Trabajo Práctico Especial 2
Reclamos Urbanos

28 de Mayo de 2025

Créditos Ilustración: Van311

Objetivo

Diseñar e implementar una aplicación de consola que utilice el **modelo de programación MapReduce** junto con el framework **Hazelcast** para el procesamiento de **reclamos urbanos**, basado en datos reales.

Para este trabajo se busca poder procesar datos de reclamos urbanos de las ciudades de **Nueva York, EEUU** y **Chicago, EEUU**.

Los datos son extraídos de los respectivos portales de gobierno en formato CSV.

Descripción Funcional

A continuación se lista el ejemplo de uso que se busca para la aplicación: procesar los datos de reclamos urbanos de las ciudades de Nueva York y Chicago. Sin embargo, es importante recordar que la mayor parte de la implementación no debe estar atada a la realidad de los ejemplos de uso. **Por ejemplo, las estadísticas serán las que la aplicación obtenga en ejecución a partir de los archivos de reclamos y de tipos de reclamos y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de reclamos de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presentan a continuación.

Datos de **reclamos** de Nueva York, a partir de ahora `serviceRequestsNYC.csv`

- **Origen (No Descargarlo):** 311 Service Requests from 2010 to Present | NYC Open Data
- **Descarga:**
  - `/afs/it.itba.edu.ar/pub/pod/v2/serviceRequestsNYC.csv.zip`
  - `/afs/it.itba.edu.ar/pub/pod/v2/serviceRequestsNYC.csv`
- **Cantidad de registros: 32.180.598**
- **Campos:**
  - `Unique Key`: Identificador Único (Entero)
  - `Created Date`: Fecha de Creación (String con formato `yyyy-MM-dd HH:mm:ss`)
  - `Agency Name`: Nombre de la Agencia (String)

- **`Complaint Type`**: Tipo (String)
- **`Incident Address`**: Dirección (String)
- **`Status`**: Estado (String)
- **`Borough`**: Nombre del Barrio (String)
- **`Latitude`**: Latitud (Double)
- **`Longitude`**: Longitud (Double)

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, **cada línea representa un reclamo** conteniendo los datos de cada uno de los campos, separados por ";". Por ejemplo:

```
Unique Key;Created Date;Agency Name;Complaint Type;Incident
Address;Status;Borough;Latitude;Longitude
61754152;2024-07-09 12:08:00;DEP;Water System;886
BROADWAY;Closed;BROOKLYN;40.698101841859376;-73.93725672896107
61754508;2024-07-09 12:01:06;HPD;APPLIANCE;204 EAST   96
STREET;Closed;BROOKLYN;40.66153484338518;-73.92196893141944
61754534;2024-07-09 23:54:14;HPD;ELECTRIC;460 WEST  149
STREET;Closed;MANHATTAN;40.82743255982856;-73.94495408753242
...
```

Datos de **tipos de reclamos** de Nueva York, a partir de ahora `serviceTypesNYC.csv`

- **Descarga: `/afs/it.itba.edu.ar/pub/pod/serviceTypesNYC.csv`**
- **Cantidad de registros: 337**
- **Campo:**
  - **`Complaint Type`**: Tipo (String)

El archivo se compone de una primera línea de encabezado con el título del campo. De la segunda línea en adelante, **cada línea representa un tipo de reclamo**. Por ejemplo:

```
Complaint Type
Smoking
Illegal Tree Damage
Highway Sign - Missing
...
```

Datos de **reclamos** de Chicago, a partir de ahora `serviceRequestsCHI.csv`

- **Origen (No Descargarlo):** [311 Service Requests | City of Chicago | Data Portal]
- **Descarga:**
  - **`/afs/it.itba.edu.ar/pub/pod/serviceRequestsCHI.csv.zip`**
  - **`/afs/it.itba.edu.ar/pub/pod/serviceRequestsCHI.csv`**
- **Cantidad de registros: 11.100.190**
- **Campos:**
  - **`SR_NUMBER`**: Identificador Único (Alfanumérico)
  - **`SR_SHORT_CODE`**: Acrónimo del Tipo (String)
  - **`OWNER_DEPARTMENT`**: Nombre de la Agencia (String)

