

Практическое задание №3

Задание:

Вам нужно протестировать класс AuthManager, который управляет пользователями, их аутентификацией, а также предоставляет функциональность для подсчета пользователей по странам и перевода средств между ними. В тестах вам нужно продемонстрировать несколько видов тестов: базовые(3 штуки), параметризованные(3 штуки), тестирование исключений(2 штуки), использование фикстур(базы данных) и меток(минимум 2).

ОБЯЗАТЕЛЬНО!!!: должен присутствовать тест на SQL инъекции.

Решение:

Код класса для тестирования

```
import sqlite3

class AuthManager:
    def __init__(self, connection):
        self.connection = connection
        self.create_tables()

    def create_tables(self):
        """Создание таблицы пользователей"""
        with self.connection:
            self.connection.execute("""
                CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL UNIQUE,
                    password TEXT NOT NULL,
                    country TEXT NOT NULL,
                    balance REAL NOT NULL
                )
            """)

    def register_user(self, username, password, country, balance):
        """Регистрация нового пользователя"""
        with self.connection:
            self.connection.execute(f"""
                INSERT INTO users (username, password, country, balance)
                VALUES ('{username}', '{password}', '{country}', {balance})
            """)

    def authenticate_user(self, username, password):
```

```

        """Аутентификация пользователя"""
        cursor = self.connection.cursor()
        cursor.execute(f"""
            SELECT * FROM users
            WHERE username = '{username}' AND password = '{password}'
        """)
        return cursor.fetchone()

def delete_user(self, user_id):
    """Удаление пользователя по ID"""
    with self.connection:
        self.connection.execute(f"""
            DELETE FROM users WHERE id = {user_id}
        """)

def get_user_by_id(self, user_id):
    """Получение пользователя по ID"""
    cursor = self.connection.cursor()
    cursor.execute(f"""
        SELECT * FROM users WHERE id = {user_id}
    """)
    return cursor.fetchone()

def count_users_by_country(self, country):
    """Подсчет пользователей по стране"""
    cursor = self.connection.cursor()
    cursor.execute(f"""
        SELECT COUNT(*) FROM users WHERE country = '{country}'
    """)
    return cursor.fetchone()[0]

def transfer_balance(self, from_user_id, to_user_id, amount):
    """Перевод средств между пользователями"""
    with self.connection:
        # Проверяем достаточность средств
        cursor = self.connection.cursor()
        cursor.execute(f"SELECT balance FROM users WHERE id = {from_user_id}")
        from_balance = cursor.fetchone()[0]

        if from_balance < amount:
            raise ValueError("Недостаточно средств для перевода")

        # Выполняем перевод
        self.connection.execute(f"""

```

```

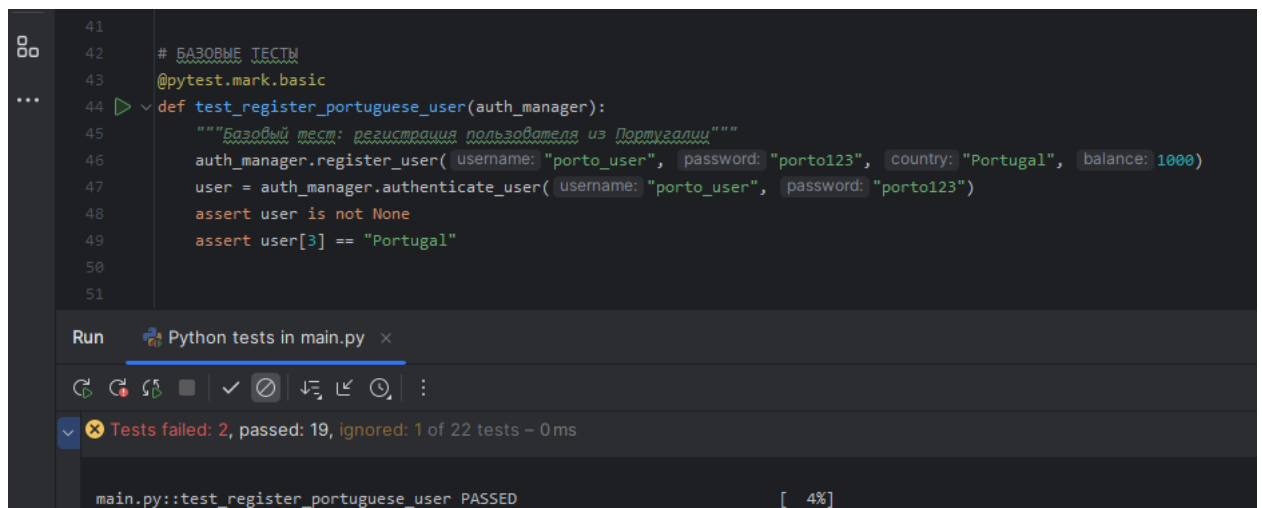
        UPDATE users SET balance = balance - {amount} WHERE id =
{from_user_id}
        """
        self.connection.execute(f"""
        UPDATE users SET balance = balance + {amount} WHERE id =
{to_user_id}
        """)

def get_all_users(self):
    """Получение всех пользователей (для тестирования)"""
    cursor = self.connection.cursor()
    cursor.execute("SELECT * FROM users")
    return cursor.fetchall()

def update_user_balance(self, user_id, new_balance):
    """Обновление баланса пользователя"""
    with self.connection:
        self.connection.execute(f"""
        UPDATE users SET balance = {new_balance} WHERE id = {user_id}
        """)

