PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (I/2018)

Tarea 7

Entrega

- `Tarea`
  - **Fecha/hora:** 1 de julio del 2018, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T07/`
- `README.md`
  - **Fecha/hora:** 2 de julio del 2018, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T07/`

Objetivos

- Interacción con distintas API de *web services* para la creación de un programa.
- Estudiar y utilizar la documentación de diferentes API.
- Crear y utilizar una sencilla API propia.
- Sintetizar conocimientos del curso.

Índice

1. Introducción

Al terminar tu tarea 6, quedaste tan emocionado con la música que decides escuchar el *Piano Concerto 21* de Wolfgang Amadeus Mozart[1]. Mientras escuchas esta maravilla, se te ocurre realizar un programa para avisarle a distintos usuarios el pronóstico del tiempo de una ciudad por medio de correos electrónicos, llamado *MaiWeather*.

2. *MaiWeather*

MaiWeather es un sistema que permite que sus usuarios puedan obtener información del tiempo —de algún lugar del mundo— en su bandeja de correo electrónico. Está compuesto por dos programas.

1. **Servidor:** Provee una API sencilla, desarrollada por ti, a la que se le puede pedir que envíe información del tiempo a cierta dirección de correo electrónico. Para hacer eso posible, este programa consume la API de OpenWeatherMap y de Gmail.

2. **Cliente:** Provee una interfaz por línea de comandos, **que consume solamente la API provista por el servidor que desarrollaste**. Esto permitirá a un usuario pedir su reporte del tiempo, con ciertos datos, a una dirección de correo a elección.

3. Desarrollo de tu API

Para esta tarea, deberás desarrollar un programa servidor y proveer una API sencilla con Flask[2], que es una librería externa escrita en Python (debes instalarla con `pip`). La API que crees debe usar las API de Gmail y OpenWeatherMap, explicadas mas adelante, para generar reportes del tiempo, y enviarlas a un *email* dado.

Queda a tu criterio la forma exacta de cómo recibir y responder los *requests*, ya que tú mismo desarrollarás el cliente de tu API.

3.1. *OpenWeatherMap API*

Para adquirir la información sobre el pronóstico del tiempo usarás la API de OpenWeatherMap, cuya documentación se puede encontrar en este enlace[3].

3.1.1. *Setup*

Para utilizar esta API se debe:

1. Crear una cuenta en `https://www.openweathermap.org`

2. Acceder a `https://home.openweathermap.org/api_keys` para obtener una *key* que te permitirá consumir la API.

3.1.2. Información a utilizar

Se requiere obtener el tiempo del día actual y el pronóstico para los próximos 5 días para cierto lugar. A la vez, deberás ser capaz de obtener el índice UV para el día actual y el pronóstico para los próximos 8 días. El lugar para el que se piden los reportes deberá poder ser recibido por tu API de las siguientes maneras:

[1] `https://youtu.be/df-eLzao63I`
[2] `http://flask.pocoo.org/`
[3] Haz clic en "este enlace" para abrirlo. De aquí en adelante, siempre que diga "este enlace" podrás pinchar sobre ese texto.

1. Nombre de ciudad
2. ID de ciudad, según un JSON que puede ser encontrado en este enlace
3. Coordenadas geográficas (latitud y longitud)

IMPORTANTE: no se debe subir el archivo JSON del enlace del punto 2, y este debe ser incluido en un `.gitignore`, para así evitar el uso de memoria innecesaria en los repositorios.

En relación al **tiempo**, se debe poder consultar como mínimo por la siguiente información, la cual es proporcionada por la respuesta de OpenWeatherMap (descripción del JSON de retorno del tiempo actual en este enlace y del pronóstico en este enlace):

1. Descripción del tiempo: `data["weather"]["description"]`
2. Temperatura: `data["main"]["temp"]`
3. Presión: `data["main"]["pressure"]`
4. Humedad: `data["main"]["humidity"]`
5. Temperatura mínima: `data["main"]["temp_min"]`
6. Temperatura máxima: `data["main"]["temp_max"]`
7. Velocidad del viento: `data["wind"]["speed"]`
8. Nubosidad: `data["clouds"]["all"]`

Es importante mencionar que la manera en que se retornan los JSON es diferente para el tiempo actual y para el pronóstico.

