Gestión de la Información en la Web
Curso 2024-25
Práctica 6 – Uso de APIs

Fecha de entrega: domingo 10 de noviembre de 2024

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero `pr6.py`. El esqueleto de este fichero se puede descargar del Campus Virtual.

Lenguaje de programación
Python 3.11 o superior.

Calificación
Se medirá la corrección mediante tests de unidad. Además de la corrección, se valorará la **calidad, concisión y claridad del código**, la incorporación de **comentarios** explicativos, su **eficiencia** tanto en tiempo como en memoria y la puntuación obtenida en Pylint.

Declaración de autoría e integridad
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo y los autores. Esta cabecera también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ningún fichero que no venga acompañado de dicha cabecera.

En esta práctica implementaremos una serie de funciones para acceder y modificar los datos almacenados en el API REST de «Go REST» (`https://gorest.co.in/`). Todas estas funciones toman como parámetro `token` el *token* necesario para usar el API, así que para poder realizar la práctica lo primero que debéis es crearos un usuario en «Go REST» y obtener vuestro *token* (podéis compartirlo entre los distintos miembros del grupo).

1. Insertar usuarios [1.5pt]

Implementar la función `inserta_usuarios(datos, token)` que recibe una lista de diccionarios con la información de varios usuarios y los almacena en el servicio web. Esta función devolverá `True` si todos los usuarios se han podido añadir correctamente, en otro caso devolverá `False`

Ejemplos de llamadas *(la segunda llamada devuelve `False` porque ya existe un usuario de email `ana@gmail.com`, aunque insertará al usuario de email `pepe@gmail.com`)*

```
>>> u1 = {'name': 'Eva', 'email': 'eva@gmail.com', 'gender': 'female', 'status': 'inactive'}
>>> u2 = {'name': 'Ana', 'email': 'ana@gmail.com', 'gender': 'female', 'status': 'active'}
>>> u3 = {'name': 'Pepe', 'email': 'pepe@gmail.com', 'gender': 'female', 'status': 'inactive'}
>>> inserta_usuarios([u1, u2], api_token)
True
>>> inserta_usuarios([u3, u2], api_token)
False
```

2. Obtener ID de usuario a partir del email [1.5pt]

Implementar la función `get_ident_email(email, token)` que devuelve el identificador del usuario cuyo email **es exactamente igual al pasado como parámetro**. En caso de que dicho usuario no exista, devolverá `None`.

Ejemplos de llamadas *(la segunda devuelve `None` porque el usuario no existe)*

```
>>> get_ident_email('eva@gmail.com', api_token)
3944143
>>> get_ident_email('barbara@gmail.com', api_token)
None
```

3. Borrar un usuario [1.5pt]

Implementar la función `borra_usuario(email, api_token)` para eliminar el usuario cuyo email es exactamente igual al pasado como parámetro. En caso de éxito en el borrado devolverá `True`, en caso contrario devolverá `False`.

Ejemplos de llamadas *(la segunda llamada no puede borrar el usuario porque ya no existe)*

```
>>> borra_usuario('eva@gmail.com', api_token)
True
>>> borra_usuario('eva@gmail.com', api_token)
False
```

4. Insertar una tarea (ToDo) a un usuario [1.5pt]

Implementar la función `inserta_todo(email, token, title, due_on, status)` para insertar una nueva tarea asociada al usuario con email exactamente igual al pasado como parámetro. Si la tarea ha sido insertada con éxito devolverá `True`, en otro caso devolverá `False`. Para representar las fechas tope de las tareas usaremos un formato `YYYY-MM-DD HH:MM` inspirado en el estándar ISO-86011[1].

Ejemplos de llamadas *(la segunda llamada falla porque el usuario no existe)*

```
>>> inserta_todo('eva@gmail.com', api_token, 'Todo 1', '2034-07-28 11:30', 'pending')
True
>>> inserta_todo('barbara@gmail.com', api_token, 'Todo 1', '2034-07-28 11:30', 'pending')
False
>>> inserta_todo('eva@gmail.com', api_token, 'Todo 2', '2020-06-28 16:45', 'completed')
True
```

5. Listar las tareas de un usuario [1.5pt]

Implementar la función `lista_todos(email, token)` que devuelve una lista de diccionarios con la información de todas las tareas asociadas al usuario con email exactamente igual al pasado como parámetro. **Esta lista tendrá las tareas en el mismo orden en el que son devueltas por el API, es decir, no debéis realizar ninguna ordenación manual.** Si no existe ningún usuario con ese email deberá devolverse la lista vacía.

Ejemplos de llamadas *(la segunda llamada falla porque el usuario no existe)*

```
>>> lista_todos('eva@gmail.com', api_token)
[{'due_on': '2013-04-28T23:59:00.000+05:30',
  'id': 21347,
  'status': 'pending',
  'title': 'Todo 3',
  'user_id': 3960157},
 {'due_on': '2020-06-28T16:45:00.000+05:30',
  'id': 21346,
  'status': 'completed',
  'title': 'Todo 2',
  'user_id': 3960157},
 {'due_on': '2034-07-28T11:30:00.000+05:30',
  'id': 21345,
  'status': 'pending',
  'title': 'Todo 1',
  'user_id': 3960157}]
>>> lista_todos('barbara@gmail.com', api_token)
[]
```

6. Listar las tareas no cumplidas de un usuario [2.5pt]

Implementar una función `lista_todos_no_cumplidos(email, token)` que devuelve una lista de diccionarios con todas las tareas *no cumplidas* asociadas al usuario del email pasado como parámetro. **Esta lista tendrá las tareas en el mismo orden en el que son devueltas por el API, es**

[1] `https://es.wikipedia.org/wiki/ISO_8601`

decir, no debéis realizar ninguna ordenación manual. Diremos que una tarea está *no cumplida* si:

- Está pendiente, es decir, `status=pending`
- La fecha tope (`due_on`) es anterior a la fecha y hora actual. Para simplificar la comparación de las fechas, tened en cuenta únicamente el dia, la hora y los minutos; es decir, ignorar los segundos, microsegundos y el huso horario. Para obtener la fecha actual y para comparar fechas, utilizar el módulo `datetime`[2] de Python.

Ejemplos de llamadas *(la segunda llamada devuelve la lista vacía porque el usuario no existe)*

```
>>> lista_todos_no_cumplidos('eva@gmail.com', api_token)
[{'due_on': '2013-04-28T23:59:00.000+05:30',
  'id': 21347,
  'status': 'pending',
  'title': 'Todo 3',
  'user_id': 3960157}]
>>> lista_todos_no_cumplidos('barbara@gmail.com', api_token)
[]
```

[2] `https://docs.python.org/3/library/datetime.html`