```

1. Базовые тесты(3 штуки)



The screenshot shows a code editor with Python test code and a terminal window. The code defines a test function `test_register_portuguese_user` that registers a user and authenticates them. The terminal shows the test results.

```

41
42 # БАЗОВЫЕ ТЕСТЫ
43 @pytest.mark.basic
44 def test_register_portuguese_user(auth_manager):
45     """Базовый тест: регистрация пользователя из Португалии"""
46     auth_manager.register_user( username: "porto_user", password: "porto123", country: "Portugal", balance: 1000)
47     user = auth_manager.authenticate_user( username: "porto_user", password: "porto123")
48     assert user is not None
49     assert user[3] == "Portugal"
50
51
Run Python tests in main.py x
[ 4%]
Tests failed: 2, passed: 19, ignored: 1 of 22 tests - 0ms
main.py::test_register_portuguese_user PASSED

```

```
52 @pytest.mark.basic
53 def test_authenticate_french_user(european_users):
54     """Базовый тест: успешная аутентификация французского пользователя"""
55     user = european_users.authenticate_user(username="marie", password="pass456")
56     assert user[1] == "marie"
57     assert user[3] == "France"
58     assert user[4] == 2000
59
60
```

Run Python tests in main.py

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

main.py::test_authenticate_french_user PASSED [9%]

2

```
61 @pytest.mark.basic
62 def test_count_german_users(european_users):
63     """Базовый тест: подсчет пользователей из Германии"""
64     count = european_users.count_users_by_country("Germany")
65     assert count == 1
66
```

Run Python tests in main.py

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

main.py::test_count_german_users PASSED [13%]

3

2. Параметризованные тесты(3 штуки)

```
68 # ПАРАМЕТРИЗОВАННЫЕ ТЕСТЫ
69 @pytest.mark.parametrize("country,expected_count", [
70     ("Portugal", 2),
71     ("France", 1),
72     ("Denmark", 1),
73     ("Czechia", 1),
74     ("Poland", 0),
75     ("Norway", 0)
76 ])
77 def test_count_european_users(auth_manager, country, expected_count):
78     """Параметризованный тест: подсчет пользователей по европейским странам"""
79     # Подготовка данных для Европы
80     auth_manager.register_user(username="user_pt", password="pass1", country="Portugal", balance=1000)
81     auth_manager.register_user(username="user_fr", password="pass2", country="France", balance=1500)
82     auth_manager.register_user(username="user_dk", password="pass3", country="Denmark", balance=1200)
83     auth_manager.register_user(username="user_cz", password="pass4", country="Czechia", balance=1800)
84     auth_manager.register_user(username="user_de", password="pass5", country="Germany", balance=2000)
85     auth_manager.register_user(username="user_it", password="pass6", country="Italy", balance=1700)
86
87     count = auth_manager.count_users_by_country(country)
88     assert count == expected_count
89
90
```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

main.py::test_count_european_users[Portugal-2]
main.py::test_count_european_users[France-1]
main.py::test_count_european_users[Denmark-1]
main.py::test_count_european_users[Czechia-1]
main.py::test_count_european_users[Poland-0]
main.py::test_count_european_users[Norway-0]

