by_country_parametrized(populated_auth_manager, country, expected_count):
235     """Параметризованный тест подсчета пользователей по странам"""
236     count = populated_auth_manager.count_users_by_country(country)
237     assert count == expected_count
238
239 @pytest.mark.parametrize("from_id,to_id,amount,expected_from,expected_to", [
240     (1, 2, 100.0, 900.0, 2100.0), # Перевод между пользователями USA
241     (3, 4, 500.0, 1000.0, 3500.0), # Перевод между разными странами
242     (5, 1, 250.0, 2250.0, 1250.0) # Обратный перевод
243 ])
244
245 def test_transfer_balance_parametrized(populated_auth_manager, from_id, to_id, amount, expected_from, expected_to):
246     """Параметризованный тест перевода средств"""
247     populated_auth_manager.transfer_balance(from_id, to_id, amount)
```

lab3.test_count_users_by_country_par... x

0ms ✓ Tests passed: 5 of 5 tests - 0ms

lab3.py::test_count_users_by_country_parametrized[USA-2]	PASSED	[20%]
lab3.py::test_count_users_by_country_parametrized[Canada-1]	PASSED	[40%]
lab3.py::test_count_users_by_country_parametrized[UK-1]	PASSED	[60%]
lab3.py::test_count_users_by_country_parametrized[Germany-1]	PASSED	[80%]
lab3.py::test_count_users_by_country_parametrized[France-0]	PASSED	[100%]

===== 5 passed, 2 warnings in 0.03s =====

Process finished with exit code 0

3) Тест перевода средств

```
244 })
245 def test_transfer_balance_parametrized(populated_auth_manager, from_id, to_id, amount, expected_from, expected_to):
246     """Параметризованный тест перевода средств"""
247     populated_auth_manager.transfer_balance(from_id, to_id, amount)
248
249     from_user = populated_auth_manager.get_user_by_id(from_id)
250     to_user = populated_auth_manager.get_user_by_id(to_id)
251
252     assert from_user[4] == expected_from
253     assert to_user[4] == expected_to
254
255 # Тестирование исключений (2 штуки)
256 def test_transfer_insufficient_funds(populated_auth_manager):
257     """Тест исключения при недостаточных средствах"""
258     with pytest.raises(ValueError):
259         populated_auth_manager.transfer_balance(1, 2, 1000.0)
```

lab3.test_transfer_balance_parametrized x

0ms ✓ Tests passed: 3 of 3 tests - 0ms

collecting ... collected 3 items

lab3.py::test_transfer_balance_parametrized[1-2-100.0-900.0-2100.0]	PASSED	[33%]
lab3.py::test_transfer_balance_parametrized[3-4-500.0-1000.0-3500.0]	PASSED	[66%]
lab3.py::test_transfer_balance_parametrized[5-1-250.0-2250.0-1250.0]	PASSED	[100%]

===== 3 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

Тестирование исключений

1) Тест исключения при недостаточных средствах

```
# Тестирование исключений (2 штуки)
> def test_transfer_insufficient_funds(populated_auth_manager):
    """Тест исключения при недостаточных средствах"""
    with pytest.raises(ValueError, match="Insufficient funds"):
        populated_auth_manager.transfer_balance(from_user_id: 1, to_user_id: 2)

> def test_register_duplicate_username(auth_manager):
    """Тест исключения при дублировании username (ожидаемое поведение SQLite)"""
    auth_manager.register_user(username="duplicate", password="pass1", country="Country1", balance=1000.0)

    with pytest.raises(sqlite3.IntegrityError):
        auth_manager.register_user(username="duplicate", password="pass2", country="Country2", balance=2000.0)

# Тесты с использованием фикстур
def test_delete_user_with_fixture(populated_auth_manager):
    """Тест удаления пользователя с использованием фикстуры populated_auth_manager"""
    initial_count = populated_auth_manager.count_users_by_country("USA")
    populated_auth_manager.delete_user(1) # Удаляем первого пользователя из USA
    new_count = populated_auth_manager.count_users_by_country("USA")
    assert initial_count == new_count + 1
```

Download grammar and spelling checker for Russian?
Download Russian Alt+Shift+Enter More actions... Alt+Enter
Parameter populated_auth_manager of lab3.test_transfer_in populated_auth_manager: AuthManager

0ms ✓ Tests passed: 1 of 1 test - 0ms

===== test session starts =====
collecting ... collected 1 item

lab3.py::test_transfer_insufficient_funds PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0

2) Тест исключения при дублировании username

```
def test_register_duplicate_username(auth_manager):
    """Тест исключения при дублировании username (ожидаемое поведение SQLite)"""
    auth_manager.register_user(username="duplicate", password="pass1", country="Country1", balance=1000.0)

    with pytest.raises(sqlite3.IntegrityError):
        auth_manager.register_user(username="duplicate", password="pass2", country="Country2", balance=2000.0)