Por otro lado, para el **índice UV**, los únicos parámetros que deberás recuperar de la respuesta de OpenWeatherMap son los siguientes (descripción del JSON de retorno en este enlace):

1. Fecha: `date`
2. Valor: `value`

Es importante mencionar que si se consulta por más de un día, el retorno del JSON es en formato de lista. Por otro lado, la API de OpenWeatherMap para el índice UV sólo permite ingresar coordenadas geográficas para la ubicación. Sin embargo, tu API deberá aceptar también un nombre de ciudad o el ID de la misma. Toda la información necesaria para hacer la *traducción* está en los JSON de los ID de las ciudades. Por ejemplo, la información correspondiente a Santiago es la siguiente:


```
{
  "id": 3873544,
  "name": "Region Metropolitana de Santiago",
  "country": "CL",
  "coord": {
    "lon": -70.64724,
    "lat": -33.472691
  }
}
```

3.1.3. Forma de hacer llamados a la API

Por defecto, la API de OpenWeatherMap retorna los datos unidades del SI (Kelvin), pero también puede retornar los datos en unidades en el sistema imperial (Farenheit) o en el sistema métrico (Celsius), por lo que se debe poder satisfacer al usuario y devolver en el sistema correspondiente al que este seleccione al minuto de ser inscrito desde la aplicación cliente —que será explicada después.

Para hacer un llamado a la API se debe hacer algo similar a lo que sale en la documentación (disponible en este enlace), con los ejemplos de cada `API Doc`, **pero** se debe incluir la *key* del usuario.

Por ejemplo, si se quiere obtener la información del tiempo **actual** de Santiago, se debería hacer lo siguiente:
`http://api.openweathermap.org/data/2.5/weather?id=3873544&units=imperial&appid=API_KEY`

Por otro lado, si se quisiera obtener el pronóstico se debería hacer lo siguiente:
`http://api.openweathermap.org/data/2.5/forecast?id=3873544&appid=API_KEY`

Mientras que si se quisiera obtener la **información UV** se debe hacer lo que sigue:
`http://api.openweathermap.org/data/2.5/uvi?lat=-33.472691&lon=-70.64724&appid=API_KEY`

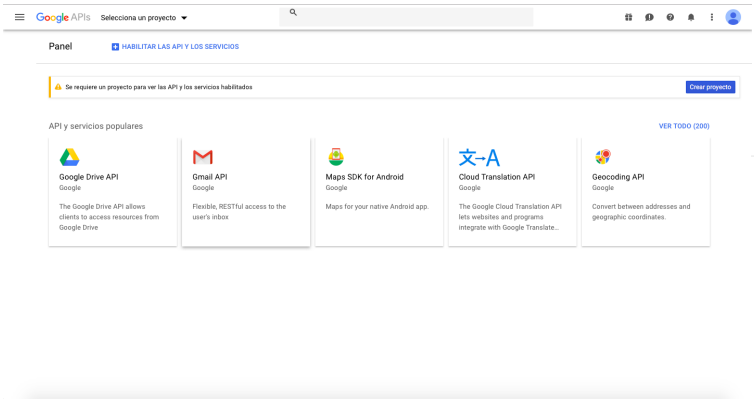
Finalmente, si se quiere obtener el pronóstico del índice UV, se debe hacer lo siguiente:
`http://api.openweathermap.org/data/2.5/uvi/forecast?lat=-33.472691&lon=-70.64724&appid=API_KEY`

3.2. *Gmail API*

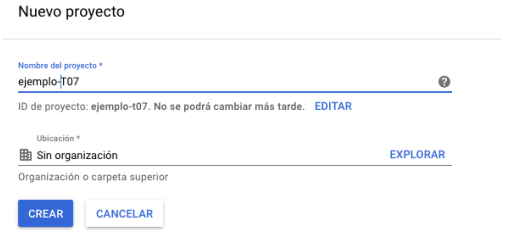
Para poder enviar correos con la información que les proporciona OpenWeatherMap, deberás integrar en tu tarea la **API de Gmail** que ofrece Google para así poder enviar correos con la información del tiempo. La mayoría de las API de Google requieren de un procedimiento parecido para habilitar el servicio. A continuación se explicará cómo llevar a cabo estos pasos:

1. Acceder a la **consola de desarrolladores de Google**
2. Crear un nuevo proyecto
3. Volver a la consola y seleccionar el proyecto ya creado
4. Habilitar la API de Gmail, probablemente aparezca ahí mismo dentro de los servicios populares. Si no aparece, la pueden buscar haciedo clic en "Habilitar las API y los servicios"
5. Crear las credenciales, descargarlas en formato JSON y cambiar el nombre a `client_secret.json`

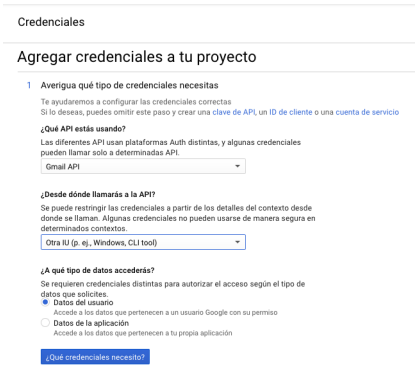
Luego, pueden continuar con el *quickstart* que ofrece Google que pueden encontrar este enlace y toda la información como guías y referencias sobre la API la pueden encontrar en este enlace.



(a) Consola de Desarrolladores

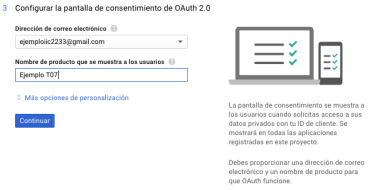


(b) Creación del Proyecto



(c) Credenciales paso 1

(d) Credenciales paso 2



(e) Credenciales paso 3

3.3. Bonus: *Yoda Translator API*

Como ya está terminando el semestre (y tu curso favorito, *Programación avanzada*), sientes que eres todo un(a) *Hackerman*. Por esta razón, deseas que tu programa sea utilizado en todo el universo, y qué mejor forma para comunicarte con ciudadanos de otros planetas que utilizando el lenguaje de *Yoda*.

3.3.1. Descripción

La API que debes consumir se llama **Yoda Translator**, y su documentación está en este enlace. La función de esta API es *traducir* o convertir la frase entregada a una nueva frase, la que estará escrita con la gramática que utiliza Yoda. En la siguiente tabla, se encuentran algunos ejemplos:

Frase normal (*input*)	**Frase en estilo Yoda (*output*)**
May the force be with you.	*With you, may the force be.*
Help me, Obi-Wan Kenobi. You're my only hope.	*Me, help, Obi-Wan Kenobi. You're my only hope.*
No. I am your father.	*No. Your father, I am.*

Como *bonus* para tu tarea, debes consultar esta API para entregar los resultados de tus consultas acerca del tiempo en lenguaje Yoda. Debes entregar una frase que contenga alguna de las palabras entregadas por la API de OpenWeatherMap, para que esta sea convertida a lenguaje Yoda y posteriormente enviada por la API de Gmail. El formato de las oraciones es libre (queda a tu elección el contenido de cada una).

3.3.2. Funcionamiento

El funcionamiento de las consultas a esta API es bastante simple. Mediante alguna librería para interactuar con *web sevices* (`requests`, por ejemplo), debes enviar la frase que quieres convertir (utilizando el método `GET`). El resultado de esta consulta será un `JSON` con la información correspondiente.
Por ejemplo, si deseamos convertir la frase *May the force be with you*:

- La consulta que se debe hacer es: `http://api.funtranslations.com/translate/yoda.json?text=May the san be with you`
- El resultado de aquella consulta es:

```
{
  "success": {
    "total": 1
  },
  "contents": {
    "translated": "With you,  may the force be",
    "text": "May the force be with you",
    "translation": "yoda"
  }
}
```

No olvides considerar que, como aparece en la documentación, esta API posee un **máximo de 60 consultas por día**, y las consultas deben ser hechas en inglés.

4. Consumir tu API

En esta segunda parte, deberás crear un cliente que, por consola, permita a un usuario ingresar su correo y algunas configuraciones, para recibir un reporte del tiempo —todo esto haciendo solicitudes a tu API generada en la sección anterior.

4.1. Validación de correos

Primero, tu cliente debe pedir una dirección **válida** de correo. Esto simulará un inicio de sesión, y se mantendrá hasta que el usuario cierre el cliente, o elija la opción de cerrar sesión. Además, se considerará una dirección de *email* válida aquellas que cumplan cada uno de los siguientes requisitos:

- Contener sólo caracteres alfanuméricos de ASCII, además de la arroba, puntos y guiones bajos.
- Contener exactamente una **@**
- Contener exactamente un **.** en el dominio (*i.e.* después de la arroba)
- Que luego del **.** sólo existan carácteres dentro del abecedario. Es decir, no puede existir una @ luego de un punto ni tampoco un número.