PASSED [22%]PASSED [27%]PASSED [31%]PASSED [36%]PASSED [40%]

```
97 def test_register_european_users(auth_manager, username, password, country, balance):
98     """Параметризованный тест: регистрация пользователей из разных европейских стран"""
99     auth_manager.register_user(username, password, country, balance)
100     user = auth_manager.authenticate_user(username, password)
101     assert user is not None
102     assert user[3] == country
103     assert user[4] == balance
104
105
106 @pytest.mark.parametrize("from_country,to_country,transfer_amount,expected_from_balance,expected_to_balance", [
107     ("Portugal", "France", 300, 1200, 2300),
108     ("Denmark", "Czechia", 500, 700, 2300),
109     ("Germany", "Italy", 200, 2000, 1900)
110 ])
111
112
113 Run Python tests in main.py x
114
115 Tests failed: 2, passed: 19, ignored: 1 of 22 tests - 0 ms
116
117 main.py::test_register_european_users[paris_user-paris_pass-France-3000]
118 main.py::test_register_european_users[copenhagen_user-copenhagen_pass-Denmark-1800]
119 main.py::test_register_european_users[prague_user-prague_pass-Czechia-2200]
```

2

```
111 def test_transfer_between_european_countries(european_users, from_country, to_country, transfer_amount,
112     expected_from_balance, expected_to_balance):
113     """Параметризованный тест: перевод средств между европейскими странами"""
114     # Находим пользователей по странам
115     from_user = None
116     to_user = None
117
118     # Простой поиск пользователей по стране (в реальном приложении был бы метод)
119     for username in ["antonio", "marie", "lars", "petr", "sophie", "giovanni"]:
120         user = european_users.authenticate_user(username, password=f"pass{username[-3:]}")
121         if user and user[3] == from_country:
122             from_user = user
123         if user and user[3] == to_country:
124             to_user = user
125
126     if from_user and to_user:
127         original_from_balance = from_user[4]
128         original_to_balance = to_user[4]
129
130         european_users.transfer_balance(from_user[0], to_user[0], transfer_amount)
131
132         # Проверяем новые балансы
133         updated_from = european_users.get_user_by_id(from_user[0])
134         updated_to = european_users.get_user_by_id(to_user[0])
135
136         assert updated_from[4] == original_from_balance - transfer_amount
137         assert updated_to[4] == original_to_balance + transfer_amount
138
139
140 Run Python tests in main.py x
141
142 Tests failed: 2, passed: 19, ignored: 1 of 22 tests - 0 ms
143
144 main.py::test_transfer_between_european_countries[Portugal-France-300-1200-2300] PASSED [ 45%]PASSED [ 50%]PASSED [ 54%]PASSED [ 59%]
145 main.py::test_transfer_between_european_countries[Denmark-Czechia-500-700-2300]
146 main.py::test_transfer_between_european_countries[Germany-Italy-200-2000-1900]
```

3

3. Тестирование исключений(2 штуки)

```
140 # ТЕСТИРОВАНИЕ ИСКЛЮЧЕНИЙ
141 @pytest.mark.exception
142 def test_transfer_insufficient_funds_denmark_to_france(european_users):
143     """Тест исключения: недостаточно средств у датского пользователя для перевода во Францию"""
144     dk_user = european_users.authenticate_user( username: "lars", password: "pass789")
145     fr_user = european_users.authenticate_user( username: "marie", password: "pass456")
146
147     with pytest.raises(ValueError, match="Insufficient funds"):
148         european_users.transfer_balance(dk_user[0], fr_user[0], amount: 1500)
```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

main.py::test_transfer_insufficient_funds_denmark_to_france PASSED [63%]PASSED [68%]PASSED [72%]FAILED [77%]

1

```
152 @pytest.mark.exception
153 def test_authenticate_nonexistent_spanish_user(european_users):
154     """Тест исключения: аутентификация несуществующего испанского пользователя"""
155     user = european_users.authenticate_user( username: "nonexistent_spanish", password: "wrongpass")
156     assert user is None
```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

main.py::test_authenticate_nonexistent_spanish_user PASSED [81%]

