

Міністерство освіти і науки, молоді та спорту України
Національний Технічний Університет України
«Київський Політехнічний Інститут»
Навчально-науковий комплекс
«Інститут прикладного системного аналізу»
Кафедра системного проектування

Лабораторна робота №5

З курсу: «Проектування інформаційних систем»

На тему: «Модульне тестування (Unit-тести) та рефакторинг»

Виконала:

Студентка 4 курсу

Групи ДА-72

Когінова А.Д.

Мета роботи: оволодіти навичками створення програмного забезпечення за методологією TDD та ознайомитися з процедурами рефакторинга.

Завдання:

1. Використовувати методологію Test Driven Development для створення класів архітектурної програмної моделі.
2. Скласти тестові сценарії, які продемонструють функціонування всіх методів проектованої моделі.
3. Виконати юніт-тестування складових частин (внутрішніх класів), що реалізують об'єкт моделювання.
4. Виконати "зовнішнє" юніт-тестування для API.
5. Провести рефакторинг коду програми, для поліпшення реалізації.

Хід виконання лабораторної роботи:

testUser.py

Містить тести для перевірки коректної роботи функцій, пов'язаних з юзерами (процедури реєстрації та авторизації), а саме:

- Перевірка роботи форми реєстрації
- Перевірка авторизації користувача
- Перевірка роботи форми логіну

```
from django.test import TestCase, Client
from user.models import Favorite
from user.forms import UserLoginForm, UserRegisterForm
from django.contrib.auth import (
    authenticate,
    login,
)

class UserTestClass(TestCase):

    def setUp(self):
        self.user_data = {
            'username': 'AlisA',
            'password': 'hdbeqnlq2eru173feo',
```

```

    }

    def test_user_registartion_validdidation(self):
        form = UserRegisterForm(self.user_data)
        self.assertTrue(form.is_valid())

        wrong_registration_data = {
            'username': 'A',
            'password': '',
        }
        form = UserRegisterForm(wrong_registration_data)
        self.assertFalse(form.is_valid())

    def test_user_authentication(self):
        form = UserRegisterForm(self.user_data)

        if form.is_valid():
            user = form.save(commit=False)
            password = form.cleaned_data.get('password')
            user.set_password(password)
            user.save()
            self.assertTrue(authenticate(username=user.username, password=password))
            resp = self.client.get('/login')
            self.assertEqual(resp.status_code, 200)

            resp = self.client.get('/logout')
            self.assertEqual(resp.status_code, 302) # redirection

    def test_user_login_validation(self):
        form = UserLoginForm(self.user_data)

        if form.is_valid():
            username = form.cleaned_data.get("username")
            password = form.cleaned_data.get("password")
            user = authenticate(username=username, password=password)
            c = Client()
            c.get('/login', self.user_data)
            self.assertTrue(login(c.get('/login'), user))

```

testCore.py - Містить тести для перевірки коректної роботи функцій пошуку фільмів та взаємодії із списком улюблених фільмів, а саме:

- Перевірка додавання фільму до папки favorites
- Перевірка видалення фільму з папки favorites
- Перевірка роботи форми пошуку фільмів
- Перевірка коректності роботи програми, при знайденому фільмі та коли фільм знайдено не було

```

import pandas as pd
from django.test import TestCase
from user.models import Favorite
from core.forms import SearchForm

class UserTestClass(TestCase):

    def setUp(self):
        self.film_name = 'Test2'
        Favorite.objects.create(myList='Test1')

    def test_created_favourite(self):
        fav_obj = Favorite.objects.get(myList='Test1')
        field_label = fav_obj._meta.get_field('myList').verbose_name
        self.assertEqual(field_label, 'myList')

    def test_add_to_favourites(self):
        model = Favorite()
        model.myList = self.film_name
        model.save()
        fav_obj = Favorite.objects.get(myList=self.film_name).myList
        self.assertEqual(fav_obj, self.film_name)

    def test_remove_from_favourites(self):
        for i in Favorite.objects.all():
            if i.myList == self.film_name:
                Favorite.objects.filter(myList=self.film_name).delete()
        self.assertFalse(Favorite.objects.filter(myList=self.film_name))

    def test_search_form_valid(self):
        form_data = {'search': 'Jumanji'}
        form = SearchForm(data=form_data)
        self.assertTrue(form.is_valid())

    def test_search_film_founded(self):
        df = pd.read_csv('C:\movies\core\movies_metadata.csv')
        name = 'Jumanji'
        film = df.loc[df['original_title'] == name]
        self.assertTrue(film['original_title'][1])
        self.assertEqual(film['original_title'][1], name)

    def test_search_film_not_founded(self):
        df = pd.read_csv('C:\movies\core\movies_metadata.csv')
        name = 'Jumanjiheqwohd'
        film = df.loc[df['original_title'] == name]
        self.assertTrue(film.empty)

```

Результат виконання тестів

```
C:\movies>python manage.py test tests
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...C:\Users\Alisa\AppData\Local\Programs\Python\Python38\lib\unittest\
on import or set low_memory=False.
return self.run(*args, **kwargs)
.....
-----
Ran 9 tests in 1.861s
OK
Destroying test database for alias 'default'...
C:\movies>
```

UPD

Приклад TDD:

