Universidad de los Andes	Taller 1 **ANÁLISIS DE INFORMACIÓN SOBRE BIG DATA**
	Jorge Andrés González Sierra Jairo Fernando Solarte Palacios Diego Alonso Herrera Castro

DOCUMENTACIÓN DE RESULTADOS

Enlace a la aplicación Web
http://172.24.101.71/scrapy/

Métodos y tecnología concretos utilizados en cada uno de los retos propuestos

- Punto 1: Descubrimiento de información utilizando crawling:
  Lenguaje de programación utilizado: Python.
  Framework de apoyo para la obtención de información de las páginas web Scrapy. https://scrapy.org/ . Este permite utilizar selectores para seleccionar ciertas partes del codigo html para extraer la información que se desea. Puede usarse expresiones regulares para filtrar información
  En la página web esta es su función: "Scrapy es un framework de aplicación para rastrear sitios web y extraer datos estructurados que se pueden utilizar para una amplia gama de aplicaciones útiles, como extracción de datos, procesamiento de información o archivo histórico."[1]

- Punto 2: Utilización de fuentes de sindicación/suscripción, RegEX y XQuery
  Lenguaje de programación utilizado: Python
  Librería de apoyo: *feedparser* https://github.com/kurtmckee/feedparser

  Fuentes de sindicación (RSS) utilizadas:
  http://www.france24.com/es/eco-tecno/rss
  http://www.france24.com/es/deportes/rss
  http://www.france24.com/es/noticias/rss
  http://cnnespanol.cnn.com/feed/
  http://feeds.bbci.co.uk/mundo/rss.xml

Expresiones en XQuery y las RegEx que utiliza en la solución

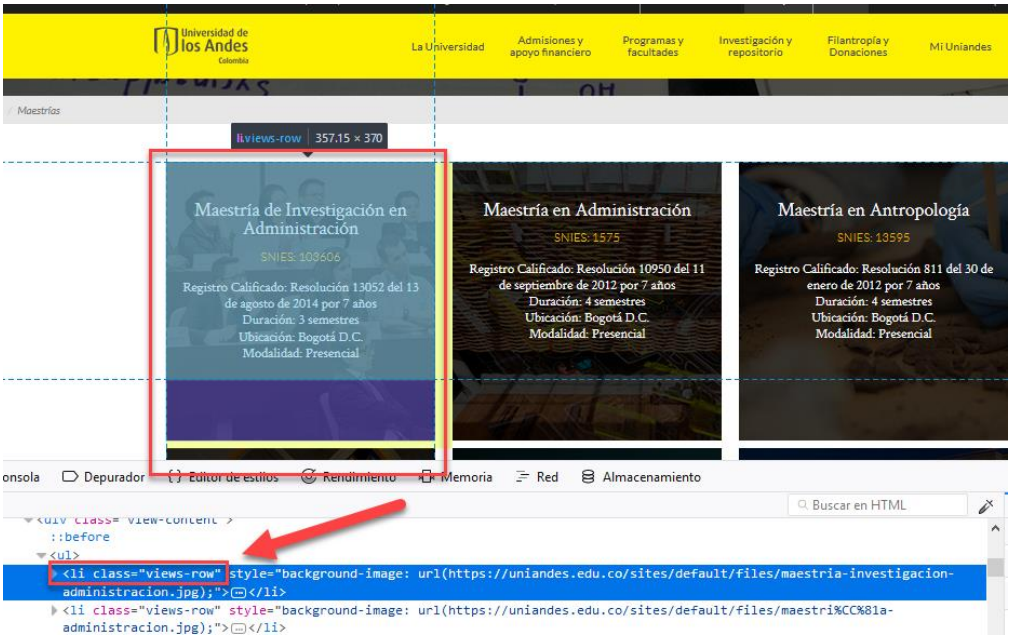
Punto 1:Descubrimiento de información utilizando crawling:
Regex:
Seleccionar Correo:
.*@uniandes[^\n\r]*
Seleccionar Extensión:
(?i)Ext.:\s*\d{1,4}
Seleccionar oficina:
(?i)oficina[^\n\r]*

El **algoritmo básico** para resolver cada uno de los retos, de manera que puedan percibirse los elementos interesantes para poner en valor en la solución

[1] Scrapy https://docs.scrapy.org/en/latest/intro/overview.html

Punto 1:Descubrimiento de información utilizando crawling:

Para obtener las unidades académicas, se buscó en el sitio uniandes.edu.co URLs relacionadas con listado de facultades/departamentos para luego ir a cada una de ellas y realizar la obtención de nombres de departamento, nombre de facultad, y código si aplica. Al estar allí en el listado se hace una obtención recorriendo una estructura identificada de tags de html, y extraída mediante selectores de Scrapy,