- **`STATUS`**: Estado (String)
- **`CREATED_DATE`**: Fecha de Creación (String con formato yyyy-MM-dd HH:mm:ss)
- **`STREET_NUMBER`**: Altura de la dirección (Entero)
- **`STREET_DIRECTION`**: Punto cardinal de la dirección (“N”, “S”, “E” o “W”)
- **`STREET_NAME`**: Nombre de la dirección (String)
- **`STREET_TYPE`**: Tipo de calle de la dirección (String), por ejemplo `ST`, `AVE`, `RD`, etc.
- **`COMMUNITY_AREA`**: Nombre del barrio (String)
- **`LATITUDE`**: Latitud (Double)
- **`LONGITUDE`**: Longitud (Double)

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, **cada línea representa un reclamo** conteniendo los datos de cada uno de los campos, separados por “;”. Por ejemplo:

```
SR_NUMBER;SR_SHORT_CODE;OWNER_DEPARTMENT;STATUS;CREATED_DATE;STREET_NUMBER;STREET
_DIRECTION;STREET_NAME;STREET_TYPE;COMMUNITY_AREA;LATITUDE;LONGITUDE
SR20-05496057;WCA2;DWM - Department of Water Management;Completed;2020-11-20
12:53:10;2851;N;ELSTON;AVE;AVONDALE;41.933352785;-87.689126234
SR20-05496117;SCC;Streets and Sanitation;Completed;2020-11-20
12:59:25;508;N;RACINE;AVE;WEST TOWN;41.89134445;-87.65756301
SR20-05496129;CHECKFOR;DWM - Department of Water Management;Completed;2020-11-20
13:00:51;2819;N;FRANCISCO;AVE;AVONDALE;41.932742873;-87.699796773
...
```

Datos de **tipos de reclamos** de Chicago, a partir de ahora **`serviceTypesCHI.csv`**

- **Descarga: `/afs/it.itba.edu.ar/pub/pod/serviceTypesCHI.csv`**
- **Cantidad de registros: 106**
- **Campos:**
  - **`SR_SHORT_CODE`**: Acrónimo del Tipo (String)
  - **`SR_TYPE`**: Tipo (String)

El archivo se compone de una primera línea de encabezado con los títulos de cada campo. De la segunda línea en adelante, **cada línea representa un tipo de reclamo**. Por ejemplo:

```
SR_SHORT_CODE;SR_TYPE
SED;Tree Planting Request
TNP;Ridesharing Complaint
SDW;Snow - Object/Dibs Removal Request
...
```

Requerimientos

La aplicación debe poder resolver un conjunto de consultas listadas más abajo. En cada una de ellas se indicará un ejemplo de invocación con un *script* propio para correr únicamente esa *query* con sus parámetros necesarios.

Cada corrida de la aplicación resuelve sólo una de las *queries* sobre los datos obtenidos a partir de los archivos CSV provistos en esa invocación (archivos CSV de reclamos y tipos de reclamos)

La respuesta a la *query* quedará en un archivo de salida CSV.

Para medir performance, se deberán escribir en otro archivo de salida los *timestamp* de los siguientes momentos:

- Inicio de la lectura de los archivos de entrada
- Fin de lectura de los archivos de entrada
- Inicio de un trabajo MapReduce
- Fin de un trabajo MapReduce (incluye la escritura del archivo de respuesta)

Todos estos momentos deben ser escritos en la salida luego de la respuesta con el timestamp en formato: `dd/mm/yyyy hh:mm:ss:xxxx` y deben ser claramente identificables.

Ejemplo del archivo de tiempos:

```
28/05/2025 14:43:09:0223 INFO  [main] Client (Client.java:76) - Inicio de la
lectura del archivo
28/05/2025 14:43:23:0011 INFO  [main] Client (Client.java:173) - Fin de lectura
del archivo
28/05/2025 14:43:23:0013 INFO  [main] Client (Client.java:87) - Inicio del
trabajo map/reduce
28/05/2025 14:43:23:0490 INFO  [main] Client (Client.java:166) - Fin del
trabajo map/reduce
```

Por ejemplo:

```
$> sh queryX.sh -Daddresses='xx.xx.xx.xx:XXXX;yy.yy.yy.yy:YYYY' -Dcity=XYZ
-DinPath=XX -DoutPath=YY [params]
```

donde:

- `queryX.sh` es el script que corre la query `X`.
- `-Daddresses` refiere a las direcciones IP de los nodos con sus puertos (una o más, separadas por punto y coma)
- `-Dcity` indica con qué dataset de ciudad se desea trabajar. Los únicos valores posibles son **`NYC`** y **`CHI`**.
- `-DinPath` indica el path donde están los archivos de entrada de reclamos y tipos de reclamos
- `-DoutPath` indica el path donde están ambos archivos de salida **`query1.csv`** y **`time1.txt`**.
- `[params]` son los parámetros extras que corresponden para algunas queries.

