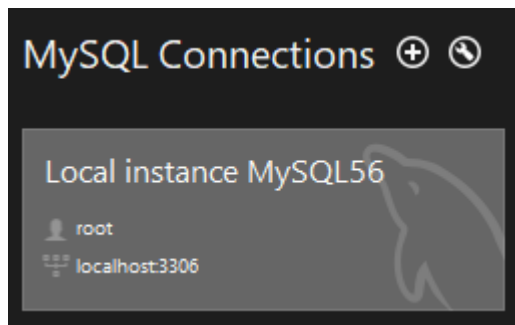


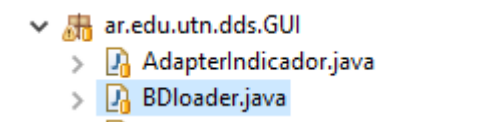
Diseño de Sistemas	Curso: K-3051 – Año 2017
Trabajo Práctico: ¿Dónde Invierto?	Grupo: 05 – Versión 1.0

Configuración de la Base de Datos:

La Base de Datos utilizada es MySQL, se debe definir el schema dds2017.



Ejecutar la clase #Bdloader.java, para iniciar la carga de datos en la DB.



- 1) Para introducir múltiples fuentes optamos por la utilización de un **Crawler**, el cual lo implementaríamos a nuestro proyecto mediante el patrón de diseño llamado “**Adapter**”.

¿Qué es un Crawler?

Un **Crawler** (también conocido como Recolector, Web spider o robot del Web) es un programa que hojea el World Wide Web de una manera metódica, automatizada. Otros nombres usados con menor frecuencia para los crawlers de la Web son hormigas, controladores paso a paso automáticos, bots, y gusanos (Kobayashi y Takeda, 2000). Los crawlers hacen uso de la estructura de grafo de la web para moverse de página a página, mediante una implementación del algoritmo de búsqueda breadth-first.

WebRACE (Zeinalipour-Yazti y Dikaiakos, 2002) es un módulo de crawling implementado en Java, y usado como parte de un sistema más genérico llamado eRACE. El sistema recibe peticiones de los usuarios para descargar Web pages, así que los actos del crawler en parte funcionan como proxy server inteligente. El sistema también maneja los pedidos o “suscripciones” a los Web pages que deben ser supervisados: cuando las páginas cambian, deben ser descargadas por el crawler y el suscriptor debe ser notificado. La característica más excepcional de WebRACE es que, mientras que la mayoría de los crawlers comienzan con un sistema de la “semilla” URLs, WebRACE está recibiendo continuamente URLs nuevos para comenzar.

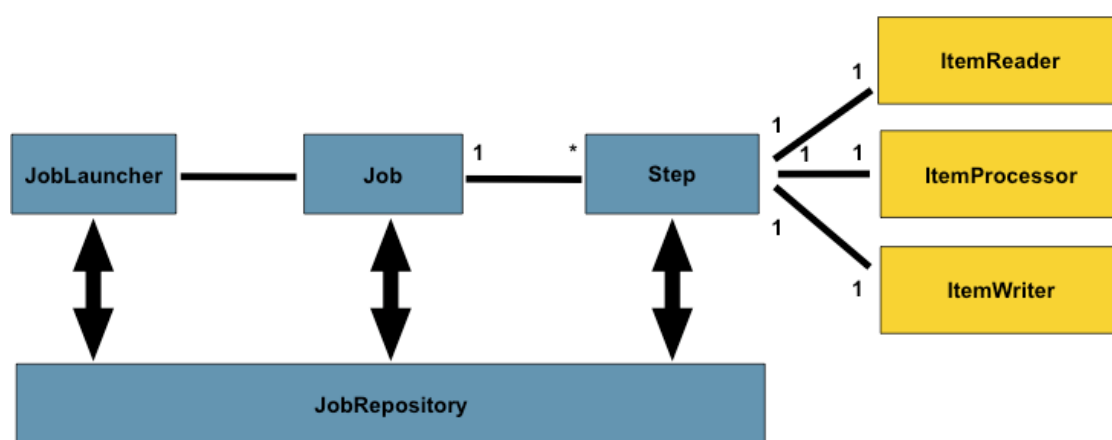
- 2) Para realizar el proceso Batch offline, encargado de la carga de datos, se ejecutará todos los días a las 24:00hs. Para dicho proceso se utilizará una herramienta llamada Spring Batch.