Спочатку був розроблен тест (перевірка роботи умови, що назва шуканого фільму має бути <20 символів):

```
def test_search_form_max_length(self):
    form_data = {'search': 'Jumanji'}
    form = SearchForm(data=form_data)
    self.assertTrue(form.is_valid())

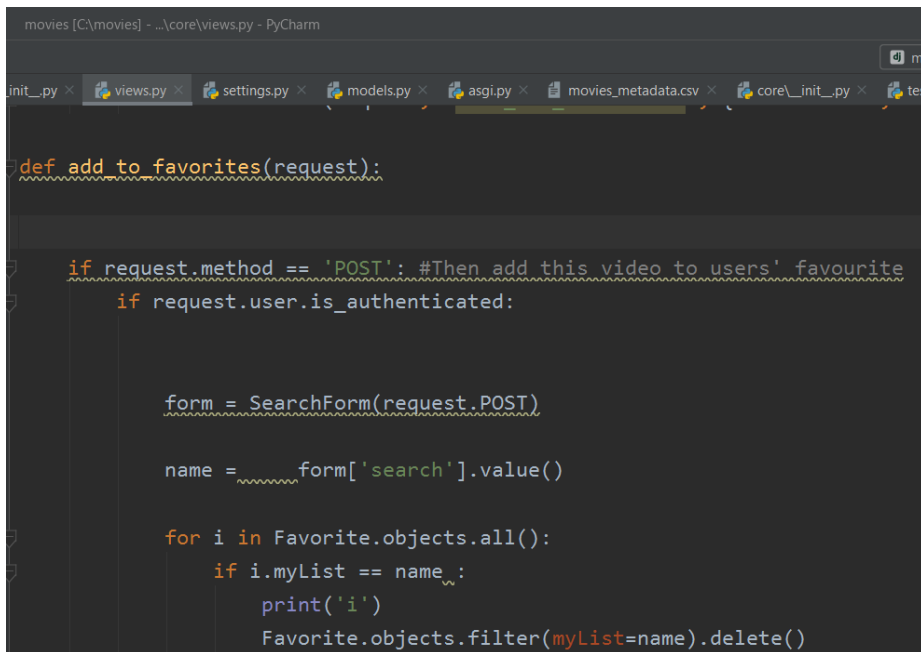
    form_data = {'search': 'JumanjiJumanjiJumanjiJumanjiJumanjiJumanji'}
    form = SearchForm(data=form_data)
    self.assertFalse(form.is_valid())
```

Вже після нього код:

```
class SearchForm(forms.Form):
    search = forms.CharField(widget=forms.TextInput(attrs={'class': "form-control", 'placeholder': 'Search',}), label='', max_length=20)
```

Рефакторинг

Код до рефакторингу



```

def add_to_favorites(request):

    if request.method == 'POST': #Then add this video to users' favourite
        if request.user.is_authenticated:

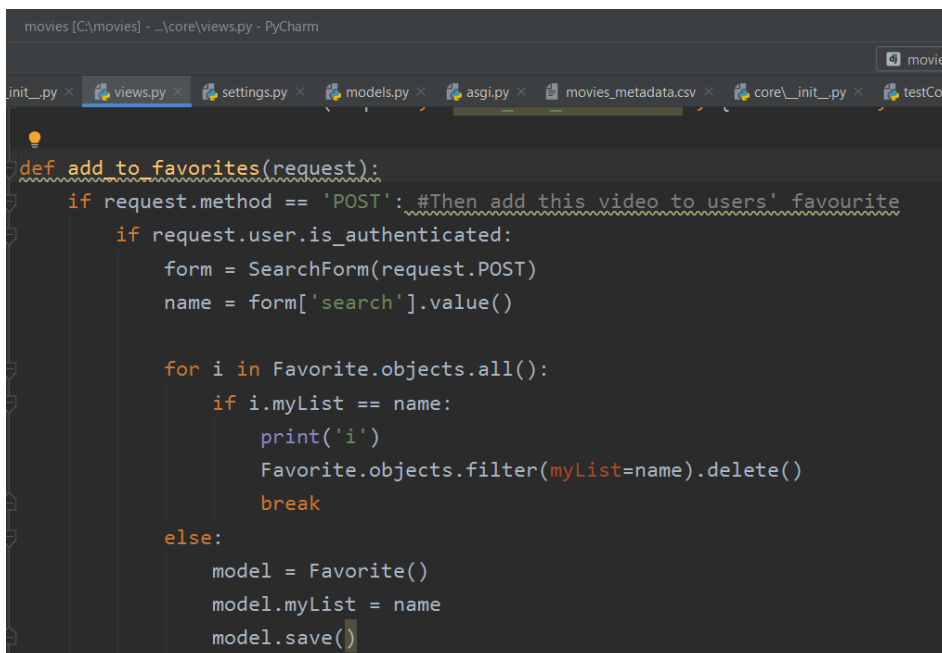
            form = SearchForm(request.POST)

            name = form['search'].value()

            for i in Favorite.objects.all():
                if i.myList == name:
                    print('i')
                    Favorite.objects.filter(myList=name).delete()

```

Код після рефакторингу



```

def add_to_favorites(request):
    if request.method == 'POST': #Then add this video to users' favourite
        if request.user.is_authenticated:
            form = SearchForm(request.POST)
            name = form['search'].value()

            for i in Favorite.objects.all():
                if i.myList == name:
                    print('i')
                    Favorite.objects.filter(myList=name).delete()
                    break
            else:
                model = Favorite()
                model.myList = name
                model.save()

```

Висновок

У ході виконання лабораторної роботи було розглянуто створення програмного забезпечення за методологією TDD та ознайомлено з процедурам рефакторингу.

Для рефакторингу коду було застосовано вбудований у PyCharm інструмент для рефакторингу. Для юніт тестів було застосовано вбудований

модуль - Django.test. Процедура тестування програмного забезпечення є важливою та у певній мірі важкою задачею, бо вона складається з декількох логічних слоїв: від HTTP-запитів, тестування моделей, валідації форм та їх обробки до рендерингу шаблонів сторінок програми.

Усі 9 тестів виконались успішно за 1.861 секунди. Тести були написані до розробки коду, тому такий підхід можна назвати – TDD (Test Driven Development) методологією.