Szoftvertesztelési Projekt

- 2025 -

Készítette
Fazakas Nimród
Számítástechnika IV. - A

1. Bevezetés

Ez a dokumentum egy szoftvertesztelési projekt megvalósítását és eredményeit mutatja be, amely az egységtesztelésre összpontosít. A projekt fő célja az volt, hogy egységteszteket fejlesszünk és implementáljunk egy meglévő munkavállalói menedzsment rendszerhez, biztosítva a kódbázis megfelelő működését és megbízhatóságát.

2. A kiválasztott tesztelési keretrendszer leírása

2.1 A pytest választásának indokai

A pytest keretrendszert több meggyőző okból választottuk ehhez a projekthez:

- Ez az egyik legszélesebb körben használt nyílt forráskódú Python tesztelési keretrendszer
- Átfogó támogatást nyújt az egységteszteléshez, funkcionális teszteléshez és API teszteléshez
- Lehetővé teszi kompakt és egyszerű tesztkészletek létrehozását minimális boilerplate kóddal
- Nagymértékben bővíthető gazdag plugin ökoszisztémán keresztül
- Kiváló dokumentációval és nagy aktív közösséggel rendelkezik a támogatáshoz
- A fixtures rendszere tiszta módot biztosít a tesztfüggőségek és beállítások kezelésére

A unittest és nose2 alternatívákhoz képest a pytest modernebb, kifejezőbb szintaxist és hatékonyabb funkciókat kínál, amelyek ideálissá teszik ehhez a projekthez.

2.2 Telepítési Folyamat

A pytest telepítése egyszerű volt a pip, a Python csomagkezelőjének használatával:

```
# Virtuális környezet létrehozása és aktiválása
python -m venv pytest-env
# Windows rendszeren
pytest-env\Scripts\activate

# Pytest telepítése
pip install pytest
```

3. A projekt repository leírása

3.1 Repository struktúra

```
.
├── calculator.py                # Egyszerű számológép példa
├── calculator_test.py           # Tesztek a számológép példához
├── employee.py                  # Employee osztály definíciója
├── employee_manager.py          # Fizetés számítás és értesítés logika
├── relations_manager.py         # Csapatkezelési funkcionalitás
├── requirements.txt             # Projekt függőségek
├── test_employee_manager.py     # Tesztek az EmployeeManager-hez
├── test_relations_manager.py    # Tesztek a RelationsManager-hez
└── .github
    └── workflows
        └── pytest.yml           # CI/CD konfiguráció
```

3.2 Az adatok strukturálása a repository-ban

A repository a következő szervezési elveket követi:

- Minden forráskód fájlhoz tartozik egy megfelelő teszt fájl
- A teszt fájlok "test_" előtaggal rendelkeznek, hogy a pytest automatikusan felismerje
- A tesztek fixture-öket használnak a beállítási kód megosztására
- Az assertion üzenetek tartalmazzák a világos visszajelzést
- A teszt készletek logikusan csoportosítva vannak a tesztelt modul szerint

4. A megvalósított egységtesztek leírása

4.1 RelationsManager tesztek

4.1.1 Teszt: John Doe mint csapatvezető, specifikus születési dátummal

Bemenet: RelationsManager példány
Várt Kimenet: John Doe létezik, csapatvezető, és a születési dátuma 1970.01.31
Leírás: Ez a teszt ellenőrzi, hogy létezik-e egy John Doe nevű alkalmazott a rendszerben, aki csapatvezetőként van kijelölve és a megfelelő születési dátummal rendelkezik.

```python
def test_john_doe_is_team_leader_with_correct_birthdate(relations_manager):
    """Check if there is a team leader called John Doe whose birthdate is 31.01.1970."""
    # Find John Doe in the employee list
    john_doe = next((emp for emp in relations_manager.get_all_employees()
                     if emp.first_name == "John" and emp.last_name == "Doe"), None)

    # Assert John Doe exists, is a team leader, and has the correct birthdate
    assert john_doe is not None, "John Doe should exist in the employee list"
    assert relations_manager.is_leader(john_doe), "John Doe should be a team leader"
    assert john_doe.birth_date == datetime.date(1970, 1, 31), "John Doe's birthdate should be 31.01.1970"
```

