



Centro Universitario de Ciencias  
Exactas e Ingenierías.



Ingeniería en computación.

Alumno: Vargas López David Guadalupe.

Computación tolerante a fallos.

Profesor: López Franco Michel Emanuel.

Sección: D06.



Guadalajara Jal. marzo del 2022.

## Workflow managers

### ¿Qué es?

Prefect es un sistema de gestión de flujo de trabajo basado en Python basado en una premisa simple: *su código probablemente funcione, pero a veces no* (fuente). Nadie piensa en los sistemas de flujo de trabajo cuando todo funciona como se esperaba. Pero cuando las cosas vayan mal, Prefect garantizará que su código falle correctamente.

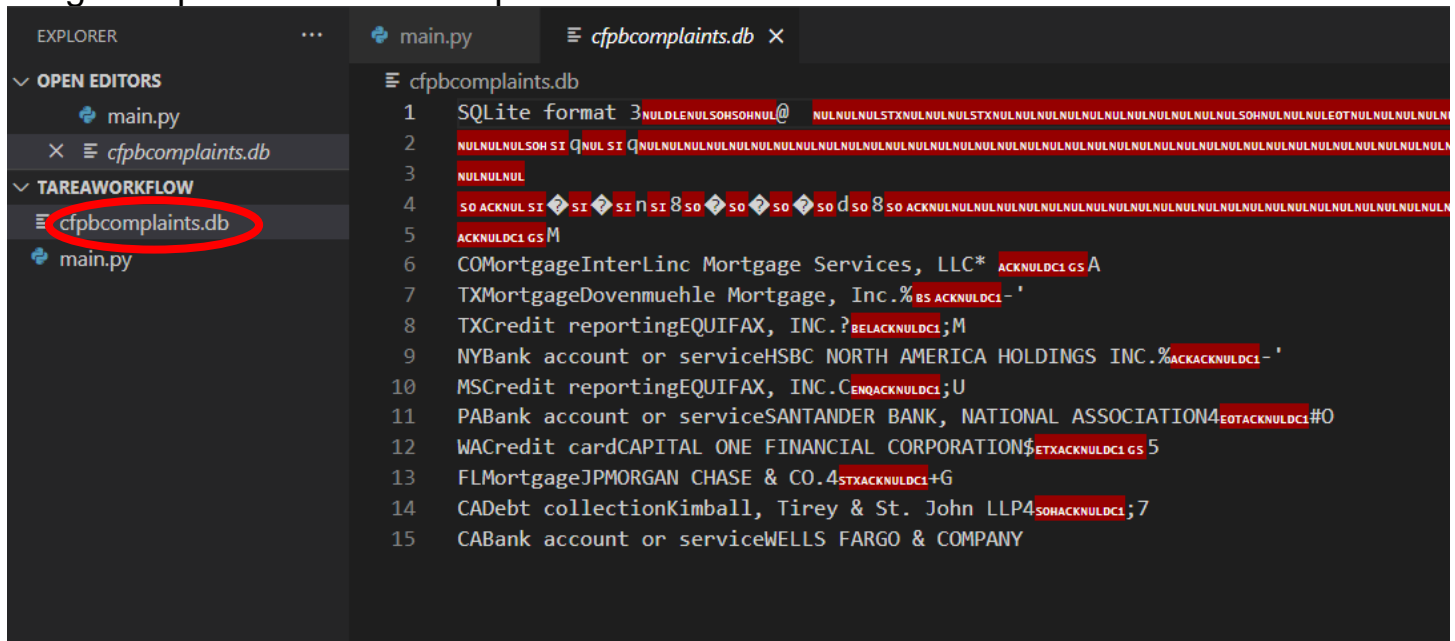
Como sistema de administración de flujo de trabajo, Prefect facilita la adición de registros, reintentos, mapeo dinámico, almacenamiento en caché, notificaciones de fallas y más a sus canales de datos. Es invisible cuando no lo necesita, cuando todo funciona como se espera y visible cuando lo hace. Algo parecido a un seguro.

Si bien Prefect no es el único sistema de gestión de flujo de trabajo disponible para los usuarios de Python, es sin duda el más competente. Las alternativas como Apache Airflow suelen funcionar bien, pero presentan muchos dolores de cabeza cuando se trabaja en grandes proyectos.

### Programa creado en tutorial:

Primero que nada, en este programa se ejecuta una especie de canalización de datos en la que se descarga datos desde una página web o una aplicación y se copian dentro de una base de datos la cual funciona como host principal para el almacenamiento de datos, por eso es por lo que al correr el siguiente programa se crea un archivo .db debido a que para esto se utiliza la librería de sqlite3 para poder crear la base de datos con los datos de la aplicación ficticia que se están almacenando.