De esta forma,

```
$> sh query1.sh -Daddresses='10.6.0.1:5701;10.6.0.2:5701' -Dcity=NYC
-DinPath=/afs/it.itba.edu.ar/pub/pod/
-DoutPath=/afs/it.itba.edu.ar/pub/pod-write/
```

resuelve la *query* 1 a partir de los datos presentes en `/afs/it.itba.edu.ar/pub/pod/serviceRequestsNYC.csv` y `/afs/it.itba.edu.ar/pub/pod/serviceTypesNYC.csv` utilizando los nodos `10.6.0.1` y `10.6.0.2` para su procesamiento. Se crearán los archivos `/afs/it.itba.edu.ar/pub/pod-write/query1.csv` y

`/afs/it.itba.edu.ar/pub/pod-write/time1.txt` que contendrán respectivamente el resultado de la *query* y los *timestamp* de inicio y fin de la lectura del archivo y de los trabajos map/reduce.

De invocarse con `-Dcity=CHI` utilizará los datos presentes en `serviceRequestsCHI.csv` y `serviceTypesCHI.csv`

Query 1: Total de reclamos por tipo y agencia

Donde cada línea de la salida contenga, separados por ";" **el tipo**, la **agencia** y la cantidad **total de reclamos de ese tipo con esa agencia**.

El orden de impresión es **descendente** por **total de reclamos** y desempata **alfabético** por **tipo** y luego **alfabético** por **agencia**.

Sólo se deben listar **los tipos presentes en el archivo de tipos**.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `sh query1.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.`

- Salida de ejemplo para NYC:

```
type;agency;requests
Noise - Residential;NYPD;3148841
Blocked Driveway;NYPD;2780131
HEAT/HOT WATER;HPD;2780131
Illegal Parking;NYC311-PRD;12047
Illegal Parking;NYPD;12047
...
```

- Salida de ejemplo para CHI:

```
type;agency;requests
311 INFORMATION ONLY CALL;311 City Services;4132667
Aircraft Noise Complaint;Aviation;1926717
Garbage Cart Maintenance;Streets and Sanitation;298051
Graffiti Removal Request;Streets and Sanitation;298051
Rodent Baiting/Rat Complaint;Streets and Sanitation;298051
...
```

Query 2: Tipo de reclamo más popular por barrio y cuadrante

Donde cada línea de la salida contenga, separados por ";" el **barrio**, el **cuadrante** y el **tipo más popular** de ese barrio y cuadrante.

Un **cuadrante** consiste en una porción del espacio geográfico visto en forma de grilla. Tiene un tamaño fijo de **q grados** en latitud y longitud, donde q es un **parámetro adicional obligatorio**. Para conseguir el cuadrante correspondiente a una coordenada geográfica se debe obtener la parte entera de la división de latitud y longitud por q. Por ejemplo para un q de 0.1 (que cubre aproximadamente $11km^2$) una coordenada de NYC (40.698101841859376, -73.93725672896107) se corresponde con el cuadrante (406, -739). Para la coordenada de CHI (41.933352785, -87.689126234) el cuadrante correspondiente es el (419, -876).

El orden de impresión es **alfabético** por **barrio** y desempata **ascendente** por **latitud del cuadrante** y luego **ascendente** por **longitud del cuadrante**.

Sólo se deben listar **los tipos presentes en el archivo de tipos**.

- Parámetros adicionales: `q` real en (0, 1]

- Ejemplo de invocación: `sh query2.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dq=0.1`

- Salida de ejemplo para NYC:

```
neighbourhood;quadLat;quadLon;topType
...
BROOKLYN;407;-739;Lost Property
BROOKLYN;408;-739;Consumer Complaint
BROOKLYN;408;-737;Consumer Complaint
MANHATTAN;406;-741;Noise - Helicopter
MANHATTAN;406;-740;Consumer Complaint
MANHATTAN;406;-739;Lost Property
...
```

- Salida de ejemplo para CHI:

```
neighbourhood;quadLat;quadLon;topType
ALBANY PARK;418;-877;Graffiti Removal Request
ALBANY PARK;419;-878;Yard Waste Pick-Up Request
ALBANY PARK;419;-877;Graffiti Removal Request
ARCHER HEIGHTS;417;-878;Graffiti Removal Request
ARCHER HEIGHTS;418;-878;Graffiti Removal Request
...
```

Query 3: Media móvil de reclamos abiertos por agencia, año y mes

Donde cada línea de la salida contenga, separados por ";" la **agencia**, el **año**, el **mes** de ese año, y **la media móvil** (promedio móvil) de reclamos abiertos para esa agencia usando **una ventana de `w` meses** donde `w` es un parámetro adicional.

