

**КПІ ім. Ігоря Сікорського**  
**Інститут прикладного системного аналізу**  
**Кафедра Системного проектування**

**Лабораторна робота № 5**

**«Модульне тестування (Unit-тести) та рефакторинг.»**

**з дисципліни «Проектування інформаційних систем»**

**Виконав:**

**Студент групи ДА-72**

**ННК «ІПСА»**

**Д'яконов Д.К.**

**Варіант № 5**

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

### Задача:

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 роботи:

Для рефакторингу коду не протязі всього циклу розробки були застосовані автоматичні форматери коду, що не тільки тримають код на високому рівні читаємості, але й дотримується загальноприйнятих стандартів.

- Black для Python, що дотримується стандарту PEP 8
- Prettier для Typescript

Для юніт-тестів був використаний вбудований в Django Rest Framework модуль для написання юніт-тестів.

1. Тест коректного створення юзера у базі даних
2. Тест коректної реєстрації за допомогою API
3. Тест помилкової реєстрації за допомогою API
4. Тест коректного логіну у додаток за допомогою API
5. Тест помилкового логіну за допомогою API
6. Тест коректності підтягування даних у додаток з відкритого API Rozklad
7. Тест коректної генерації титульної сторінки за допомогою API

В результаті маємо:

```
(env) PS C:\Users\ddk20\source\repos\lab-templates-back> python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....
-----
Ran 7 tests in 2.139s

OK
Destroying test database for alias 'default'...
```

Тобто, додаток успішно пройшов всі 7 тестів трохи більше ніж за 2 секунди.

Оскільки всі тести були написані після розробки функціоналу, такий підхід не можна назвати прикладом ТДД. Він був би таким, якщо б наприклад тести для АПІ логіну були б написані до самої логіки, тобто під час розробки замість покрокового тестування вручну можна б було скористатися наперед написаними юніт тестами.

**Висновок:** В ході даної лабораторної роботи було написано 7 юніт тестів для перевірки логіки роботи додатку, які були пройдені за 2.139 секунди, також були встановлені системи для автоматичного форматування коду, що позбавляють від необхідності рефакторінгу, зберігаючи код в чистоті та в одному стилі на протязі всієї розробки.

**Додаток:**

Пройдемо шлях з прикладом TDD

Опишемо початковий тест

```
def test_lessons_search(self):
    url = reverse("lessons") # Беремо URL
    user = User.objects.filter(email="test@test.com")[0] # Беремо юзера

    self.client.login(email="test@test.com", password="reAllySafePas5w0rd")
    for i in range(5): # Створюємо предмети
        lesson = Lesson(name=f"test {i}")
        lesson.save()
        lesson.students.add(user)
    res1 = self.client.get(
        url + f"?search=test 2",
    )
    self.assertEqual(res1.status_code, 200) # Перевіряємо статус
    self.assertEqual(len(json.loads(res1.content)), 1) # Перевіряємо результат
```

Перевіримо результат

```
=====
FAIL: test_lessons_search (generator.tests.GeneratorTestCase)
-----
Traceback (most recent call last):
  File "C:\Users\ddk20\source\repos\lab-templates-back\generator\tests.py", line 70, in test_lessons_search
    self.assertEqual(len(json.loads(res1.content)), 1)
AssertionError: 5 != 1
```

Реалізуємо необхідний функціонал та перевіримо знову

```
-----
Ran 6 tests in 1.336s
```

```
OK
```

## Код для юніт тестів

```
from rest_framework.test import APITestCase
from rest_framework.reverse import reverse
from rest_framework import status
from .models import User

class UserTestCase(APITestCase):
    def setUp(self):
        user = User.objects.create(
            email="test@test.com",
            name="Test U.S.",
            group_id=5156,
            group_name="ДА-72",
            gender="m",
            variant=5,
        )
        user.set_password("reAllySafePas5w0rd")
        user.save()

    def test_created_user(self):
        qs = User.objects.filter(email="test@test.com")
        self.assertEqual(qs.count(), 1)

    def test_register_user_api(self):
        url = reverse("register")
        data = {
            "email": "test_api@test.com",
            "name": "Test A.P.I.",
            "group_id": 5156,
            "group_name": "ДА-72",
            "gender": "m",
            "variant": 8,
            "password": "reAllySafePas5w0rd",
            "password2": "reAllySafePas5w0rd",
        }
        response = self.client.post(url, data, format="json")
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        qs = User.objects.filter(id=response.data["id"])
        self.assertEqual(qs.count(), 1)

    def test_register_user_api_fail(self):
        url = reverse("register")
        data = {
            "email": "test_api@test.com",
            "name": "Test A.P.I.",
            "password": "reAllySafePas5w0rd",
        }
        response = self.client.post(url, data, format="json")
        self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
```

```

def test_login_user_api(self):
    url = reverse("login")
    data = {
        "username": "test@test.com",
        "password": "reAllySafePas5w0rd",
    }
    response = self.client.post(url, data, format="json")
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertGreater(len(response.data.get("token")), 0)

def test_login_user_api_fail(self):
    url = reverse("login")
    data = {
        "username": "test@test.com",
        "password": "qwerty123",
    }
    response = self.client.post(url, data, format="json")
    self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST)
    self.assertEqual(response.data["non_field_errors"][0].code, "authorization"
)

```

```

from rest_framework.test import APITestCase
from rest_framework.reverse import reverse
from rest_framework import status
from accounts.models import User
from rest_framework.authtoken.models import Token
from .utils import get_lessons
from .models import Lesson
import requests

```

```

class GeneratorTestCase(APITestCase):
    def setUp(self):
        user = User.objects.create(
            email="test@test.com",
            name="Test U.S.",
            group_id=5156,
            group_name="ДА-72",
            gender="m",
            variant=5,
        )
        user.set_password("reAllySafePas5w0rd")
        user.save()

    def test_lessons_from_rozklad(self):
        user = User.objects.filter(email="test@test.com")[0]

```

```

        res = requests.get(
            f"https://api.rozklad.org.ua/v2/groups/{user.group_id}/lessons"
        )
        self.assertEqual(res.status_code, requests.codes.ok)
        number_from_api = len({obj["lesson_full_name"] for obj in res.json()["data"]
    })

    number_of_objects = len(get_lessons(user))
    self.assertEqual(number_from_api, number_of_objects)
    self.assertEqual(number_from_api, user.lessons.count())

    def test_generator(self):
        user = User.objects.filter(email="test@test.com")[0]
        token, created = Token.objects.get_or_create(user=user)
        url1 = reverse("lessons")
        response1 = self.client.post(
            url1,
            {"name": "Testing API"},
            format="json",
            HTTP_AUTHORIZATION=f"Token {token}",
        )
        self.assertEqual(response1.status_code, status.HTTP_201_CREATED)
        qs = Lesson.objects.filter(id=response1.data["id"])
        self.assertEqual(qs.count(), 1)
        lesson = qs[0]
        url2 = reverse("generator", args=[lesson.pk])
        response2 = self.client.get(url2 + f"?token={token}&lab_name=test&number=6")

        self.assertEqual(response2.status_code, 200)
        self.assertEqual(
            response2["Content-Disposition"], 'attachment; filename="lab-6.docx"'
        )

```