# Тесты с использованием фикстур
def test_delete_user_with_fixture(populated_auth_manager):
    """Тест удаления пользователя с использованием фикстуры populated_auth_manager"""
    initial_count = populated_auth_manager.count_users_by_country("USA")
    populated_auth_manager.delete_user(1) # Удаляем первого пользователя из USA
    new_count = populated_auth_manager.count_users_by_country("USA")
    assert initial_count == new_count + 1
```

lab3.test_register_duplicate_username

38ms ✓ Tests passed: 1 of 1 test - 38ms

===== test session starts =====
collecting ... collected 1 item

lab3.py::test_register_duplicate_username PASSED [100%]

===== 1 passed, 2 warnings in 0.05s =====

Process finished with exit code 0

Тесты с использованием фикстур

- 1) Тест удаления пользователя с использованием фикстуры populated_auth_manager

```
272 > def test_delete_user_with_fixture(populated_auth_manager):
273     """Тест удаления пользователя с использованием фикстуры populated_auth_manager"""
274     initial_count = populated_auth_manager.count_users_by_country("USA")
275     populated_auth_manager.delete_user(1) # Удаляем первого пользователя из USA
276
277     new_count = populated_auth_manager.count_users_by_country("USA")
278     assert new_count == initial_count - 1
279
280     # Проверяем, что пользователь действительно удален
281     user = populated_auth_manager.get_user_by_id(1)
282     assert user is None
283
284
285 > def test_get_user_by_id_with_fixture(populated_auth_manager):
286     """Тест получения пользователя по ID с использованием фикстуры"""
287
lab3.test_delete_user_with_fixture x

38 ms ✓ Tests passed: 1 of 1 test - 38 ms

===== test session starts =====
collecting ... collected 1 item

lab3.py::test_delete_user_with_fixture PASSED [100%]

===== 1 passed, 2 warnings in 0.06s =====

Process finished with exit code 0
```

- 2) Тест получения пользователя по ID с использованием фикстуры

```
283
284
285 > def test_get_user_by_id_with_fixture(populated_auth_manager):
286     """Тест получения пользователя по ID с использованием фикстуры"""
287     user = populated_auth_manager.get_user_by_id(2) # Второй пользователь
288
289     assert user is not None
290     assert user[0] == 2 # ID
291     assert user[1] == "user2" # username
292     assert user[3] == "USA" # country
293
294
295 # Тесты с метками
296 @pytest.mark.slow
297 > def test_large_number_transfers(populated_auth_manager):
298     """Тест большого количества переводов с использованием фикстуры"""
299
lab3.test_get_user_by_id_with_fixture x

0 ms ✓ Tests passed: 1 of 1 test - 0 ms

===== test session starts =====
collecting ... collected 1 item

lab3.py::test_get_user_by_id_with_fixture PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0
```

Тесты с метками

1) Медленный тест множественных переводов

```
293
294
295 # Тесты с метками
296 @pytest.mark.slow
297 def test_large_number_transfers(populated_auth_manager):
298     """Медленный тест множественных переводов (помечен как slow)"""
299     # Выполняем много маленьких переводов
300     for i in range(100):
301         populated_auth_manager.transfer_balance(from_user_id=1, to_user_id=2, amount=1.0)
302
303     user1 = populated_auth_manager.get_user_by_id(1)
304     user2 = populated_auth_manager.get_user_by_id(2)
305
306     assert user1[4] == 1000.0 - 100.0 # Исходный баланс минус 100 переводов
307     assert user2[4] == 0.0 + 100.0 # Исходный баланс плюс 100 переводов
```

lab3.test_large_number_transfers x

1 ms ✓ Tests passed: 1 of 1 test – 1 ms

```
===== test session starts =====
collecting ... collected 1 item

lab3.py::test_large_number_transfers PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0
```

2) Интеграционный тест полного workflow

```
9
10
11 @pytest.mark.integration
12 def test_integration_workflow(auth_manager):
13     """Интеграционный тест полного workflow (помечен как integration)"""
14     # Регистрация
15     auth_manager.register_user(username="integration_user", password="integ_pass", country="IntegrationLand", balance=1000.0)
16
17     # Аутентификация
18     user = auth_manager.authenticate_user(username="integration_user", password="integ_pass")
19     assert user is not None
20
21     # Добавляем второго пользователя для перевода
22     auth_manager.register_user(username="receiver_user", password="receiver_pass", country="IntegrationLand", balance=500.0)
```

3.test_integration_workflow x

0 ms ✓ Tests passed: 1 of 1 test – 0 ms

```
===== test session starts =====
collecting ... collected 1 item

lab3.py::test_integration_workflow PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0
```

Тест демонстрации SQL-инъекции при аутентификации

```
351 def test_sql_injection_authenticate_user(auth_manager):
352     """Тест демонстрации SQL-инъекции при аутентификации"""
353     # Регистрация нормального пользователя
354     auth_manager.register_user( username: "normaluser", password: "normalpass", country: "Country", balance: 1000.0)
355
356     # Попробуем обойти аутентификацию с помощью SQL-инъекции
357     user = auth_manager.authenticate_user( username: "normaluser" OR '1'='1", password: "anypassword")
358
359     # Из-за уязвимости инъекция может сработать
360     # Этот тест демонстрирует проблему безопасности
361     assert user is not None
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
```

lab3.test_sql_injection_authenticate_us... x

0ms ✓ Tests passed: 1 of 1 test - 0ms

```
===== test session starts =====
collecting ... collected 1 item

lab3.py::test_sql_injection_authenticate_user PASSED [100%]

===== 1 passed, 2 warnings in 0.02s =====

Process finished with exit code 0
```

217/8 (54)