## Vargas López David Guadalupe.



```
1 SQLite format 3
2
3
4
5
6 COMortgageInterLinc Mortgage Services, LLC*
7 TXMortgageDovenmuehle Mortgage, Inc.%
8 TXCredit reportingEQUIFAX, INC.?
9 NYBank account or serviceHSBC NORTH AMERICA HOLDINGS INC.%
10 MSCredit reportingEQUIFAX, INC.C
11 PABank account or serviceSANTANDER BANK, NATIONAL ASSOCIATION
12 WACredit cardCAPITAL ONE FINANCIAL CORPORATION$
13 FLMortgageJPMORGAN CHASE & CO.4
14 CADebt collectionKimball, Tirey & St. John LLP4
15 CABank account or serviceWELLS FARGO & COMPANY
```

Posteriormente esto tenemos las librerías del programa las cuales son variadas dependiendo a las funciones que se requieren para hacer funcionar el programa cómo se desea, se tienen algunas librerías importantes como lo son la librería de sqlite3 ya que como se dijo anteriormente esta es la encargada de poder almacenar los datos en otra base de datos externa a donde se están almacenando los datos de El sitio web, por ejemplo, también se tiene la librería. Json la cual se encarga de poder hacer la extracción de los datos a través de los request y los json.

```
import requests
import json
from collections import namedtuple
from contextlib import closing
import sqlite3
import os
os.environ["PATH"] += os.pathsep + 'C:\Program Files\Graphviz\bin'
```

Vargas López David Guadalupe.

Como se dijo anteriormente este programa funciona a base de una base de datos en donde se van almacenando los datos generados por un sitio en este caso simularemos un sitio web, por lo que primero que nada se deben extraer los datos para posteriormente tratarlos y almacenarlos en la base de datos correspondiente, para ello se crea una función específica para hacer esta acción mediante la librería JSON en la que utiliza la función get para obtener los datos y transformarlos o un arreglo de datos cómo logras un diccionario o una lista.

```
@task
def get_complaint_data():
    r = requests.get("https://www.consumerfinance.gov/data-research/consumer-complaints/search/api/v1/", params={'size':10})
    response_json = json.loads(r.text)
    return response_json['hits']['hits']
```

Posteriormente se tiene otra función específica para el tratamiento de los datos que se están extrayendo para posteriormente tratarlos y enviarlos a la base de datos, como se mencionó anteriormente este tipo de ejemplos tratan los datos convirtiéndolos ya sea en tuplas de un diccionario o en una lista por lo que se utiliza un for para leer línea por línea todos los datos que se están almacenando y de esta forma hacer el parceo correspondiente de todos los datos.

```
@task
def parse_complaint_data(raw):
    complaints = []
    Complaint = namedtuple('Complaint', ['data_received', 'state', 'product', 'company', 'complaint_what_happened'])
    for row in raw:
        source = row.get('_source')
        this_complaint = Complaint(
            data_received=source.get('date_recieved'),
            state=source.get('state'),
            product=source.get('product'),
            company=source.get('company'),
            complaint_what_happened=source.get('complaint_what_happened')
        )
        complaints.append(this_complaint)
    return complaints
```

Vargas López David Guadalupe.

Después lo que sigue en el proceso, es básicamente una vez tratados los datos como se menciona anteriormente, se tienen que introducir a una nueva base de datos de manera ordenada para que de esta forma se almacenen ahí en segundo plano, cabe destacar que para esto se utilizan instrucciones de SQL que en este caso en más específicos se utiliza la función insert como se muestra en el ejemplo que a continuación se presenta.

```
@task
def store_complaints(parsed):
    create_script = 'CREATE TABLE IF NOT EXISTS complaint (timestamp TEXT, state TEXT,
product TEXT, company TEXT, complaint_what_happened TEXT)'
    insert_cmd = "INSERT INTO complaint VALUES (?, ?, ?, ?, ?)"

    with closing(sqlite3.connect("cfpbcomplaints.txt")) as conn:
        with closing(conn.cursor()) as cursor:
            cursor.executescript(create_script)
            cursor.executemany(insert_cmd, parsed)
            conn.commit()
```