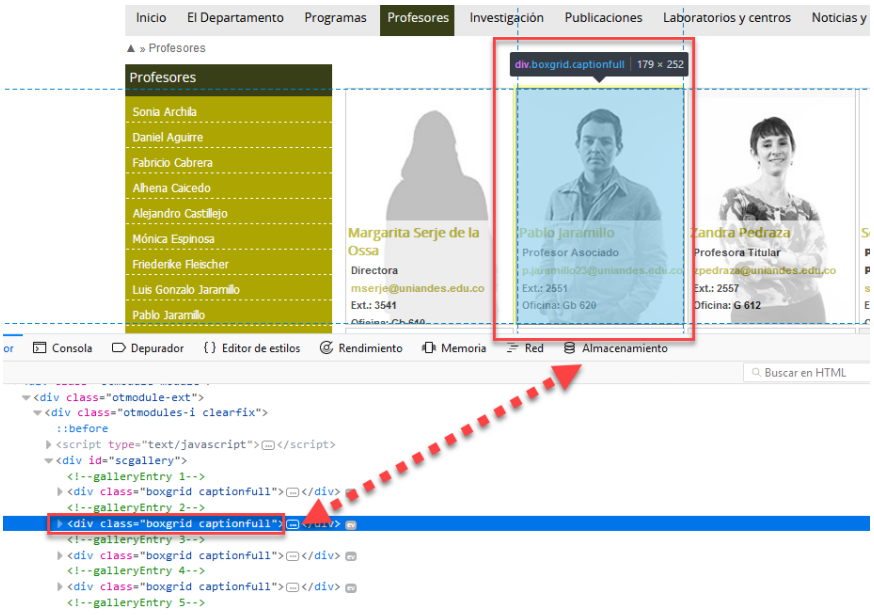
La siguiente grafica muestra parte del código que recorre cada una de estas estructuras de tags, en este caso de los programas de la universidad, obteniendo luego el nombre, url, y código SNIES.

```
for programa in response.css('li.views-row'):
    yield {

        'nombreDepartamento': programa.css('div.views-field.views-field-title-1 span::text').extract_first(),
        'url': programa.css('div.views-field.views-field-title-1 a::attr(href)').extract_first(),
        'snies': programa.css('div.field-content::text').re(r'.*SNIES.*'),

    }
```

De la misma manera similar sucede para los profesores se recorre una estructura identificada mediante tags y se obtiene la información.



Punto 2

La obtención del titular, el link y la fecha de publicación de cada noticia se realiza con una implementación en python de la librería *feedparser*. Dicha librería tiene toda la lógica para descubrir todas las etiquetas RSS y Atom. La implementación consistió en utilizar la función parser del módulo, con la que se logra realizar la obtención de la información en forma de lista. La información de cada item queda almacenada en la tupla entries, por lo que acceder a la información se hace como de cualquier lista de python se tratase. El primer registro de entries es el primer feed consultado, por lo que no se pierde esta secuencia en caso de ser necesaria.

Análisis de resultados obtenidos:

Dificultades:

- Demasiada variedad en las versiones del software, librerías y componentes que se deben utilizar. Los frameworks y librerías utilizadas para el desarrollo tienen como requerimiento python 2.7 mínimo, lo cual presento dificultad ya que la versión del sistema operativo Centos 6.9 maneja Python 2.6. y su actualización genera conflicto con utilidades del Sistema operativo como YUM. Finalmente se logra utilizar las librerías y frameworks con cierta dificultad.
- Información en Javascript no es posible seleccionarla únicamente mediante scrapy.

Logros:

- Encontrar e implementar la librería feedparser que simplifica mucho la adquisición de la información contenida en las etiquetas xml de los *feeds rss*.
- Implementación de framework scrapy destinado para la extraer información de los sitios web, lo que facilito obtener la información sin necesidad de una implementación desde cero.

Posibilidades de generalización de la solución:

- Sería necesario definir una especie de diccionario de etiquetas para acceder con mayor facilidad a la información de las páginas de la Universidad.

Calidad de los resultados obtenidos desde el punto de vista de la información entregada al usuario.

- Se puede mejorar la calidad de los resultados debido a que aparecen muy diversos a pesar de que deben hacer referencia a la misma información. Por ejemplo, en la información de contacto de profesores, hay distintos tipos de etiquetas para el correo electrónico y para el número de teléfono.

Problemas encontrados:

- Se encuentra mucha diversidad en los componentes de software y librerías que se deben utilizar, causando conflictos de incompatibilidad en varias ocasiones.
- Para el segundo punto, la manera en que cada portal utilizaba las etiquetas, a pesar de estar estandarizadas. Sobre todo, en los formatos de categoría de la noticia y fecha de publicación.
- Se encuentra dificultad en realizar un barrido para obtener la información de los profesores, dado que la estructura en cada página web por facultad o por departamento no es genérica, aunque pareciese idéntica no hay atributos o tags de html que se puedan reutilizar en la búsqueda, y que pueda aprovechar el framework de scrapy al máximo. Si tuviese la misma estructura de tags facilitaría en gran medida la búsqueda y obtención de la información.
- En algunos casos información como correos están mal escritos por lo que una selección o búsqueda de información muy estricta por REGEX dejaría por fuera esta información.

Mejoras posibles:

- Para el segundo punto, la implementación de xquery se hizo complicada, por lo que se sugiere para una próxima entrega, utilizar componentes disponibles más robustos y funcionales como XPath o feedparser

Retos por resolver:

- Entregar un filtrado de xml más robusto, en la medida de que hubo dificultad en la implementación de xquery.
- Tener una máquina virtual que no obstaculice la implementación de la solución. La instalación de los componentes y librerías requeridos, garantizando que no haya conflictos y que funcione para toda la lógica que debe implementarse, es una tarea que consumió mucho tiempo y es posible que se repita si los requerimientos para próximos talleres llegan a ser diferentes.

Posibles extensiones de valor agregado.

- Que el usuario tenga la posibilidad de generar un filtrado antes de solicitar la información, lo que podría ayudar disminuir la cantidad de elementos que trae el *scrapy*, lo que a su vez disminuiría la carga de información que hay que procesar.
- Permitirle al usuario escoger qué *feeds rss* desea consultar
- Entregar información categorizada de los feeds rss. Por ejemplo, entregar información solamente de deportes, de todos los feeds a los que se les haya solicitado información. Se puede hacer con etiquetas existentes, leyendo el titular o el link (Como en el caso del *feed* de la BBC)