Los **reclamos abiertos** de NYC son aquellos que tienen un estado estado distinto a "`Closed`". En CHI son aquellos que tienen el estado "`Open`".

La **media móvil** de reclamos abiertos consiste en el promedio de reclamos abiertos de los últimos `w` meses incluyendo el mes actual (total de reclamos abiertos de los últimos `w` meses dividido por w).

La **ventana** sólo cubre meses del año actual. En los primeros meses del año si no hay suficientes datos para completar la ventana de tamaño `w`, la media móvil se calculará utilizando únicamente los meses disponibles. Por ejemplo para `w = 3` en un año, en Enero sólo hay un mes disponible por lo que la media móvil coincide con el total de reclamos abiertos de ese mes, en Febrero será el total de reclamos abiertos de Enero y Febrero dividido 2 y recién desde Marzo en adelante la media móvil será el total de reclamos de los últimos tres meses dividido 3. El promedio debe incluir los meses con 0 reclamos, es decir, si para un mes no hay reclamos abiertos, sumará 0 al total correspondiente.

El orden de impresión es **alfabético** por **agencia** y desempata **cronológico** por **año y** luego **cronológico** por **mes**.

Los valores de los promedios se deben truncar (no redondear) a dos decimales.

- Parámetros adicionales: `w` entero en [1, 12]

- Ejemplo de invocación: `sh query3.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=. -Dw=3`

- Salida de ejemplo para NYC:

```
agency;year;month;movingAvg
...
DSNY;2024;11;225.66
DSNY;2024;12;551.33
DSNY;2025;1;1986.00
EDC;2015;7;3.00
EDC;2015;8;1.00
EDC;2016;2;2.00
...
```

- Salida de ejemplo para CHI:

```
agency;year;month;movingAvg
Animal Care and Control;2020;4;3.00
Animal Care and Control;2020;5;1.50
Animal Care and Control;2020;12;2.66
...
Streets and Sanitation;2024;11;1688.66
Streets and Sanitation;2024;12;1796.50
Streets and Sanitation;2025;1;4310.00
```

Query 4: Porcentaje de tipos de reclamo por calle

Donde cada línea de la salida contenga, separados por ";" la **calle** y el **porcentaje de tipos de reclamo** en esa calle.

El **porcentaje de tipos de reclamo** en una calle consiste en el cociente entre la cantidad de tipos de reclamo distintos para una calle y el total de tipos de reclamo distintos de todas las calles de la ciudad, multiplicado por 100**.**

Se deben **listar únicamente las calles correspondientes al barrio indicado** por un parámetro adicional donde se debe reemplazar los guiones bajo "_" por espacios " ". Por ejemplo si el cliente recibe "STATEN_ISLAND" entonces tomarlo como "STATEN ISLAND".

Para identificar a una calle en NYC, usar el campo `Incident Address` y además:

- Si cumple la expresión `Número + Espacio + Texto` entonces `Texto` será el identificador (así se descarta el número que corresponde a la altura de la calle)
  Ejemplos:
  - `2230 UNIVERSITY AVENUE` corresponde a la calle `UNIVERSITY AVENUE`
  - `460 WEST  149 STREET` corresponde a la calle `WEST  149 STREET`
- Sino tomarlo sin cambios (ya que no hay altura para descartar)
  Ejemplos:
  - `BROADWAY`
  - `HENDRIX STREET`

Para identificar a una calle en CHI, usar los campos `STREET_DIRECTION`, `STREET_NAME` y `STREET_TYPE` de forma que si alguno de los tres es distinto entonces corresponde a una calle distinta.

Ejemplos de calles distintas entre ellas:

- `N;MICHIGAN;AVE` corresponde a la calle `N MICHIGAN AVE`
- `S;MICHIGAN;AVE` corresponde a la calle `S MICHIGAN AVE`
- `N;MICHIGAN;ST` corresponde a la calle `N MICHIGAN ST`

El orden de impresión es **descendente por porcentaje** y desempata **alfabético por calle**.
Los valores de los porcentajes se deben truncar (no redondear) a dos decimales.
Sólo se deben listar **los tipos presentes en el archivo CSV de tipos**.

