Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2022-2

17 de Noviembre de 2022
Actividad Bonus

Actividad Bonus

DCCoordinando IIC2233

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AB/`
- **Hora del *push*:** Viernes 18 de noviembre 20:00

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

¡Has sido seleccionad@ como el nuev@ coordinador de programación avanzada! Dentro de tu nuevas labores se encuentra recolectar las entregas, filtrar los *commit* realizados después de la fecha de entrega y publicar las *issues* con el *commit* a revisar de esta misma actividad. Sin embargo, hay un pequeño problema: aún no sabes cómo hacerlo.

Es en ese momento que recuerdas tus conocimientos aprendidos de *Webservices* y *Regex*, por lo que decides intentar usar la API de Github para publicar las *issues*, pero no sin primero hacer una pequeña prueba de que tu código funciona en tu repositorio personal del curso, usando como fuente de datos una API de Anime que ~~te impuso el profesor~~ encontraste. ¡El futuro de IIC2233 está en tus manos, programador avanzad@!

Flujo del programa

El programa deberá inicialmente interactuar con Anime Ranking, una API que te entregará información sobre animes. Una vez recibidos los datos, se filtrarán los datos en función de una expresión regular.

Luego, haciendo uso de la API de Github deberás completar el proceso de manipulación de *issues*: crear una *issue*, bloquear la *issue* creada[1] y desbloquear una *issue*. Para utilizar la API de Github, el programa solicitará demostrar que eres el dueño de tu cuenta mediante un token de acceso personal.

[1] Un *issue* bloqueada es una que no puede recibir más comentarios.

Archivos

Archivos de código

- Modificar `parametros.py`: Contiene los parámetros a usar en la actividad tales como: rutas a archivos de texto, urls de las APIs a utilizar, y datos que necesitarás completar para realizar las *requests*.
- Modificar `apis.py`: Contiene las funciones que realizarán los llamados a las API de la actividad.
- No modificar `main.py`: Inicia el menú principal. Debes ejecutar la actividad desde este archivo.
- No modificar `utils/anime.py`: Contiene la `NamedTuple` de Anime. También incluye la función para filtrar los animes y un ejemplo de cómo instanciar la `NamedTuple` de Anime.
- No modificar `utils/menus.py`: Contiene la lógica de los distintos menús de la actividad.
- No modificar `utils/data.py`: Contiene las clases para manejar a datos en archivos .csv.

Archivos de datos

- `data/tokens.csv`: este archivo almacena tu token de acceso personal a Github para que puedas ejecutar la actividad sin ingresarlo nuevamente.
- `data/issues.csv`: cada vez que crees una issue en tu repositorio, este archivo almacenará el número de dicha issue en una nueva fila.

Parte I: Consiguiendo tus Animes

En esta primera parte conseguirás los datos que usarás como contenido para crear una *Issue*. Para esto, necesitarás de un ID que determinará qué animes recibir. Lo puedes calcular realizando la operación **módulo 6** sobre los **dos penúltimos dígitos** de tu **número de alumno**. Por ejemplo:

- Si tu número de alumno es 15638**35**9, el ID de tus animes asignado es **35** % **6** = es **5**.
- Si tu número de alumno es 0560**83**J, el ID de tus animes asignado es **83** % **6** = es **5**.

Deberás modificar la variable `ANIMES_NUMERO` del archivo `parametros.py` con el ID de tus animes asignados. Luego, deberás modificar la siguiente función del archivo `api.py`, procurando utilizar los parámetros `ANIME_BASE_URL` y `ANIMES_NUMERO` al momento de trabajar:

- Modificar `def get_animes() -> Tuple[int, List[Anime]]:`
  Este función deberá llamar a la API de Anime Ranking y obtener los datos de tus animes asignados. Para lograr esto, deberás hacer una `requests` del tipo **GET** a la siguiente ruta:

  `https://backend.chan.ing.puc.cl/animeranking/v1/AB?id=ANIMES_NUMERO`

  Una vez obtenidos los datos, debes retornar una lista de `NamedTuple` Anime siguiendo el mismo orden en el cuál llegaron al momento de hacer la `requests`.

Una vez completada esta parte, puedes ejecutar la opción *Get Animes* del menú contenido en `main.py` y deberías poder ver los datos obtenidos de tus animes en la terminal/consola.

Importante: En caso que la API de Anime Ranking presente problemas o caídas, puedes utilizar temporalmente el siguiente enlace para obtener un JSON con **el mismo formato** que el esperado por la API.

Parte II: Filtrar la lista

En esta parte deberás implementar un **patrón en REGEX** para filtrar la lista de animes. Para esto, deberás modificar la variable `REGEX_FILTRO` del archivo `parametros.py` con la expresión regular que permita filtrar los animes. Esta expresión debe seleccionar aquellos animes que cumplan con alguna de las siguientes condiciones:

- Existe algún substring que empiece con "ha", luego sigan 1 o más caracteres alfanuméricos y finalmente contenga una "o".
- Existe algún *substring* con al menos 3 "a".