Diseño de Sistemas	Curso: K-3051 – Año 2017
Trabajo Práctico: ¿Dónde Invierto?	Grupo: 05 – Versión 1.0

Los **procesos batch** son aquellos programas que se lanzan generalmente de manera programada y que no requieren ningún tipo de intervención humana. Suelen ser procesos relativamente pesados, que tratan una gran cantidad de información, por lo que normalmente se ejecutan en horarios con baja carga de trabajo para no influir en el entorno transaccional.

Spring Batch es un framework ligero enfocado específicamente en la creación de procesos batch. Además de marcar unas directrices para el diseño de procesos, Spring Batch proporciona una gran cantidad de componentes que intentan dar soporte a las diferentes necesidades que suelen surgir a la hora de crear estos programas: trazas, transaccionalidad, contingencia, estadísticas, paralelismo, particionamiento, lectura y escritura de datos, etc.

Componentes Principales:



JobRepository: es el componente encargado de la persistencia de metadatos relativos a los procesos tales como procesos en curso o estados de las ejecuciones.

JobLauncher: es el componente encargado de lanzar los procesos suministrando los parámetros de entrada deseados.

Job: es la representación del proceso. Un proceso, a su vez, es un contenedor de pasos (steps).

Step: es un elemento independiente dentro de un Job (un proceso) que representa una de las fases de las que está compuesto dicho proceso. Un proceso (Job) debe tener, al menos, un step.

- 3) División de la aplicación en módulos de servicios funcionalmente relacionados que puedan ser mantenidos y escalados conjuntamente. La idea es tener varios componentes independientes e intercomunicados. La falla de uno de los componentes no debe porque afectar al funcionamiento del sistema. Cuanto más desacoplados estén los módulos entre sí, mayor flexibilidad se va a tener para escalar los componentes independientemente. Al tener los módulos separados, el código también lo está por lo que se puede tener grupos especializados de cada módulo o componente y no necesitan preocuparse por los demás componentes; esto conlleva también a una facilitación a la hora de escalar la base de datos. Se pueden utilizar diferentes bases de datos para cada uno de los componentes.

Diseño de Sistemas	Curso: K-3051 – Año 2017
Trabajo Práctico: ¿Dónde Invierto?	Grupo: 05 – Versión 1.0

