API

Що таке API

Додавши до свого додатка один із багатьох відкритих API, ви можете розширити функціональність цього додатка або доповнити його необхідними даними.

Відповідь проста: потрібно створити своє власне API.

Незважаючи на те, що це на початку може здатися складним завданням, насправді все просто. Ми розповімо, як це зробити за допомогою Python.

Що потрібно для початку роботи

Для розробки API потрібно:

- Python 3;
- Flask — простий і легкий у використанні фреймворк для створення веб-додатків;
- Flask-RESTful — розширення для Flask, яке дозволяє розробити REST API швидко і з мінімальною настройкою.

Встановлення здійснюється командою:

```
pip install flask-restful
```

Перед тим, як почати

Ми збираємося розробити RESTful API з базовою CRUID-функціональністю.

Щоб повністю зрозуміти завдання, давайте розберемося з двома термінами, згаданими вище.

Що таке REST?

REST API (Representational State Transfer) — це API, яке використовує HTTP-запити для обміну даними.

REST API повинні відповідати певним критеріям:

Архітектура клієнт-сервер: клієнт взаємодіє з користувацьким інтерфейсом, а сервер — з бекендом і сховищем даних. Клієнт і сервер незалежні, будь-який з них може бути замінений окремо від іншого.
Stateless — жодні клієнтські дані не зберігаються на сервері. Стан сеансу зберігається на стороні клієнта.
Кешування — клієнти можуть кешувати відповіді сервера для покращення загальної продуктивності.

Що таке CRUD?

CRUD — концепція програмування, яка описує чотири базові дії (create, read, update і delete).

У REST API типи запитів і методи запиту відповідають за такі дії, як post, get, put, delete.

Тепер, коли ми розібралися з базовими термінами, можна приступити до створення API.

Розробка

Давайте створимо репозиторій цитат про штучний інтелект. ШІ — одна з найбільш активно розвиваючихся технологій сьогодні, а Python — популярний інструмент для роботи з ШІ.

З цим API розробник Python зможе швидко отримувати інформацію про ШІ та надихатися новими досягненнями. Якщо у розробника є цінні думки з цієї теми, він зможе додавати їх до репозиторію.

Почнемо з імпорту необхідних модулів та налаштування Flask:

```
from flask import Flask
from flask_restful import Api, Resource, reqparse
import random
app = Flask(__name__)
api = Api(app)
```

У цьому фрагменті Flask, Api і Resource — класи, які нам потрібні.

Reqparse — це інтерфейс парсингу запитів Flask-RESTful... Також знадобиться модуль random для ві

дображення випадкової цитати.

Тепер ми створимо репозиторій цитат про ШІ.

Кожен запис репо буде містити:

цифровий ID;
ім'я автора цитати;
цитату.

Оскільки це лише приклад для навчання, ми збережемо всі записи у списку Python. У реальному застосуванні ми, ймовірно, використовували б базу даних.

```
ai_quotes = [
    {
        "id": 0,
        "author": "Кевін Келлі",
        "quote": "Бізнес-плани наступних 10 000 стартапів л
егко передбачити: " +
                 "Візьміть X та додайте ШІ (Штучний Інтелек
т)."
    },
    {
        "id": 1,
        "author": "Стівен Хокінг",
        "quote": "Розвиток повноцінного штучного інтелекту
може " +
                 "означити кінець людської раси… " +
                 "Він вийде поза контроль та перепроектує с
ебе " +
                 "з зростанням швидкості. " +
                 "Люди, обмежені повільною біологічною евол
юцією, " +
                 "не зможуть конкурувати та будуть витіснен
і."
    },
    {
        "id": 2,
        "author": "Клод Шеннон",
        "quote": "Я уявляю час, коли ми будемо для роботів
тим, " +
                 "що собаки для людей, " +
                 "і я вболіваю за машини."
```

```
    },
    {
        "id": 3,
        "author": "Ілон Маск",
        "quote": "Темпи прогресу в галузі штучного інтелект
у " +
                 "(я не маю на увазі вузький ШІ) " +
                 "неймовірно великі. Якщо у вас немає прямо
го " +
                 "знайомства з групами, такими як Deepmind,
" +
                 "ви не уявляєте, наскільки швидко це росте
" +
                 "з темпом, що близький до експоненційного.
" +
                 "Ризик серйозних небезпек надзвичайно вели
кий " +
                 "і припадає на п'ятирічний період." +
                 "Максимум 10 років."
    },
    {
        "id": 4,
        "author": "Джеффрі Гінтон",
        "quote": "Я завжди був переконаний, що єдиний спосі
б " +
                 "зробити штучний інтелект працюючим, " +
                 "це виконувати обчислення подібно до мозку
людини. " +
                 "Це ціль, яку я переслідую. Ми робимо прогр
ес, " +
                 "хоча нам ще багато вивчити про те, " +
                 "як насправді працює мозок."
    },
    {
        "id": 5,
        "author": "Педро Домінгос",
        "quote": "Люди стурбовані тим, що комп'ютери " +
                 "стануть занадто розумними та взяти у св
```

```
ої руки світ, " +
                "але справжня проблема в тому, " +
                "що вони надто дурні і вже взяли у свої ру
ки світ."
    },
    {
        "id": 6,
        "author": "Алан Тьюрінг",
        "quote": "Здається ймовірним, що коли машинне мисле
ння " +
                "метод розпочнеться, не пройде довго, " +
                "поки воно не випередить наші слабкі здібн
ості… " +
                "Вони зможуть спілкуватися між собою, " +
                "щоб підточувати свою гостроту. " +
                "Отже, на деякому етапі ми повинні були б
" +
                "очікувати, що машини візьмуть під контрол
ь."
    },
    {
        "id": 7,
        "author": "Рей Курцвейл",
        "quote": "Штучний інтелект досягне " +
                "рівня людського до 2029 року. " +
                "Далі виведіть це ще на, скажімо, 2045 рі
к, " +
                "і ми помножимо інтелект, " +
                "людський біологічний машинний інтелект "
+
                "нашої цивілізації мільярди разів."
    },
    {
        "id": 8,
        "author": "Себастьян Трун",
        "quote": "Ніхто не формулює це так, але я вважаю, "
+
                "що штучний інтелект майже гуманітарний пр
```

```
едмет. " +
                "Це насправді спроба зрозуміти людський ін
телект та когніцію."
    },
    {
        "id": 9,
        "author": "Ендрю Нг",
        "quote": "Ми робимо таку аналогію, що ШІ - це нова
електроенергія." +
                "Електроенергія трансформувала галузі: сіл
ьське господарство, "
+
                "транспорт, зв'язок, виробництво."
    }
]
```