4.1.2 Teszt: John Doe csapattagjai

Bemenet: RelationsManager példány
Várt Kimenet: John Doe csapata tartalmazza Myrta Torkelson-t és Jettie Lynch-et
Leírás: Ez a teszt ellenőrzi, hogy a csapatkezelési rendszer helyesen rendeli-e hozzá a csapattagokat John Doe csapatához.

```python
def test_john_doe_team_members(relations_manager):
    """Check if John Doe's team members are Myrta Torkelson and Jettie Lynch."""
    # Find John Doe
    john_doe = next((emp for emp in relations_manager.get_all_employees()
                     if emp.first_name == "John" and emp.last_name == "Doe"), None)

    # Get John's team members (should now be Employee objects, not IDs)
    team_members = relations_manager.get_team_members(john_doe)

    # Check if Myrta and Jettie are in John's team
    myrta = next((emp for emp in team_members
                  if emp.first_name == "Myrta" and emp.last_name == "Torkelson"), None)
    jettie = next((emp for emp in team_members
                   if emp.first_name == "Jettie" and emp.last_name == "Lynch"), None)

    assert myrta is not None, "Myrta Torkelson should be in John Doe's team"
    assert jettie is not None, "Jettie Lynch should be in John Doe's team"
    assert len(team_members) == 2, "John Doe should have exactly 2 team members"
```

4.1.3 Teszt: Tomas Andre nincs John Doe csapatában

Bemenet: RelationsManager példány
Várt Kimenet: Tomas Andre nincs felsorolva John Doe csapattagjai között
Leírás: Ez a teszt ellenőrzi, hogy a más csapatokból származó alkalmazottak nincsenek-e helytelenül John Doe csapatához rendelve.

```python
def test_tomas_andre_not_in_john_doe_team(relations_manager):
    """Make sure that Tomas Andre is not John Doe's team member."""
    # Find John Doe and Tomas Andre
    john_doe = next((emp for emp in relations_manager.get_all_employees()
                     if emp.first_name == "John" and emp.last_name == "Doe"), None)
    tomas_andre = next((emp for emp in relations_manager.get_all_employees()
                        if emp.first_name == "Tomas" and emp.last_name == "Andre"), None)

    # Get John's team member IDs
    team_member_ids = relations_manager.get_team_members(john_doe)

    assert tomas_andre.id not in team_member_ids, "Tomas Andre should not be in John Doe's team"
```

4.1.4 Teszt: Gretchen Walford alapfizetése

Bemenet: RelationsManager példány
Várt Kimenet: Gretchen Walford alapfizetése 4000$
Leírás: Ez a teszt ellenőrzi, hogy az alkalmazotti adatrendszer helyesen tárolja-e és kéri le a fizetési információkat.

```python
def test_gretchen_walford_salary(relations_manager):
    """Check if Gretchen Walford's base salary equals 4000$."""
    # Find Gretchen Walford
    gretchen = next((emp for emp in relations_manager.get_all_employees()
                     if emp.first_name == "Gretchen" and emp.last_name == "Watford"), None)

    assert gretchen is not None, "Gretchen Walford should exist in the employee list"
    assert gretchen.base_salary == 4000, "Gretchen Walford's base salary should be 4000$"
```

4.1.5 Teszt: Tomas Andre nem csapatvezető

Bemenet: RelationsManager példány
Várt Kimenet: Tomas Andre nem csapatvezető, és a csapattagjainak lekérése üres listát eredményez
Leírás: Ez a teszt ellenőrzi a rendszer viselkedését, amikor egy nem vezető csapattagjait próbáljuk elérni.

```python
def test_tomas_andre_not_team_leader(relations_manager):
    """Make sure Tomas Andre is not a team leader. Check what happens if you try to retrieve his team members."""
    # Find Tomas Andre
    tomas_andre = next((emp for emp in relations_manager.get_all_employees()
                        if emp.first_name == "Tomas" and emp.last_name == "Andre"), None)

    assert not relations_manager.is_leader(tomas_andre), "Tomas Andre should not be a team leader"

    # Check what happens when trying to get team members for a non-leader
    team_members = relations_manager.get_team_members(tomas_andre)
    assert team_members == [], "Getting team members for a non-leader should return an empty list"
```

4.1.6 Teszt: Jude Overcash nincs az adatbázisban

Bemenet: RelationsManager példány
Várt Kimenet: Nincs Jude Overcash nevű alkalmazott a rendszerben
Leírás: Ez a teszt ellenőrzi, hogy a rendszer nem tartalmaz-e adatokat nem létező alkalmazottakról.

```
def test_jude_overcash_not_in_database(relations_manager):
    """Make sure that Jude Overcash is not stored in the database."""
    # Check if anyone named Jude Overcash exists in employee list
    jude_overcash = next((emp for emp in relations_manager.get_all_employees()
                          if emp.first_name == "Jude" and emp.last_name == "Overcash"), None)

    assert jude_overcash is None, "Jude Overcash should not exist in the employee list"
```