Y por último lo que queda es observar los resultados de este programa de forma gráfica por lo que el siguiente fragmento de código pertenece a la función que realiza esa función y muestra una pequeña gráfica o tableta en donde se muestran los datos del programa en sí.

```
with Flow("my etl flow") as f:
    raw = get_complaint_data()
    parsed = parse_complaint_data(raw)
    store_complaints(parsed)

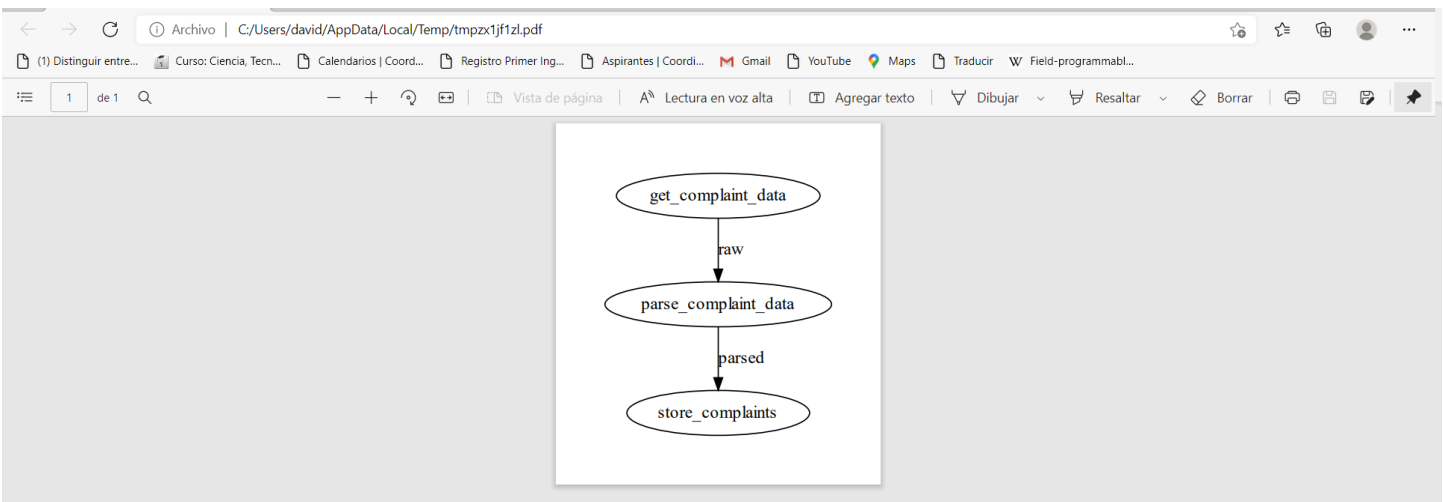
f.visualize()
f.run()
```

## Capturas de pantalla:

Aquí se muestran algunas de las tareas o procesos que se están ejecutando y el estatus de cada uno de ellos desde el momento en que inicia hasta el momento de su finalización y de esta forma se crea una especie de bucle en el que se están interceptando todos los procesos que se están ejecutando ya sea en una página web o en cualquier otro sitio.

```
PS C:\Users\david\OneDrive\Escritorio\6to semestre\tolerante a fallos\tareaworkflow> python -u "c:\Users\david\OneDrive\Escritorio\6to semestre\flow\main.py"
[2022-03-07 16:11:49-0600] INFO - prefect.FlowRunner | Beginning Flow run for 'my etl flow'
[2022-03-07 16:11:49-0600] INFO - prefect.TaskRunner | Task 'get_complaint_data': Starting task run...
[2022-03-07 16:11:50-0600] INFO - prefect.TaskRunner | Task 'get_complaint_data': Finished task run for task with final state: 'Success'
[2022-03-07 16:11:50-0600] INFO - prefect.TaskRunner | Task 'parse_complaint_data': Starting task run...
[2022-03-07 16:11:50-0600] INFO - prefect.TaskRunner | Task 'parse_complaint_data': Finished task run for task with final state: 'Success'
[2022-03-07 16:11:50-0600] INFO - prefect.TaskRunner | Task 'store_complaints': Starting task run...
[2022-03-07 16:11:50-0600] INFO - prefect.TaskRunner | Task 'store_complaints': Finished task run for task with final state: 'Success'
[2022-03-07 16:11:50-0600] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
PS C:\Users\david\OneDrive\Escritorio\6to semestre\tolerante a fallos\tareaworkflow>
```

Como se dijo anteriormente este programa además se ejecuta por medio de una función gráfica la hora realiza una gráfica de todos los puntos o nodos que se obtienen para obtener el resultado de la imagen anterior y obtener todos los datos necesarios que se están manejando dentro de la página web.



## Conclusiones:

En esta práctica se utilizó una función Perfect la cual es una manera muy nueva para la administración del flujo de datos que en este caso pues se hizo la extracción transformación y carga de datos dentro de una base de datos externó, sin embargo esto se puede transformar en diferentes acciones o necesidades conforme tenga el usuario, cabe destacar que además de la función y librería Prefect, se utilizaron los decoradores task los cuales son capaces de identificar cuáles son las tareas que se están ejecutando dentro de algún lado para poder interceptar las y leerlas por el programa .Py

El módulo profe se puede utilizar de muchas maneras debido a que nos permite un buen control sobre las tareas que se están realizando y poder cambiar el flujo de estas como se hizo en este caso para almacenar los datos dentro de una base de datos externo para en caso de que el programa fallase no se perdieran los datos.

## Bibliografía:

- *Getting Started with Prefect (PyData Denver)*. (2020, April 17). YouTube. Retrieved March 7, 2022, from <https://www.youtube.com/watch?v=FETN0iivZps&t=2545s>
- *JSONPlaceholder - Fake online REST API for developers*. (n.d.). Blog. Retrieved March 7, 2022, from <https://jsonplaceholder.cypress.io/>
- P. (2019, January 3). *Task Failed Successfully - Jeremiah Lowin*. YouTube. [https://www.youtube.com/watch?list=PLMGWGsnelbxcHmA5cVRq8a39S9s\\_gxAMq&v=TlawR\\_gi8-Y&feature=youtu.be](https://www.youtube.com/watch?list=PLMGWGsnelbxcHmA5cVRq8a39S9s_gxAMq&v=TlawR_gi8-Y&feature=youtu.be)

**Enlace al repositorio de GitHub:** <https://github.com/David-1212/prefect.py>