No es necesario considerar mayúsculas, ya que el código usará `.lower()` antes de realizar el `.match()`.

Ejemplos de frases que cumplen con estas condiciones son:

- Banana
- Chacarero estilo italiano
- Hanako-kun besto husbando
- Muchas Patatas Fritas

Parte III: Crear y modificar *issues* de Github

En esta segunda parte deberás trabajar con los datos de animes obtenidos anteriormente para generar una *issue* en tu repositorio personal de del curso.

Es importante saber que la API de Github requiere *headers* de Autenticación para cualquiera de las *requests* pedidas en esta actividad. Para esto, necesitarás tu *token* de acceso personal para tu cuenta de Github. Podrás obtener tu token desde esta página, y durante los primeros minutos de la cátedra tu profesor mostrará cómo configurarlo adecuadamente.

TIP: el formato necesario para crear un diccionario que posteriormente puede ser pasado como *headers* a la función de *request* es la siguiente:

```
headers = {
    'llave': 'valor de la llave',
    'llave 2': 'otro valor',
    ...
}
```

Además, podrás encontrar la documentación de la API de Github respecto al manejo de Issues en la siguiente página. Deberás buscar, para cada función pedida en esta parte, la documentación del *endpoint* apropiado que te permitirá realizar lo pedido.

En ellas deberás usar los parámetros `GITHUB_BASE_URL`, `GITHUB_REPO_OWNER`, `GITHUB_REPO_NAME` y `GITHUB_USERNAME` para construir las llamadas a la API. Deberás completar `GITHUB_REPO_NAME` y `GITHUB_USERNAME` con el nombre de tu repositorio y tu nombre de usuario respectivamente.

Las funciones a implementar son:

- Modificar `def post_issue(token: str, animes: List[Anime]) -> Tuple[int, int]:`
  Recibe un **token de acceso personal de Github**, y el **listado con información de tus animes** obtenido en la parte anterior. Con estos datos, deberás realizar una *request* de tipo `POST`, para generar una *issue* cuyo título sea tu **nombre de usuario de Github** y cuyo contenido o *body* sea el listado con la **información de los animes**. El *body* **debe contener, al menos, el nombre de cada anime en el formato que estimen conveniente**. Una vez realizada la *request*, deberás retornar una tupla que contenga el código de estado de la respuesta y el *number* de la *issue* creada.
- Modificar `def put_lock_issue(token: str, numero_issue: int) -> int:`
  Recibe un **token de acceso personal de Github**, y un *número de issue a bloquear*. Debes realizar una *request* que permita bloquear (agregar un *lock*) a la *issue* mediante el método `PUT` y retornar el código de estado que entregue la respuesta.
- Modificar `def delete_lock_issue(token: str, numero_issue: int) -> int:`
  Recibe un **token de acceso personal de Github**, y un **número de *issue* a desbloquear**. Deberás realizar una *request* que permita desbloquear (o eliminar el *lock*) de la *issue* ingresada mediante el método `DELETE` y retornar el código de estado que entregue la respuesta.

Una vez completada cada función de esta parte, podrás ejecutar los métodos respectivos en el menú del archivo `main.py` y comprobar en la sección de *issues* de tu repositorio privado si tu *issue* fue creada, bloqueada o desbloqueada según corresponda.

Notas

- Esta actividad será corregida de **manera automática**, así que procura seguir al pie de la letra las instrucciones de cada método respecto a qué argumentos reciben y qué deberían retornar. Además, procura que tu código no genere excepciones al momento de ejecutarse.
- Siéntete libre de agregar nuevos `print()` en cualquier lugar de tu código para encontrar errores. Es una herramienta muy útil para comprobar tu desarrollo, pero recuerda borrarlos antes de hacer el *commit* final, así evitas problemas con la corrección automática.
- En general, las APIs poseen *rate limiters*, lo que significa que en caso de detectar un flujo inesperadamente alto de *requests* de una misma dirección, denegarán el uso de sus servicios a dicha dirección temporalmente. ¡Ten cuidado con que tu código no envíe cientos de *requests* en un segundo![2]
- **Importante**: En esta actividad estarás haciendo uso de tu *Personal Access Token de Github*, lo que para fines prácticos equivale a tu contraseña de acceso a la aplicación. El menú de la actividad solicita el token como *input* de terminal con el fin de no almacenarla como parámetro en el código, y la almacena dentro de un archivo de extensión `.csv` en la carpeta `data`. Como medida de seguridad, incluimos un archivo `.gitignore` que evitará que dichos archivos de la carpeta se suban al momento de subir tu código. Te recordamos que seas cuidados@ con no dejar tu *token* como variable en el código, o incluirla al momento de subirlo.

Requerimientos

- (0.5 punto) La función `get_animes()` completada correctamente.
- (1 punto) Expresión regular para filtrar animes realizada correctamente.

[2] En esta actividad no se espera que hagas *requests* dentro de un flujo como `while` o `for`.

- (1 punto) La función `post_issue()` completada correctamente.
- (0.5 punto) Las funciones `put_lock_issue()` y `delete_lock_issue()` completadas correctamente.