Introducción

En este informe explico brevemente mi proceso para completar la etapa individual del laboratorio 3 por medio de la metodología elegida: una mezcla de ChatGPT, Documentación y Stack Overflow. Luego paso a expandir sobre qué problemas surgieron y como estos recursos me sivieron para solucionarlos.

Método

Elegí trabajar como usualmente lo hago: leo documentación introductoria sobre las tecnologías / herramientas que no conozco y trato de tener un mapa mental de "lo que no sé" y luego voy llenando esos "huecos" según los problemas que van sugiendo.

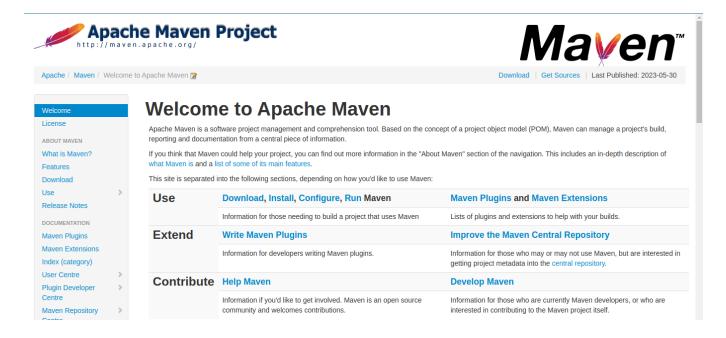
Recientemente empecé a usar ChatGPT para encontrar información conceptual(no tanto para generar código) más rápido que a través de buscadores como Google.

Las herramientas que usé fueron:

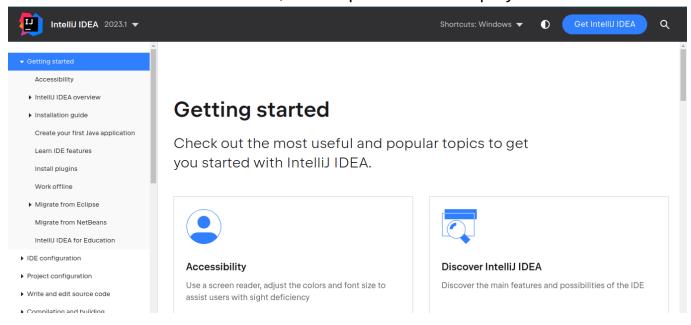
- Inteligencia artificial(ChatGPT) para preguntas conceptuales.
- Documentación para ejemplos de código.
- Documentación para how-to relacionado a herramientas(IDE, Build Tool).
- StackOverflow para preguntas sobre mensajes de error y ChatGPT en su defecto.

Proceso

En un principio anticipé que instalar spark incluiría, además, otras dependencias de Spark y la librería de JSON(dependencias de Spark principalmente) así que busqué sobre **Maven**, una herramienta para administrar proyectos en Java.



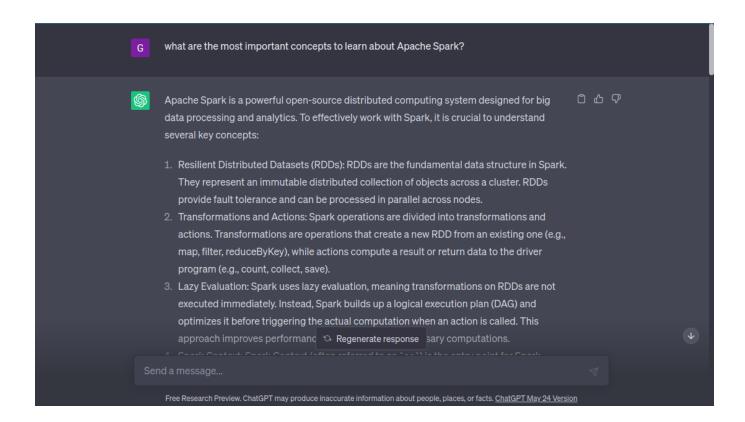
También sabía sobre IntelliJ Idea, un IDE para desarrollar proyectos en Java.



Entre sus features(listados en su página) confirmé que incluía soporte para Maven. Así que decidí usar el IDE.

Conocimiento básico de Spark

No tenía idea de spark, así que lo primero que hice fue pedirle a ChatGPT que me resuma los principales conceptos. La respuesta fue la siguiente:



Instalación de spark

En la documentación de IntelliJ leí como manejar un proyecto Maven desde el IDE, que básicamente consiste en declarar las dependencias del proyecto a un archivo pom.xml. Este archivo contiene los datos necesarios sobre el proyecto para que Maven pueda instalar las dependencias necesarias y compilar nuestro código.

