Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра Програмних засобів

**ЗВІТ**

з лабораторної роботи № 4

з дисципліни «Проектування сучасних інформаційних систем»

з теми: **«**Принципи тестування у Django**»**

Виконала:

студентка групи КНТ-219м А. В. Кущ

Прийняв:

к.т.н., доцент Г.В. Табунщик

2019

**1. Мета роботи**

Ознайомитися із принципами тестування у Django, навчитися створювати власні тести та аналізувати результати.

**2. Завдання до роботи**

1. Застосувавши конспект лекцій та додаткову літературу, вивчити принципи тестування у Django.
2. Протестувати роботу веб-застосування та виявити чи має застосування порушення у функціональності або містить помилки, а також виявити та виправити помилки, якщо вони були допущені при написанні тестів.
3. Оформити звіт. Звіт з лабораторної роботи повинен містити: мету роботи; завдання на виконання роботи; тексти програм; результати роботи; висновки.

**3. Результати роботи**

**Код test.py**

from django.urls import reverse, reverse\_lazy

from django.test import TestCase

from django.http import HttpResponseNotAllowed

from .views import generate\_lights\_data

from .models import Article

class TestCalls(TestCase):

def setUp(self):

self.create\_url = 'create'

self.list\_view = 'main'

def test\_generating\_data(self):

ligths\_test\_value = 400

response = generate\_lights\_data(

f'2019-11-13T12:55:53.612598 {ligths\_test\_value}')

expected\_response = {

'lights': [1, 1, 1, 1, 0, 0, 0, 0],

'lightning\_value': ligths\_test\_value

}

self.assertDictEqual(response, expected\_response)

def test\_call\_view\_loads(self):

for url in [self.create\_url, self.list\_view]:

path = str(reverse\_lazy(url)) #генерація повного шляху

response = self.client.get(path)# імітація відправки запиту на отримання

self.assertEqual(response.status\_code, 200)

self.assertTemplateUsed(response, f"blog/{url}.html")

def test\_call\_view\_fails\_blank(self):

path = str(reverse\_lazy(self.create\_url))

response = self.client.post(path, {})

self.assertFormError(

response,

'form',

'title',

['This field is required.']

)

def test\_call\_view\_fails\_incorrect(self):

path = str(reverse(self.create\_url))

value = '1' \* 65

data = {

'title': value,

'description': 'test description'

}

response = self.client.post(path, data, follow=True)

self.assertFormError(

response,

'form',

'title',

['Ensure this value has at most 64 characters (it has 65).']

)

def test\_call\_view\_succeed(self):

path = str(reverse\_lazy(self.create\_url))

value = '1'\*10

description = 'test description'

article\_data = {

'title': value,

'description': description

}

response = self.client.post(path, article\_data)

self.assertEqual(response.status\_code, 302)

article = Article.objects.get(id=1)

for field in article\_data.keys():

article\_field = getattr(article, field)

self.assertEqual(article\_field, article\_data[field])

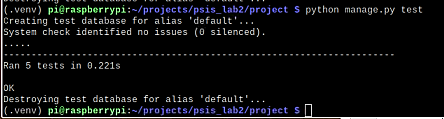


Рисунок 1 – Результат виконання тестів

**4. Висновок**

На цій лабораторній роботі я ознайомилась із принципами тестування у Django, навчилась створювати власні тести та аналізувати результати.