Тепер потрібно створити клас ресурсів Quote, який буде визначати операції ендпоінтів нашого API. Усередині класу слід заявити чотири методи: get, post, put, delete.

Почнемо з методу GET

Він надає можливість отримати конкретну цитату, вказавши її ID, або ж випадкову цитату, якщо ID не вказано.

```
class Quote(Resource):
    def get(self, id=0):
        if id == 0:
            return random.choice(ai_quotes), 200
        for quote in ai_quotes:
            if(quote["id"] == id):
                return quote, 200
        return "Цитату не знайдено", 404
```

Метод GET повертає випадкову цитату, якщо ID містить значення за замовчуванням, тобто при виклику методу ID не було задано.

Якщо він заданий, метод шукає серед цитат ту, яка містить вказане ID. Якщо ж нічого не знайдено, виводиться повідомлення "Цитату не знайдено", 404.

Пам'ятайте: метод повертає HTTP-статус 200 в разі успішного запиту та 404, якщо запис не знайдено.

Тепер давайте створимо метод POST

для додавання нової цитати до репозиторію

Він буде отримувати ідентифікатор кожної нової цитати при введенні. Крім того, POST буде використовувати reqparse для парсингу параметрів, які будуть йти в тілі запиту (автор і текст цитати).

```
def post(self, id):
      parser = reqparse.RequestParser()
      parser.add_argument("author")
      parser.add_argument("quote")
      params = parser.parse_args()
      for quote in ai_quotes:
          if(id == quote["id"]):
              return f"Цитата з ID {id} вже існує", 400
      quote = {
          "id": int(id),
          "author": params["author"],
          "quote": params["quote"]
      }
      ai_quotes.append(quote)
      return quote, 201
```

У вищенаведеному коді POST-метод прийняв ID цитати. Потім, використовуючи reqparse, він отримав автора та цитату з запиту, зберігаючи їх в словнику params.

Якщо цитата з вказаним ID вже існує, метод виводить відповідне повідомлення та код 400.

Якщо цитата з вказаним ID ще не була створена, метод створює новий запис з вказаним ID, автором та іншими параметрами. Потім він додає запис до списку ai_quotes і повертає запис з новою цитатою разом з кодом 201.

PUT

```
def put(self, id):
      parser = reqparse.RequestParser()
      parser.add_argument("author")
      parser.add_argument("quote")
      params = parser.parse_args()
      for quote in ai_quotes:
          if id == quote["id"]:
              quote["author"] = params["author"]
              quote["quote"] = params["quote"]
              return quote, 200

      quote = {
          "id": id,
          "author": params["author"],
          "quote": params["quote"]
      }

      ai_quotes.append(quote)
      return quote, 201
```

PUT-метод, подібно до попереднього прикладу, приймає ID та введення, і розбирає параметри цитати за допомогою reqparse.

Якщо цитата з вказаним ID існує, метод оновить її новими параметрами, а потім виведе оновлену цитату з кодом 200. Якщо цитати з вказаним ID ще немає, буде створено новий запис із кодом 201.

Наостанок, створимо **DELETE**-метод для видалення цитати, яка вже не надихає.

```
def delete(self, id):
      global ai_quotes
      ai_quotes = [quote for quote in ai_quotes if quote["i
d"] != id]
      return f"Цитату з ID {id} видалено.", 200
```

Цей метод отримує ID цитати при введенні і оновлює список `ai_quotes`, використовуючи глобальний список.

Тепер, коли ми створили всі методи, все, що нам залишається, - це просто додати ресурс до API, вказати шлях і запустити Flask.

```
api.add_resource(Quote, "/ai-quotes", "/ai-quotes/", "/ai-q
uotes/<int:id>")
if __name__ == '__main__':
    app.run(debug=True)
```

Наш REST API Service готовий!

Далі ми можемо зберегти код у файл `app.py`, запустивши його в консолі за допомогою команди:

```
python3 app.py
```

Якщо все добре, ми отримаємо щось подібне до цього:

```
* Debug mode: on
* Running on 127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: XXXXXXX
```

Тестируємо API

Після створення API його потрібно протестувати.

Це можна зробити за допомогою консольного інструмента curl або клієнта Insomnia REST, або ж опублікувавши API на Rapid API