2

4. Тесты с использованием фикстур базы данных(1 штука)

```

6  # Фикстура для базы данных в памяти
7  @pytest.fixture 2 usages
8  def db_connection():
9      conn = sqlite3.connect(':memory:')
10     yield conn
11     conn.close()
12
13
14  @pytest.fixture 34 usages
15  def auth_manager(db_connection):
16      return AuthManager(db_connection)
17
18
19  # Фикстура с тестовыми данными
20  @pytest.fixture 17 usages
21  def european_users(auth_manager):
22      """Фикстура с предзаполненными данными пользователей из Европы"""
23      users_data = [
24          ("antonio", "pass123", "Portugal", 1500),
25          ("marie", "pass456", "France", 2000),
26          ("lars", "pass789", "Denmark", 1200),
27          ("petr", "pass000", "Czechia", 1800),
28          ("sophie", "pass111", "Germany", 2200),
29          ("giovanni", "pass222", "Italy", 1700),
30          ("carlos", "pass333", "Spain", 1900),
31          ("anna", "pass444", "Sweden", 1600),
32          ("jan", "pass555", "Netherlands", 1400),
33          ("eva", "pass666", "Belgium", 1300)
34      ]
35
36      for username, password, country, balance in users_data:
37          auth_manager.register_user(username, password, country, balance)
38
39      return auth_manager
40

```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

```

main.py::test_count_german_users PASSED [ 13%]
main.py::test_count_european_users[Portugal-2]
main.py::test_count_european_users[France-1]
main.py::test_count_european_users[Denmark-1]
main.py::test_count_european_users[Czechia-1]

```


5. Тесты с метками(2 штуки)

```

174 # ТЕСТЫ С МЕТКАМИ ДЛЯ ФИЛЬТРАЦИИ
175 @pytest.mark.slow
176 @pytest.mark.europe
177 def test_performance_all_european_countries(european_users):
178     """Медленный тест: работа со всеми европейскими странами"""
179     european_countries = ["Portugal", "France", "Denmark", "Czechia", "Germany",
180                          "Italy", "Spain", "Sweden", "Netherlands", "Belgium"]
181
182     total_users = 0
183     for country in european_countries:
184         count = european_users.count_users_by_country(country)
185         total_users += count
186         print(f"Users in {country}: {count}")
187
188     assert total_users == 10 # Всего 10 пользователей в фикстуре
189

```

Run Python tests in main.py ✕

✖ Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0 ms

```
main.py::test_performance_all_european_countries PASSED [ 90%]Users in Portugal: 1
Users in France: 1
Users in Denmark: 1
Users in Czechia: 1
Users in Germany: 1
Users in Italy: 1
Users in Spain: 1
Users in Sweden: 1
Users in Netherlands: 1
Users in Belgium: 1
```

```
191 @pytest.mark.integration
192 @pytest.mark.europe
193 def test_european_user_full_lifecycle(auth_manager):
194     """Интеграционный тест: полный жизненный цикл европейского пользователя"""
195
196     auth_manager.register_user( username="czech_user", password="prague123", country="Czechia", balance=5000)
197
198
199     user = auth_manager.authenticate_user( username="czech_user", password="prague123")
200     assert user is not None
201     assert user[3] == "Czechia"
202
203
204     count = auth_manager.count_users_by_country("Czechia")
205     assert count == 1
206
207
208     auth_manager.register_user( username="portuguese_user", password="lisbon456", country="Portugal", balance=3000)
209     pt_user = auth_manager.authenticate_user( username="portuguese_user", password="lisbon456")
210
211
212     auth_manager.transfer_balance(user[0], pt_user[0], amount=1000)
213
214
215     updated_cz = auth_manager.get_user_by_id(user[0])
216     updated_pt = auth_manager.get_user_by_id(pt_user[0])
217
218     assert updated_cz[4] == 4000 # 5000 - 1000
219     assert updated_pt[4] == 4000 # 3000 + 1000
220
221
222     auth_manager.delete_user(user[0])
223     auth_manager.delete_user(pt_user[0])
224
225
226     assert auth_manager.count_users_by_country("Czechia") == 0
227     assert auth_manager.count_users_by_country("Portugal") == 0
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
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
```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests - 0 ms

main.py::test_european_user_full_lifecycle PASSED [95%]

6. Тестирование различных векторов SQL-инъекций(1 штука)

```
158 # ТЕСТ НА SQL ИНЪЕКЦИИ
159 @pytest.mark.security
160 def test_sql_injection_european_countries(auth_manager):
161     """Тест на уязвимость SQL инъекции с европейскими странами"""
162     auth_manager.register_user( username="europe_user", password="europe_pass", country="Portugal", balance=1000)
163
164
165     malicious_country = "Portugal' OR '1'='1' --"
166
167
168     count = auth_manager.count_users_by_country(malicious_country)
169
170     print(f"SQL Injection country test result: {count}")
171
172
```

Run Python tests in main.py x

Tests failed: 2, passed: 19, ignored: 1 of 22 tests – 0ms

main.py::test_sql_injection_european_countries PASSED [86%]SQL Injection country test result: 1