

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

Лабораторна робота №6
з дисципліни

«Проектування інформаційних систем»

Система автоматичного створення довідника користувача та
оформлення коду за допомогою Coding Convention

Виконала:
студентка групи ДА-72
Потапова С. С.
Варіант 24

Київ – 2020

Мета роботи: за допомогою системи генерації довідника користувача створити документ у форматі PDF і HTML для архітектурної програмної моделі.

Задача:

1. Вивчити теги системи генерації керівництва користувача.
2. Створити опис для всіх класів API з описом призначення кожного класу, методів класу і членів класу.
3. Згенерувати документацію у форматах PDF, HTML.

Завдання:

1. Для кожного з класів API в код програми додати теги з описом керівництва користувача для архітектурної програмної моделі.
2. Обрати Coding Convention. Оформити код програми згідно Coding Convention.
2. Встановити налаштування системи автоматичного створення керівництва користувача.
3. Згенерувати HTML документацію керівництва користувача.
4. Згенерувати PDF документацію керівництва користувача.

Хід роботи

1 Теги автоматичного створення документації

Для автоматичного створення документації в Python може використовуватись стандартна бібліотека Sphinx, що має стандартні теги, такі як `param`, `type`, `return`, які можуть оформлюватись, як зазначено нижче.

```
def start(self, ampChild=None):  
    """Starts the ProcessPool with a given child protocol.  
  
    :param ampChild: a :class:`ampoule.child.AMPChild` subclass.  
    :type ampChild: :class:`ampoule.child.AMPChild` subclass  
    """
```

“”” ... “”” відповідають за позначення коментарів як раз для оформлення документації, в той час як відомо, що звичайний строковий коментар позначається як #.

2 Автоматичне створення документації

Існує безліч способів генерації документації. Використаний для прикладу у даній лабораторній роботі це онлайн ресурс створення <https://app.swaggerhub.com/>

SwaggerHub надає повне рішення для проектування, управління та публікації документації API способами, які спрощують життя технічного письменника API.

| | | | |
|------------|--------|--|---|
| admins | | Secured Admin-only calls | ▼ |
| POST | /users | adds the user | ↶ |
| developers | | Operations available to regular developers | ▼ |
| GET | /users | searches users | ↶ |
| tasks | | | ▼ |
| GET | /tasks | searches tasks | ↶ |
| POST | /tasks | adds the task | ↶ |
| DELETE | /tasks | deletes the task | ↶ |
| PUT | /tasks | updates the task | ↶ |

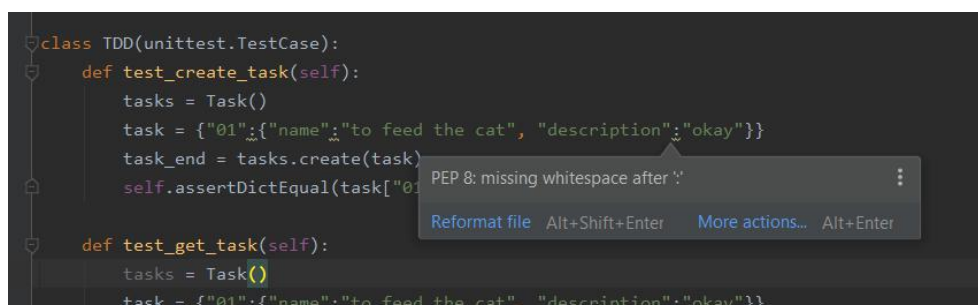
Процес створення відображеного вище.

```
161 /users:
162   get:
163     tags:
164       - developers
165     summary: searches users
166     operationId: searchUsers1
167     description: |
168       By passing in the appropriate options, you can search for
169       available user in the system
170     produces:
171       - application/json
172     parameters:
173       - in: query
174         name: searchString
175         description: pass an optional search string for looking up for
176           user
177         required: false
178         type: string
179       - in: query
180         name: skip
181         description: number of records to skip for pagination
182         type: integer
183         format: int32
```

3 Обрати Coding Convention. Оформити код програми згідно Coding Convention

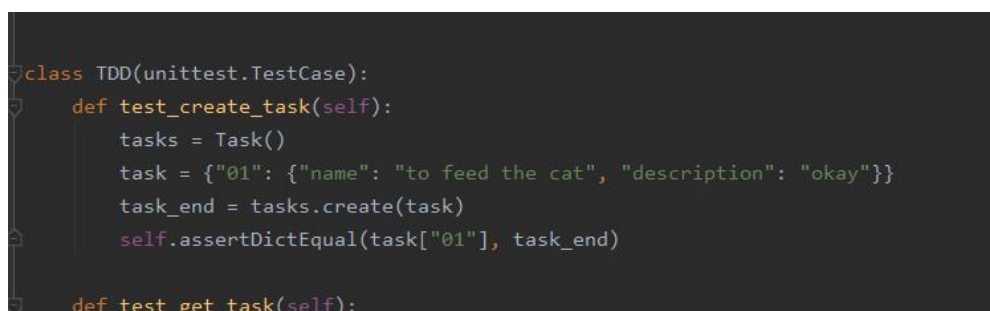
PEP8 можна визначити, як документ, що описує загальноприйнятий стиль написання коду на мові Python. Python Enhanced Proposal (PEP) - перекладається, як заявки щодо поліпшення мови Python.

Нижче можна побачити типічний приклад автоматичного використання PEP8.



```
class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
        task_end = tasks.create(task)
        self.assertEqual(task["01"], task_end)

    def test_get_task(self):
        tasks = Task()
        task = {"01":{"name":"to feed the cat", "description":"okay"}}
```



```
class TDD(unittest.TestCase):
    def test_create_task(self):
        tasks = Task()
        task = {"01": {"name": "to feed the cat", "description": "okay"}}
        task_end = tasks.create(task)
        self.assertEqual(task["01"], task_end)

    def test_get_task(self):
```

PEP 8 створений на основі рекомендацій Гуїдо ван Россум з додатками від Баррі. І, звичайно, цей PEP може бути неповним (фактично, його, напевно, ніколи не буде закінчено).

Ключова ідея Гуїдо така: код читається набагато більше разів, ніж пишеться. Власне, рекомендації про стиль написання коду спрямовані на те, щоб поліпшити читаність коду і зробити його узгодженим між великим числом проєктів. В ідеалі, весь код буде написаний в єдиному стилі, і будь-хто зможе легко його прочитати.

Висновки

У результаті виконання лабораторної роботи було розібрано основні теги генерації документації, розібрано Coding Convention для обраної мови програмування, а саме PEP8 для Python, створено документацію API проекту , за допомогою онлайн ресурсу Swagger, було приведено приклади до кожного пункту.