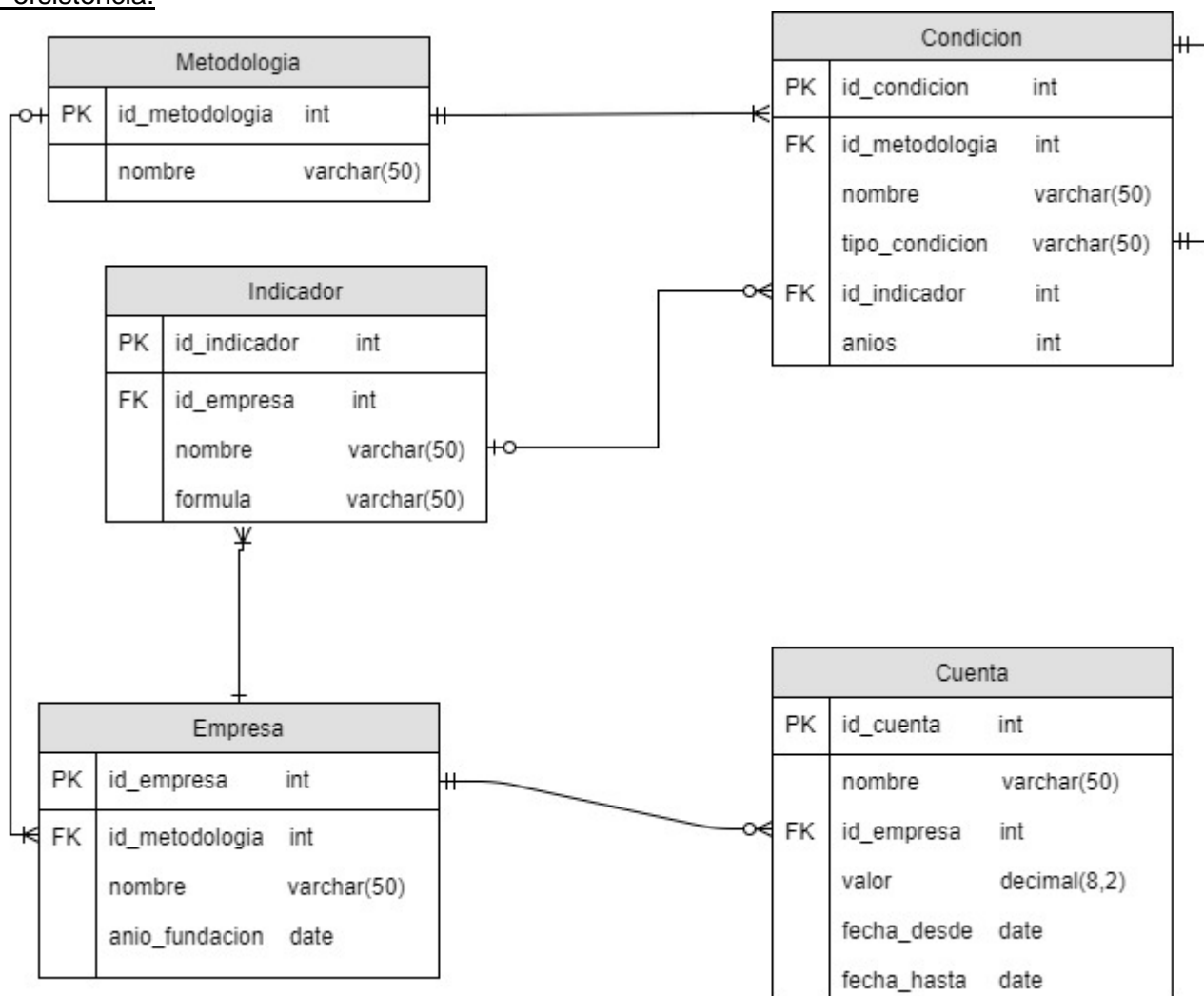
Cache distribuida: esto ayuda a tomar los datos directamente de la memoria cache de donde se esté corriendo el proyecto en vez de utilizar queries a la base de datos o archivos del sistema que son muy lentos. A su vez, ayuda a la escalabilidad lineal el hecho de agregar más tupas o registros a la cache. El fin de utilizar una cache es minimizar la cantidad de trabajo que el sistema realiza y hacer que el mismo responda más rápido ante cada petición de los datos.

La utilización de "**Entradas-Salidas No Bloqueantes**" permite a los programadores obtener gran performance en el procesamiento de los datos y brinda una mejor escalabilidad. Las IO no bloqueantes se pueden implementar por medio del paquete de java NIO permitiendo acercarnos al manejo del sistema operativo. Este feature permite tener múltiples conexiones HTTP's manejadas por un único hilo; la única limitante en cuanto a número de conexiones es la cantidad de memoria disponible para el stack.

No almacenar Estados en el nivel de la aplicación: almacenar estados conlleva mucho espacio y cuidado. Si es necesario realizar uno, el mismo debe ser hecho en la base de datos para que cada sesión pueda tener acceso a dicho estado almacenado.

Paralelizar las tareas: esto es posible mediante la utilización de hilos, java nos brinda la posibilidad de realizar Fork/Join mediante un framework para poder tener "multiprocesamiento". La idea es utilizar la mayor cantidad de hilos con tareas que puedan ser ejecutadas en paralelo para poder incrementar la performance del sistema

Persistencia:



Para la persistencia de datos utilizamos las anotaciones de “Hibernate” para cada columna. Al momento de realizar la persistencia de una herencia implementada a través de “Strategy” en objetos, decidimos realizar dicha persistencia por medio de una “Single Table”.

La utilización del mapeo “Single Table” nos permite realizar queries polimórficos. A cambio de esto debemos permitir el costo de poseer un campo nulo en el peor de los casos. En base a esto y a que si optáramos por otro estilo de mapeo íbamos a tener más tablas implicando más accesos para acceder a los datos persistidos y, a su vez, tener campos repetidos

Diagrama de Clases:

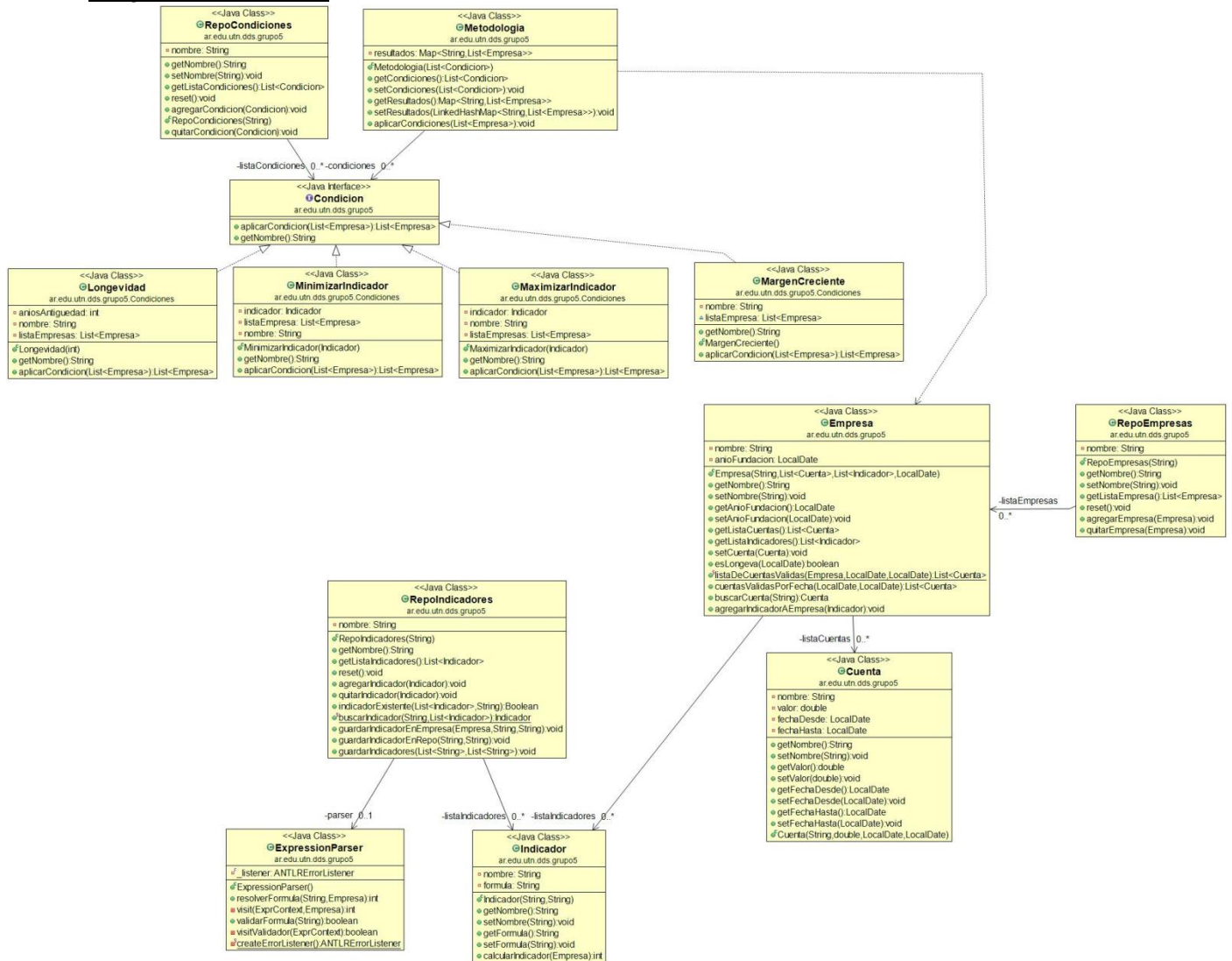