4.2 EmployeeManager tesztek

4.2.1 Teszt: Normál alkalmazott fizetésszámítása

Bemenet: Alkalmazott, aki nem csapatvezető, 1998.10.10-én lett felvéve, 1000$ alapfizetéssel
Várt Kimenet: Kiszámított fizetés 3000$ (1000$ + 20 × 100$)
Leírás: Ez a teszt ellenőrzi a fizetésszámítási logikát normál alkalmazottak esetében, figyelembe véve a szolgálati éveket.

```
def test_regular_employee_salary(employee_manager):
    """
    Check an employee's salary who is not a team leader whose hire date is 10.10.1998
    and his base salary is 1000$. Make sure the returned value is 3000$ (1000$ + 20 X 100$).
    """
    # Create a test employee who is not a team leader
    today = datetime.date.today()
    years_employed = 20  # According to the requirement (1000$ + 20 X 100$)
    hire_date = datetime.date(today.year - years_employed, 10, 10)

    test_employee = Employee(
        id=100,  # ID that doesn't exist in the teams dictionary
        first_name="Test",
        last_name="Employee",
        birth_date=datetime.date(1980, 1, 1),
        base_salary=1000,
        hire_date=hire_date
    )

    # Calculate salary
    salary = employee_manager.calculate_salary(test_employee)

    # Expected: base_salary + (years * yearly_bonus)
    expected_salary = 1000 + (years_employed * 100)
    assert salary == expected_salary, f"Expected salary of {expected_salary}$, got {salary}$"
```

4.2.2 Teszt: Csapatvezető fizetésszámítása

Bemenet: Alkalmazott, aki csapatvezető 3 csapattaggal, 2008.10.10-én lett felvéve, 2000$ alapfizetéssel
Várt Kimenet: Kiszámított fizetés 3600$ (2000$ + 10 × 100$ + 3 × 200$)
Leírás: Ez a teszt ellenőrzi a fizetésszámítási logikát csapatvezetők esetében, figyelembe véve a szolgálati éveket és a csapatméret bónuszokat.

```python
def test_team_leader_salary(employee_manager):
    """
    Check an employee's salary who is a team leader and his team consists of 3 members.
    She was hired on 10.10.2008 and has a base salary of 2000$.
    Validate if the returned value is 3600$ (2000$ + 10 X 100$ + 3 X 200$).
    """
    # Create a mock relations_manager with a custom team leader
    mock_relations_manager = MagicMock()

    # Today's date for year calculation
    today = datetime.date.today()
    years_employed = 10  # According to the requirement (2000$ + 10 X 100$ + 3 X 200$)
    hire_date = datetime.date(today.year - years_employed, 10, 10)

    # Create test team leader
    team_leader = Employee(
        id=101,
        first_name="Team",
        last_name="Leader",
        birth_date=datetime.date(1975, 5, 15),
        base_salary=2000,
        hire_date=hire_date
    )

    # Configure mocks
    mock_relations_manager.is_leader.return_value = True
    mock_relations_manager.get_team_members.return_value = [201, 202, 203]  # 3 team members

    # Create employee manager with mock
    custom_employee_manager = EmployeeManager(mock_relations_manager)

    # Calculate salary
    salary = custom_employee_manager.calculate_salary(team_leader)

    # Expected: base_salary + (years * yearly_bonus) + (team_members * leader_bonus_per_member)
    expected_salary = 2000 + (years_employed * 100) + (3 * 200)
    assert salary == expected_salary, f"Expected salary of {expected_salary}$, got {salary}$"
```

4.2.3 Teszt: E-mail értesítés a fizetés átutalásáról

Bemenet: Employee példány
Várt Kimenet: E-mail értesítés a helyes alkalmazott névvel és kiszámított fizetéssel
Leírás: Ez a teszt ellenőrzi, hogy az e-mail értesítési rendszer helyesen formázza-e az üzeneteket az alkalmazott információival és fizetési részleteivel.

```python
def test_salary_email_notification(employee_manager):
    """
    Make sure that when you calculate the salary and send an email notification,
    the respective email sender service is used with the correct information.
    """
    # Find an employee to test with
    relations_manager = RelationsManager()
    test_employee = relations_manager.get_all_employees()[0]  # Use the first employee

    # Mock the print function to check if it's called with the right message
    with patch('builtins.print') as mock_print:
        # Call the method that sends the notification
        employee_manager.calculate_salary_and_send_email(test_employee)

        # Check if print was called with the correct message format
        expected_message = f"{test_employee.first_name} {test_employee.last_name} your "
        "salary: {employee_manager.calculate_salary(test_employee)} has been transferred to you."
        mock_print.assert_called_with(expected_message)
```

5. Folyamatos integráció (CI) beállítása

5.1 GitHub Actions munkafolyamat

A folyamatos integráció engedélyezéséhez GitHub Actions munkafolyamatot implementáltunk a tesztek automatikus futtatására kódváltoztatások esetén:

```yaml
name: Python Tests

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: "3.10"

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install pytest
          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi

      - name: Test with pytest
        run: |
          pytest
```

