Классификация сообщества

Scratch — "Сообщество кода": по свойствам — образовательное, детское, коллаборативное (студии как группы проектов). Концепт: группы по темам (игры, анимация), метрикам (активность API).

В рамках проекта был разработан инструмент на Python для взаимодействия с открытым API социальной сети Scratch — платформы образовательного программирования для детей и подростков. Это позволило собрать данные о конкретном участнике (профиль, статистика активности) и студии (групповое сообщество), преобразовать JSON в визуальные диаграммы PlantUML, а также провести количественный анализ метрик влияния (followers, views, проекты).

Польза работы

Анализ данных помогает классифицировать сообщества по активности, выявлять лидеров и динамику коллабораций, что полезно для образовательных исследований.

Данные из Scratch API дают insights в творческие процессы детей: высокий views/loves указывает на образовательную ценность проектов, способствуя развитию логического мышления и командной работы в сообществе. В итоге проект усиливает цифровую дидактику, предоставляя инструменты для рефлексии и анализа сетевых взаимодействий без ручного сбора.

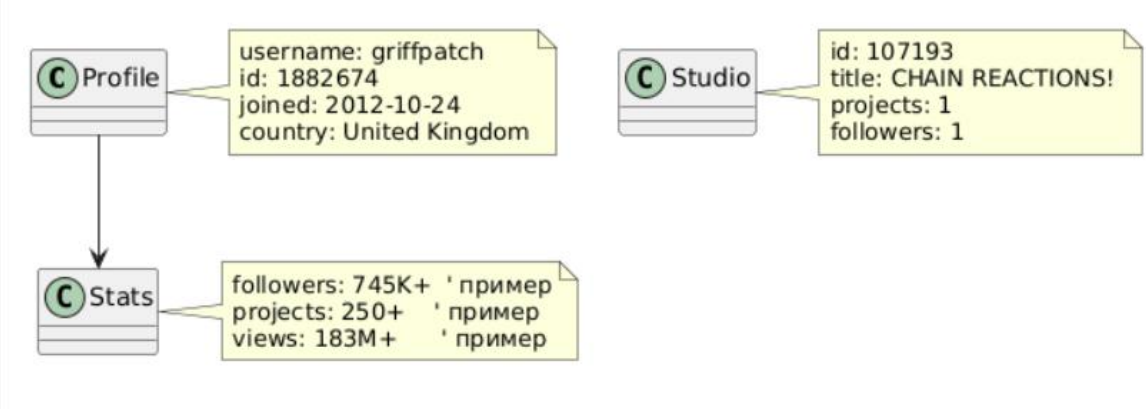
Результат работы программы:


```
== Запрос к Scratch API ==

== Данные пользователя (raw JSON) ==
{
  "id": 1882674,
  "username": "griffpatch",
  "scratchteam": false,
  "history": {
    "joined": "2012-10-24T12:59:31.000Z"
  },
  "profile": {
    "id": 1267661,
    "images": {
      "90x90": "https://cdn2.scratch.mit.edu/get_image/user/1882674_90x90.png?v=",
      "60x60": "https://cdn2.scratch.mit.edu/get_image/user/1882674_60x60.png?v=",
      "55x55": "https://cdn2.scratch.mit.edu/get_image/user/1882674_55x55.png?v=",
      "50x50": "https://cdn2.scratch.mit.edu/get_image/user/1882674_50x50.png?v=",
      "32x32": "https://cdn2.scratch.mit.edu/get_image/user/1882674_32x32.png?v="
    },
    "status": "YouTube Tutorials ▶  www.youtube.com/griffpatch\nMe: @griffpatch_tutor | @Griffpatch-Academy\nPlease don't spam: Max 1 ad per user a day",
    "bio": "Got hooked on coding when I was a kid, now I'm a parent and nothing's changed! My day job involves java coding. In my spare time I love making g
ames, being creative & drumming in church.",
    "country": "United Kingdom",
    "membership_avatar_badge": 0,
    "membership_label": 1
  },
  "join_date": "2012-10-24",
  "country": "United Kingdom"
}

== Данные студии (raw JSON) ==
{
  "id": 107193,
  "title": "CHAIN REACTIONS!",
  "host": 447016,
  "description": "Add your own chain reactions and put them here.",
  "visibility": "visible",
  "public": true,
  "open_to_all": false,
```