Uso de Spark en el contexto del lab anterior

El lab anterior realizamos (al menos en la implementación de nuestro grupo) tareas de forma **secuencial**. Mi objetivo fue entonces hacer uso de map y reduce para realizar estas tareas de forma paralelizada.

Ejemplos

Para esto usé principalmente ejemplos de la <u>página sobre RDD's</u> y nociones sobre map y reduce que ya tenía.

Map

Por ejemplo, en las requests y parsing de los Feeds:

```
JavaRDD<SimpleSubscription> rSubList =
sc.parallelize(subscriptions.getSubscriptionList());
var feeds = rSubList.flatMap((simpleSubscription -> {
        List<FunctionO<Tuple2<Feed, String>>> frs = new ArrayList<>
();
        for (int j = 0, szj =
simpleSubscription.getUrlParametersSize(); j < szj; j++) {</pre>
                final int i = j;
                frs.add(() -> {
                        try {
                                 return new
Tuple2(simpleSubscription.parse(i), null);
                        } catch (Exception e) { // Omito todos los
casos de error
                                 return new Tuple2(null,
                                                 "Error while getting
feed in "
simpleSubscription.getFormattedUrlForParameter(i));
                });
        }
        return frs.iterator();
})).mapToPair(Function0::apply);
```

En el snippet de arriba vemos como:

- Usamos nuestra lista de subscripciones simples para conseguir una estructura de datos propia de Spark: RDD. Esta estructura es inmutable y es usada por el framework para distribuir computo a traves de distintos CPU's o incluso distintas computadoras.
- A partir de esta lista, aplicamos un map para obtener una lista de funciones
 "fetcher"
- Paralelizamos estos "fetchers" para obtener los feeds en simultaneo, usando otro tipo de map

Reduce

En el siguiente snippet podemos ver un ejemplo de como introduje reduce para sumar las frecuencias de cada entidad nombrada. En este caso usamos reduceByKey que reduce todos los elementos de la colección con la misma clave a una única tupla clave-valor. Usamos este snippet para conseguir la frecuencia global de cada named entity.

```
parsedFeeds.flatMap(feed -> {
        if(feed.getArticleList() != null) {
                return feed.getArticleList().iterator();
        } else {
                return Collections.emptyIterator();
}).flatMap(article -> {
        Heuristic heur = new QuickHeuristic();
        article.computeNamedEntities(heur);
        if(article.getNamedEntityList() != null) {
                return article.getNamedEntityList().iterator();
        } else {
                return Collections.emptyIterator();
        }
}).filter(Objects::nonNull).mapToPair(namedEntity -> new Tuple2<>
(namedEntity.getName(), namedEntity)).reduceByKey((n1, n2) -> {
        var n = new NamedEntity(n1.getName(), n1.getCategory(),
n1.getFrequency() + n1.getFrequency());
        n.setTheme(n1.getTheme());
        return n;
}).map(Tuple2::_2).foreach(namedEntity -> {
        System.out.println(namedEntity.getName());
        System.out.println(namedEntity.getFrequency());
        System.out.println(namedEntity.getCategory());
        System.out.println(namedEntity.getTheme());
        System.out.println(namedEntity.getClass().toString());
        System.out.println("----");
});
```

Integración de OOP con map-reduce y Spark

A pesar de ser posible escribir código más conciso usando map-reduce en Spark, se presentan ciertas dificultades al integrar con OOP. Principalmente en torno al

encapsulamiento. Si la naturaleza de cierto comportamiento dentro de una de nuestras clases es secuencial, entonces no tenemos otra que:

- Replicar este comportamiento(y el estado del que depende!) fuera para poder usar Spark sobre su estado.
- Cambiar la implementación de la clase

Esto puede llegar a ser un problema si no controlamos o entendemos gran parte de un proyecto, limitando las mejoras que podemos hacer usando paralelismo.

Conclusiones

Trabajar con Spark en Java puede tener sus desafíos a la hora de integrar código orientado a objetos que haga uso de paralelismo. De todas formas, usando la documentación para obtener ejemplos y modelos predictivos para sintetizar información rapidamente esta tarea se hace posible y -al menos durante este laboratorio- llevadera.