- Parámetros adicionales: `neighbourhood` String

- Ejemplo de invocación: `sh query4.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.` **`-Dneighbourhood=MANHATTAN`**

- Salida de ejemplo para NYC:

```
street;typePercentage
BROADWAY;68.00%
2 AVENUE;66.15%
5 AVENUE;66.15%
AMSTERDAM AVENUE;58.15%
3 AVENUE;58.14%
...
```

- Ejemplo de invocación: `sh query4.sh -Daddresses='10.6.0.1:5701' -Dcity=CHI -DinPath=. -DoutPath=.` **`-Dneighbourhood=LOOP`**

- Salida de ejemplo para CHI:

```
street;typePercentage
S DEARBORN ST;77.77%
S WABASH AVE;75.24%
S WELLS ST;75.24%
S MICHIGAN AVE;73.69%
S CLARK ST;66.13%
...
```

Query 5: Total de reclamos resueltos YTD por agencia (Exclusivo para grupos de 4 integrantes)

Donde cada línea de la salida contenga, separados por ";" la **agencia**, el **año**, el **mes** de ese año, y la cantidad **total de reclamos resueltos *Year To Date*** (YTD) **para esa agencia**.

Los **reclamos resueltos** de NYC son aquellos que tienen el Status "`Closed`". En CHI son aquellos que tienen el estado "`Completed`".

El **total YTD de un mes** consiste en la suma de los totales desde el primer mes del año hasta ese mes inclusive. Por ejemplo, el total YTD de Enero de 2025 sólo contendrá el total de Enero de 2025. El YTD de Febrero de 2025 contendrá el total de Enero de 2025 y de Febrero de 2025. El YTD de Diciembre de 2025 contendrá el total de todos los meses de 2025.

El orden de impresión es **alfabético por agencia** y desempata **cronológico por año y mes**.

- Parámetros adicionales: Ninguno

- Ejemplo de invocación: `sh query5.sh -Daddresses='10.6.0.1:5701' -Dcity=NYC -DinPath=. -DoutPath=.`

- Salida de ejemplo para NYC:

```
agency;year;month;resolvedYTD
...
NYPD;2024;11;1402687
NYPD;2024;12;1402687
NYPD;2025;1;61324
OSE;2020;6;2758
OSE;2020;7;12748
OSE;2020;8;21097
...
```

- Salida de ejemplo para CHI:

```
agency;year;month;resolvedYTD
311 City Services;2019;2;3248
311 City Services;2019;3;37489
311 City Services;2019;4;73658
...
Streets and Sanitation;2024;11;435645
Streets and Sanitation;2024;12;478695
Streets and Sanitation;2025;1;12577
```

Hechos y Consideraciones

Para simplificar el desarrollo se pueden tomar los siguientes considerandos como ciertos:

- El formato de los archivos es correcto, no es necesario validar que los nombres de las columnas coincidan, ni que el tipo de dato de los valores de una columna es el esperado ni que la cantidad de columnas es la esperada
- Los valores de los parámetros de los clientes no contienen espacios
- Si el archivo de salida existe, reemplazar el contenido (no *appendear*)

Requisitos

El trabajo práctico debe realizarse en los mismos grupos formados para el primer trabajo práctico especial.

Se requiere implementar:

- Cada una de las consultas utilizando **uno o más *job*s MapReduce** que pueda correr en un ambiente distribuido utilizando un *grid* de Hazelcast.
- Los clientes indicados cada uno como una aplicación de consola diferente

Los componentes del *job*, clases del modelo, *tests* y el diseño de cada elemento del proyecto queda a criterio del grupo, pero debe estar enfocado en:

- Que funcione correctamente en un ambiente concurrente MapReduce en Hazelcast.
- Que sea eficiente para un gran volumen de datos, particularmente en tráfico de red.
- Mantener buenas prácticas de código como comentarios, reutilización, legibilidad y mantenibilidad.

Muy Importante:

➔ **Respetar EXACTAMENTE**
- **Los nombres de los *scripts* y sus parámetros** (Si se pide `-DinPath` no se aceptará `-DinputPath`, `-DpathEntrada`, etc.)
- **El orden y formato de salida enunciado, tanto para los clientes de consola como para los archivos CSV de salida**

➔ **En TODOS los `pom.xml` que entreguen deberán definir**
- El `artifactId` de acuerdo a la siguiente convención: "`tpe2-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<artifactId>`**`tpe1-g7-api`**`</artifactId>`**
- El `name` con la siguiente convención: "`tpe2-gX-Z`" donde X es el número de grupo y Z es `parent`, `api`, `server` o `client`. Por ejemplo: **`<name>`**`tpe1-g7-api`**`</name>`**
- La versión **3.8.6** de **`hazelcast-all`**

➔ **El nombre del cluster (`<group><name>`) y los nombres de las colecciones de Hazelcast** a utilizar en la implementación deben comenzar con "g" seguido del número de grupo. Por ej `g7` para así evitar conflictos con las colecciones y poder hacer pruebas de distintos alumnos en simultáneo utilizando la misma red.