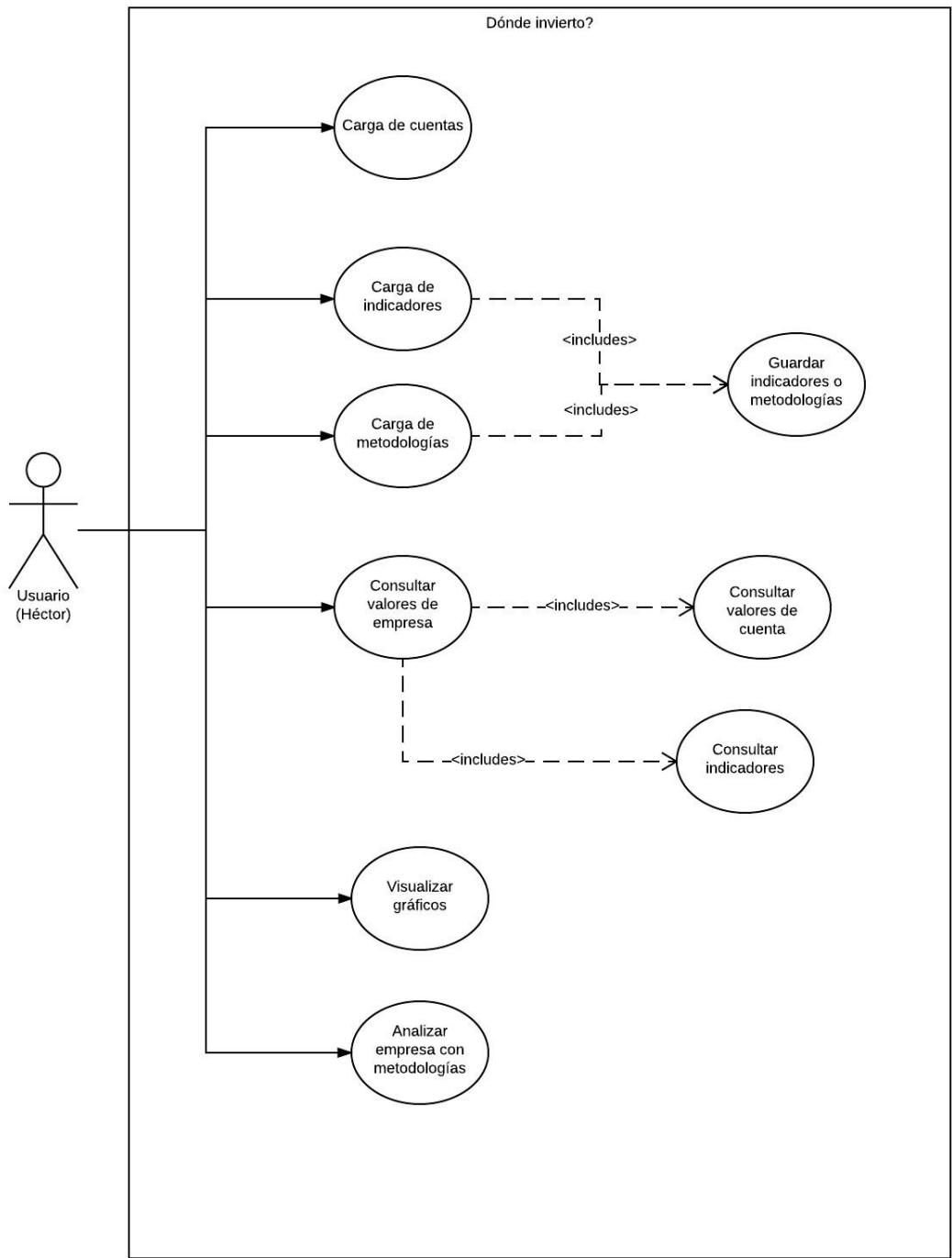


