INFORME: Actores en Scala

Integrantes:

- **Francisco Joray**
- **Matias Valle**
- **Marcos Cravero**

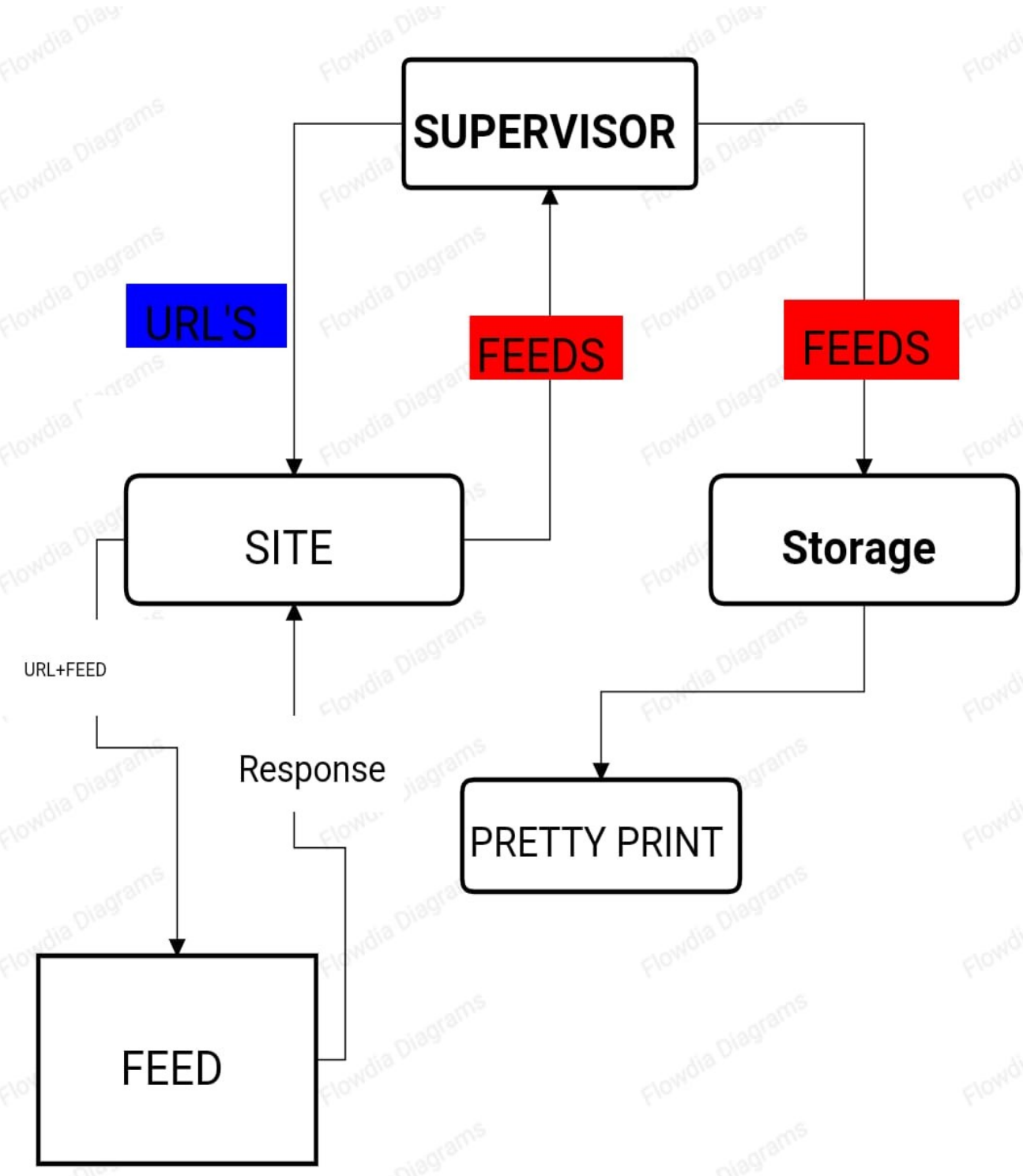
Desarrollo del proyecto

Actores involucrados en el problema

Desarrallamos un sistema el cual esta gobernado por un “Actor Supervisor”, que se comunica(enviando y reciendo mensaje) con sub-actores, “Actor Storage”, “Actor Site” y “Actor Feed”