Сгенерированная Диаграмма классов:



Приложение:

Исходный код программы

```
import requests
import json

# -------- 1. ЗАПРОСЫ К SCRATCH API --------

def get_user_data(username: str) -> dict | None:
    """Получает данные пользователя Scratch (открытое API)."""
    url = f"https://api.scratch.mit.edu/users/{username}/"
    try:
        r = requests.get(url, timeout=10)
        if r.status_code == 200:
            data = r.json()
            # чуть нормализуем под себя
            data["join_date"] = data.get("history", {}).get("joined", "")[:10]
            data["country"] = data.get("profile", {}).get("country", "")
            return data
        print("API error:", r.status_code)
        return None
    except Exception as e:
        print("Request error:", e)
        return None
```

```python
def get_studio_data(studio_id: int) -> dict | None:
    """Получает данные студии Scratch."""
    url = f"https://api.scratch.mit.edu/studios/{studio_id}/"
    try:
        r = requests.get(url, timeout=10)
        if r.status_code == 200:
            return r.json()
        print("API error:", r.status_code)
        return None
    except Exception as e:
        print("Request error:", e)
        return None



# -------- 2. PLANTUML ДЛЯ ПОЛЬЗОВАТЕЛЯ --------

def generate_plantuml_user(user: dict) -> str:
    """
    Генерирует простой PlantUML под твой движок:
    только объявления class + note, без { } внутри.
    """
    username = user.get("username", "N/A")
    uid = user.get("id", "N/A")
    join_date = user.get("join_date", "N/A")
    country = user.get("country", "N/A")

    diagram = "@startuml\n"
    diagram += "class Profile\n"
    diagram += "class Stats\n"
    diagram += "Profile --> Stats\n"
    diagram += "note right of Profile\n"
    diagram += f"  username: {username}\n"
    diagram += f"  id: {uid}\n"
    diagram += f"  joined: {join_date}\n"
    diagram += f"  country: {country}\n"
    diagram += "end note\n"
    diagram += "note right of Stats\n"
    diagram += "  followers: 745K+  ' пример\n"
    diagram += "  projects: 250+    ' пример\n"
    diagram += "  views: 183M+      ' пример\n"
    diagram += "end note\n"
    diagram += "@enduml\n"
    return diagram



# -------- 3. PLANTUML ДЛЯ СТУДИИ --------
```

```python
def generate_plantuml_studio(studio: dict) -> str:
    """Простая диаграмма для студии."""
    sid = studio.get("id", "N/A")
    title = studio.get("title", "N/A")
    projects = studio.get("stats", {}).get("projects", 0)
    followers = studio.get("stats", {}).get("followers", 0)

    diagram = "@startuml\n"
    diagram += "class Studio\n"
    diagram += "note right of Studio\n"
    diagram += f"  id: {sid}\n"
    diagram += f"  title: {title}\n"
    diagram += f"  projects: {projects}\n"
    diagram += f"  followers: {followers}\n"
    diagram += "end note\n"
    diagram += "@enduml\n"
    return diagram


# -------- 4. MAIN --------

if __name__ == "__main__":
    username = "griffpatch"
    studio_id = 107193

    print("== Запрос к Scratch API ==")
    user = get_user_data(username)
    studio = get_studio_data(studio_id)

    print("\n== Данные пользователя (raw JSON) ==")
    print(json.dumps(user, indent=2, ensure_ascii=False))

    print("\n== Данные студии (raw JSON) ==")
    print(json.dumps(studio, indent=2, ensure_ascii=False))

    # генерим PlantUML
    user_puml = generate_plantuml_user(user)
    studio_puml = generate_plantuml_studio(studio)

    # сохраняем в файлы
    with open("scratch_user_plantuml.txt", "w", encoding="utf-8") as f:
        f.write(user_puml)

    with open("scratch_studio_plantuml.txt", "w", encoding="utf-8") as f:
```

```
        f.write(studio_puml)

    print("\n== PlantUML для пользователя ==")
    print(user_puml)

    print("\n== PlantUML для студии ==")
    print(studio_puml)

    print("\nСкопируй содержимое *.txt в PlantUML / вики и получишь
диаграммы.")
```