Por ejemplo, direcciones como `fernand0@pepperoni.come` o `hern@n.vegetariano` tienen un formato válido, pero otras como `nebil@pasamecon.4` o `cobreloa.2@B` no lo tienen. Es necesario usar expresiones regulares (*regex*) para validar estas reglas.

4.2. Enviar reporte del tiempo

Al seleccionar esta opción, tu cliente debe hacer una *request* a tu API, que, dada la configuración seleccionada (explicada después) genere un reporte del tiempo usando la API de OpenWeatherMap y luego envíe un correo al usuario con la API de Gmail.

Por ejemplo, para el usuario `Fernando` con correo `fernand0@pepperoni.come` que sólo le interesa la información acerca de la humedad y la temperatura máxima en las ciudades que previamente señaló, debería recibir un correo del siguiente estilo:

```
 Hola Fernando, MaiWeather te reportará información del tiempo:
- San Diego: humedad : 25%, temperatura máxima: 32°C
- Santiago: humedad : 34%, temperatura máxima: 21°C
```

4.3. Configuración del reporte

Luego de que el usuario indique su correo, deberá partir con ingresando una ciudad y su país en el formato que se estime conveniente para que reciba reportes del tiempo de ese lugar. Debe revisarse que esta ciudad pertenezca al archivo `city.list.json`, y avisar en caso contrario.

Luego, el usuario debe poder entrar a la configuración del reporte y elegir qué información va a recibir en su informe. Este tiene dos partes: el informe del tiempo y la radiación UV. Las opciones existentes son:

1. Añadir o eliminar ciudades (siempre tiene que quedar **al menos una**).
2. Elegir si el reporte del tiempo va a ser sobre el día actual, sobre los próximos 5 días, ambas opciones o si no quiere ver ninguna información respecto al tiempo.
3. En el reporte del tiempo, elegir qué parámetros desea recibir:
   a) Descripción del tiempo
   b) Temperatura
   c) Presión
   d) Humedad
   e) Temperatura mínima
   f) Temperatura máxima
   g) Velocidad del viento
   h) Nubosidad
4. Elegir si el reporte UV será sobre el día actual, los próximos 8 días, ambas opciones o ninguna.
5. Indicar sus unidades de preferencia, ya sean unidades del SI (Kelvin), sistema imperial (Farenheit)o sistema métrico (Celsius).

Por defecto, el usuario recibe los dos informes, para el día actual y con todos los parámetros existentes.

4.4. Interacción en consola

Finalmente, la interacción con tu cliente debería seguir este orden.

1. Pedir el correo y el nombre
2. Mostrar un menú con las siguientes opciones.

- Ver configuración actual
- Cambiar configuración de reportes
- Mandar reporte del tiempo
- Cerrar sesión

5. Consideraciones

5.1. `.gitignore`

Para no saturar los repositorios de GitHub, **deberás** agregar en tu carpeta `Tareas/T07/` un archivo llamado `.gitignore` de tal forma de **no** subir el archivo `city.list.json` que se describe en la sección 3.1.

6. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el *Código de honor de la Escuela*: haz clic aquí para leer.
- Tu programa debe ser desarrollado en Python v3.6.
- Tu código debe seguir la guía de estilos descrita en el PEP8.
- Si no se encuentra especificado en el enunciado, asume que el uso de cualquier librería Python está prohibida. Pregunta en el foro si es que es posible utilizar alguna librería en particular.
- El ayudante puede castigar el puntaje[4] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- Debes adjuntar un archivo `README.md` donde comentes sus alcances y el funcionamiento del sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tendrás hasta 24 horas después de la fecha de entrega** de la tarea para subir el `README.md` a tu repositorio.
- Crea un módulo para cada conjunto de clases. Divídelas por las relaciones y los tipos que poseen en común. **Se descontará hasta cinco décimas si se entrega la tarea en un solo módulo[5].**
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).

[4] Hasta 5 décimas.
[5] No agarres tu código de un solo módulo para dividirlo en dos; separa su código de forma lógica