Responsabilidades de los agentes encontrados

El sistema esta dirigido por el “Supervisor”, que se encarga de iniciar el proceso de construccion del feed. Este actor recibe un mensaje del main con los datos de los feeds a subscribirse y comienza su funcion. Una vez recibido estos datos, realiza un “ask”(pregunta a un actor) hacia el actor “site”, que es quien se encarga de la administracion de los feeds a subscribirse.Una vez recibe el primer feed, envia una pregunta hacia un actor “feed”, que es quien se encarga de obtener la respuesta http del enlace al que nos queremos subscribir.Esto ocurre con todos los feeds de un mismo enlace al cual nos queremos subscribir. Una vez terminado con todos los feeds a subscribirse, el actor “site” le responde el mensaje al supervisor con estos feeds.
Ahora que tenemos su respuesta, el supervisor se encarga de pasar este feed,mediante un mensaje, al actor “Storage” que es el que se va a encargar de almacenar este feed en una variable para luego imprimirlo por pantalla.
Una vez vez se terminen de pasar todos los feeds a los que el usuario se quiere subscribir, esperaremos 10 segundos para que se haga todo este proceso y mandaremos al actor “Supervisor” un mensaje que detenga la ejecucion. Con esto nos aseguramos que todo actor termine su trabajo, y si no lo termino, al deterner el actor supervisor, se detengan el resto de sus sub-actores.


SUPERVISOR
URL'S
FEEDS
FEEDS
SITE
Storage
URL+FEED
Response
PRETTY PRINT
FEED

PREGUNTAS:

- Si quisieran extender el sistema para soportar el conteo de entidades nombradas del laboratorio 2, ¿qué parte de la arquitectura deberían modificar?

  Agregariamos otro actor que se dedique al conteo de entidades nombradas. Con este, podemos modificar el supervisor para que se comunique con este nuevo actor que se encargaria de contar las entidades. Usamos un "ask-pattern" para mandarle como mensaje a este nuevo actor el feed y este nos responderia con las entidades nombradas en el feed y ya luego podrimos imprimir la lista de ellas.

- Si quisieran exportar los datos (ítems) de las suscripciones a archivos de texto (en lugar de imprimirlas por pantalla):
  1. ¿Qué tipo de patron de interaccion creen que les serviría y por qué? (hint: es mejor acumular todo los items antes de guardar nada).

Podriamos tener una variable en el actor supervisor que se encargue de ir recopilando todos los feeds y al final escribirlo en un archivo de texto a todos los datos.

Podemos importar "import java.io._" de java que tiene funciones para escribir en archivos, en ella podemos crear un nuevo archivo de texto creando un nuevo objeto de la clase "PrintWritter" y luego con el metodo "pw.write("mi texto")" podemos escribir lo que queramos dentro del archivo.

  2. ¿Dónde deberían utilizar dicho patrón si quisieran acumular todos los datos totales? ¿Y si lo quisieran hacer por sitio?

En caso de hacerlo por sitio, lo unico que deberiamos hacer es cambiar el metodo de imprimir por pantalla por un metodo que nos imprima en un archivo de texto.
Si en cambio, queremos escribir los datos totales, basta hacer lo que dijimos antes.

- ¿Qué problema trae implementar este sistema de manera síncrona?

  Implementar este sistema de manera sincrona nos evitaria poder subscribir varios feeds en diferentes hilos, ya que, le mandariamos un feed a la vez para que haga todo el proceso. En cambio al poder hacerlo de manera asincrona, podemos ir subscribiendo diferentes feeds en diferentes hilos en el mismo momento.

- ¿Qué les asegura el sistema de pasaje de mensajes y cómo se diferencia con un semáforo/mutex?
  Este sistema de pasaje de mensajes nos asegura que el sistema no se quedara esperando un mensaje que puede nunca llegar, es decir, una vez que enviamos un mensaje, el actor no se queda esperando a recibir una respuesta, puede ir a checkear su "mailbox" y revisar que otros mensajes tiene y seguir realizando su tarea, mientras que otros actores cumplen tambien con su pequeña tarea de manera que cada uno no depende de otro actor.
  Recordemos que cada actor se activa al recibir un mensaje, mientras no tengan mensaje, estos no estan activos.
  A diferencia de un semaforo, el pasaje de mensaje no espera a que ocurra un evento para avanzar, cada actor actua de manera asincrona.

Puntos Estrella

1) Para este punto, lo que hicimos fue agregar en cada enlace de un feed a subscribirse, un parametro "UrlType" que nos dira el parser a usar dependiendo la direccion url pasada. Con esto seguimos el mismo proceso que en la rama principal del proyecto, diferenciandose en que la respuesta http pedida tendra formato json o rss, dependiendo el feed. Para el parseo de estas nuevas direcciones usamos el parseador que programamos para el proyecto 2 y aprovechamos del re-uso de codigo ya probado y verificado.

2 y 4) Para que el actor Supervisor sepa cuántas suscripciones quedan aún por procesar, agregamos un contador que se va decrementando a medida que las respuestas de los distintos actores Site van llegando. Dicho contador se setea inicialmente con el largo del archivo de suscripciones ingresado, a través de un mensaje Supervisor.SetCounter(counter: Int). Cuando llega la última respuesta (i.e., cuando el contador vale 0), enviamos al actor Storage el mensaje Storage.GetNEs para que llame al modelo de reconocimiento y conteo de entidades nombradas, imprima las primeras 30 por pantalla y finalmente le envíe al Supervisor la orden de detenerse.