Material a entregar

Cada grupo deberá subir al **Campus <u>UNICAMENTE UN ARCHIVO COMPACTADO</u>** conteniendo:

- El **código fuente** de la aplicación:
  - Utilizando el arquetipo de Maven utilizado en las clases
  - Con una correcta separación de las clases en los módulos *api*, *client* y *server*
  - Un README indicando cómo preparar el entorno a partir del código fuente para ejecutar la aplicación en un ambiente con varios nodos
  - El directorio oculto `.git/` donde se encuentra la historia de commits y modificaciones. Recordar que la descarga desde github.com no incluye el directorio .git/.
  - **<u>No se deben entregar los binarios.</u>** Recordar de ejecutar el comando `mvn clean` antes de la entrega.
- Un **documento breve** explicando:
  - **Cómo se diseñaron los componentes de cada trabajo MapReduce**, qué decisiones se tomaron y con qué objetivos. Además alguna alternativa de diseño que se evaluó y descartó, comentando el porqué.
  - **El análisis de los tiempos para la resolución de cada query**: En caso de poder, analizar la diferencia de tiempos de correr cada query aumentando la cantidad de nodos (hasta 5 nodos) en una red local. De no poder, intentar predecir cómo sería el comportamiento.
  - **Potenciales puntos de mejora y/o expansión**
  - **La comparación de los tiempos de una de las queries ejecutándose con y sin *Combiner*.**
  - **Otro análisis de tiempos de ejecución de las queries** utilizando algún otro elemento de optimización a elección por el grupo.

- Para todos los puntos anteriores, no olvidar de indicar el tamaño de los archivos utilizados como entrada para las pruebas (cantidad de registros).

Corrección

El trabajo no se considerará aprobado si:

- No se entregó el trabajo práctico en tiempo y forma
- Faltan algunos de los materiales solicitados en la sección anterior
- El código no compila utilizando Maven en consola (de acuerdo a lo especificado en el README a entregar)
- El cluster no inicia cuando se siguen los pasos del README
- Los clientes no corren al seguir los pasos del README

Si nada de esto se cumple, se procederá a la corrección donde se tomará en cuenta:

- Que los procesos y *queries* funcionen correctamente según las especificaciones dadas
- El resultado de las pruebas y lo discutido en el coloquio
- La aplicación de los temas vistos en clase: Concurrencia y Hazelcast
- La modularización, diseño, testeo y reutilización de código
- El contenido y desarrollo del informe

Uso de Git

Es obligatorio el uso de un repositorio Git para la resolución de este TPE. No se aceptarán entregas que utilicen un repositorio git con un único *commit* que consista en la totalidad del código a entregar.

Los distintos *commits* deben permitir ver la evolución individual del trabajo.

Muy importante: **los repositorios creados deben ser privados, solo visibles para los autores y la cátedra en caso de que se lo solicite específicamente.**

Cronograma

- **Presentación del Enunciado: miércoles 28/05**
- **Entrega del trabajo: Estará disponible hasta el sábado 14/06 a las 23:59** la actividad "TPE 2" localizada en la sección Contenido / Evaluaciones. En la misma deberán cargar el archivo compactado.
- **El día del coloquio será el miércoles 18/06**
- **El día del recuperatorio será el miércoles 25/06**
- **No se aceptarán entregas pasado el día y horario establecido como límite**

Dudas sobre el TPE

Las mismas deben volcarse en los **Debates** del Campus ITBA.

Recomendaciones

- **Para el parseo de argumentos del cliente como `-Dcity=NYC` consultar** `java.lang.System#getProperty(java.lang.String)`

- **Para la escritura de archivos .csv consultar** `java.nio.file.Files.write`
- **Para la lectura de archivos .csv consultar** `java.nio.file.Files.lines`
- **Para el manejo de números decimales:** `java.text.DecimalFormat` y `java.math.RoundingMode`
- **Para el manejo de fechas:** `java.time.format.DateTimeFormatter` y `java.time.YearMonth`