5.2 CI/CD munkafolyamat előnyei

Az implementált CI/CD pipeline több előnyt is kínál:

- Automatikus tesztfuttatás minden kódpush esetén
- Regressziók és hibák korai felismerése
- Konzisztens tesztkörnyezet
- Látható tesztállapot a GitHub felületén keresztül

```
▶ Run pytest
============================= test session starts ==============================
platform linux -- Python 3.10.16, pytest-8.3.5, pluggy-1.5.0
rootdir: /home/runner/work/Software-Testing/Software-Testing
collected 12 items

calculator_test.py ...                                                   [ 25%]
test_employee_manager.py ...                                             [ 50%]
test_relations_manager.py ......                                         [100%]

============================== 12 passed in 0.07s ==============================
```

6. Tesztfuttatási Eredmények

6.1 Teszt Összefoglaló

Minden implementált teszt sikeresen fut, megerősítve az alkalmazott-menedzsment rendszer helyes működését.

```
PS C:\SAPIENTIA EMTE\Negyedik ev\2nd_semester\Software_Testing\Software-Testing> pytest -v
============================= test session starts ==============================
platform win32 -- Python 3.10.7, pytest-8.3.4, pluggy-1.5.0 -- C:\SAPIENTIA EMTE\Negyedik ev\2nd_semester\Software_Testing\Software-Testing\pytest-env\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\SAPIENTIA EMTE\Negyedik ev\2nd_semester\Software_Testing\Software-Testing
collected 12 items

calculator_test.py::test_calc_addition PASSED                            [  8%]
calculator_test.py::test_calc_substraction PASSED                        [ 16%]
calculator_test.py::test_calc_multiply PASSED                            [ 25%]
test_employee_manager.py::test_regular_employee_salary PASSED            [ 33%]
test_employee_manager.py::test_team_leader_salary PASSED                 [ 41%]
test_employee_manager.py::test_salary_email_notification PASSED          [ 50%]
test_relations_manager.py::test_john_doe_is_team_leader_with_correct_birthdate PASSED [ 58%]
test_relations_manager.py::test_john_doe_team_members PASSED             [ 66%]
test_relations_manager.py::test_tomas_andre_not_in_john_doe_team PASSED  [ 75%]
test_relations_manager.py::test_gretchen_walford_salary PASSED           [ 83%]
test_relations_manager.py::test_tomas_andre_not_team_leader PASSED       [ 91%]
test_relations_manager.py::test_jude_overcash_not_in_database PASSED     [100%]

============================== 12 passed in 0.10s ==============================
```

6.2 Kód Lefedettség

Bár nem kifejezetten követelmény, a kód lefedettségi statisztikák betekintést nyújtanak a tesztek alaposságába:

```
============================= test session starts =============================
platform win32 -- Python 3.10.7, pytest-8.3.4, pluggy-1.5.0
rootdir: C:\SAPIENTIA EMTE\Negyedik ev\2nd_semester\Software_Testing\Software-Test
ing
plugins: cov-6.0.0
collected 12 items

calculator_test.py ...                                                   [ 25%]
test_employee_manager.py ...                                             [ 50%]
test_relations_manager.py ......                                         [100%]

---------- coverage: platform win32, python 3.10.7-final-0 -----------
Name                         Stmts   Miss  Cover
------------------------------------------------
calculator.py                    6      0   100%
calculator_test.py              10      0   100%
employee.py                     10      0   100%
employee_manager.py             27      6    78%
relations_manager.py            17      0   100%
test_employee_manager.py        39      0   100%
test_relations_manager.py       37      0   100%
------------------------------------------------
TOTAL                          146      6    96%


============================== 12 passed in 0.20s ==============================
```

7. Következtetés

Ez a projekt sikeresen implementált egy átfogó egységtesztelési készletet egy alkalmazott-menedzsment rendszerhez. A pytest használatával biztosítottuk, hogy a rendszer alapvető funkcionalitása helyesen működik.

A tesztek olyan kritikus szempontokat ellenőriznek, mint a csapatstruktúra kezelése, az alkalmazotti adatok integritása és a fizetésszámítási logika. Az implementált folyamatos integrációs (CI) pipeline tovább javítja a projektet, az automatizált tesztfuttatás biztosításával.

8. Hivatkozások

1. Pytest dokumentáció

   https://docs.pytest.org/

2. Python Testing with pytest (Brian Okken):

   https://pragprog.com/titles/bopytest2/python-testing-with-pytest-second-edition/

3. GitHub Actions dokumentáció

   https://docs.github.com/en/actions