Diagrama de Casos de Uso:



Bocetos:

Carga de cuenta

Importar archivo:

Consulta de empresa

Seleccione la empresa:

Facebook

Empresa2

Fecha desde:

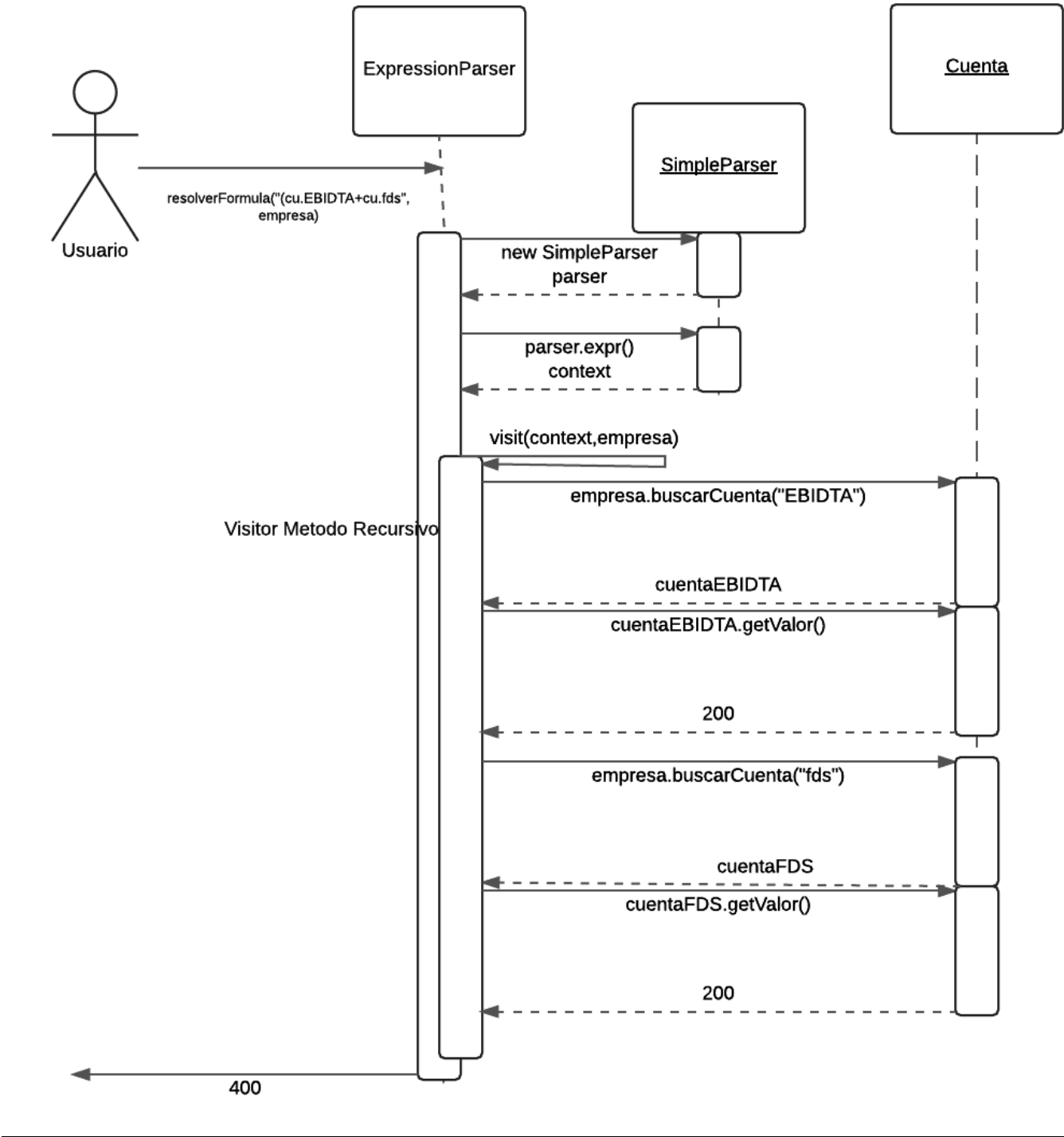
Fecha hasta:

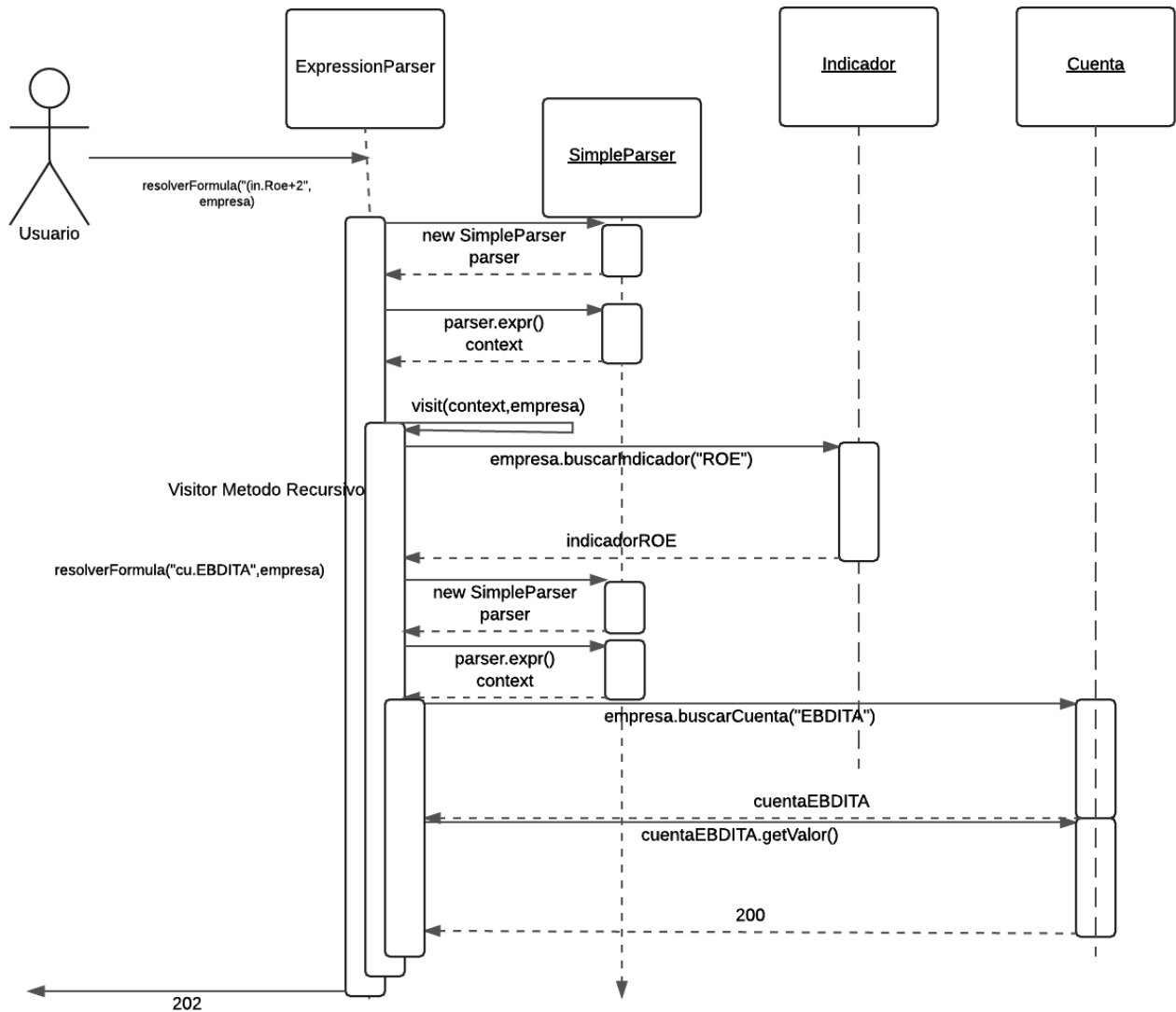
Valores de cuenta

Facebook

Cuenta	Valor
EBITDA	\$14.870.000.000
Ingreso neto	\$ 4.273.000.000

GRAMA DE SECUENCIA



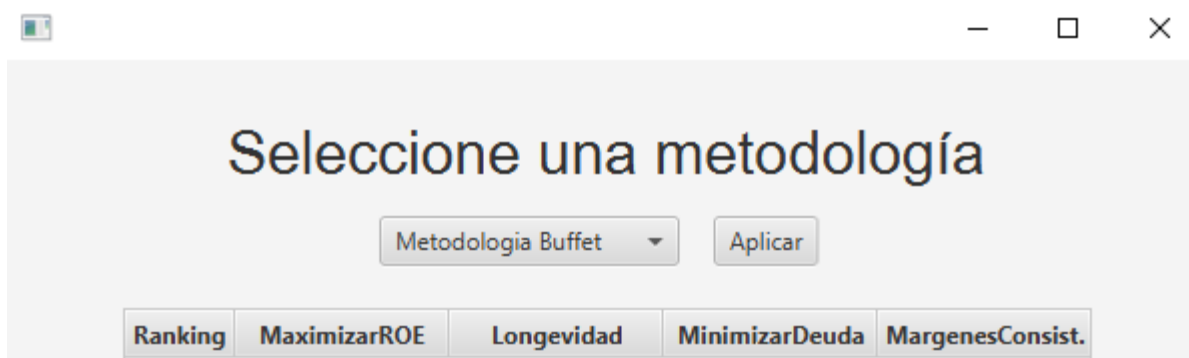


Diseño de Sistemas	Curso: K-3051 – Año 2017
Trabajo Práctico: ¿Dónde Invierto?	Grupo: 05 – Versión 1.0

Decisiones de diseño

Las metodologías son cargadas con las condiciones correspondientes, dentro de una lista. La misma se aplica a una lista de empresas, aplicando cada condición a la lista.

Cada una de las condiciones devuelve una lista ordenada según el criterio de la condición. Estas listas ordenadas se agregan una a una a un Hashmap que posee como clave el nombre de la condición, y como valor la lista ordenada. Al final, dichas listas se mostrarán una al lado de la otra con la clave como título de la columna.



Seleccione una metodología

Metodologia Buffet ▼ Aplicar

Ranking MaximizarROE Longevidad MinimizarDeuda MargenesConsist.