

# Índice

## Optimización

Consideraciones preliminares.....	2
Análisis de la aplicación.....	2
Solución a emplear.....	3
1º iteración: Mejora en el uso de memoria.....	3
2º iteración: Mejora en el tiempo de procesamiento.....	4
3º iteración: mejora en el tiempo de procesamiento.....	4
4º iteración: mejora en el tiempo de procesamiento.....	5
5º iteración: mejora en la funcionalidad.....	6

## Agregado de funcionalidad

Descripción.....	6
Lo que se aprendió.....	6

# Optimización de la aplicación

## Consideraciones preliminares

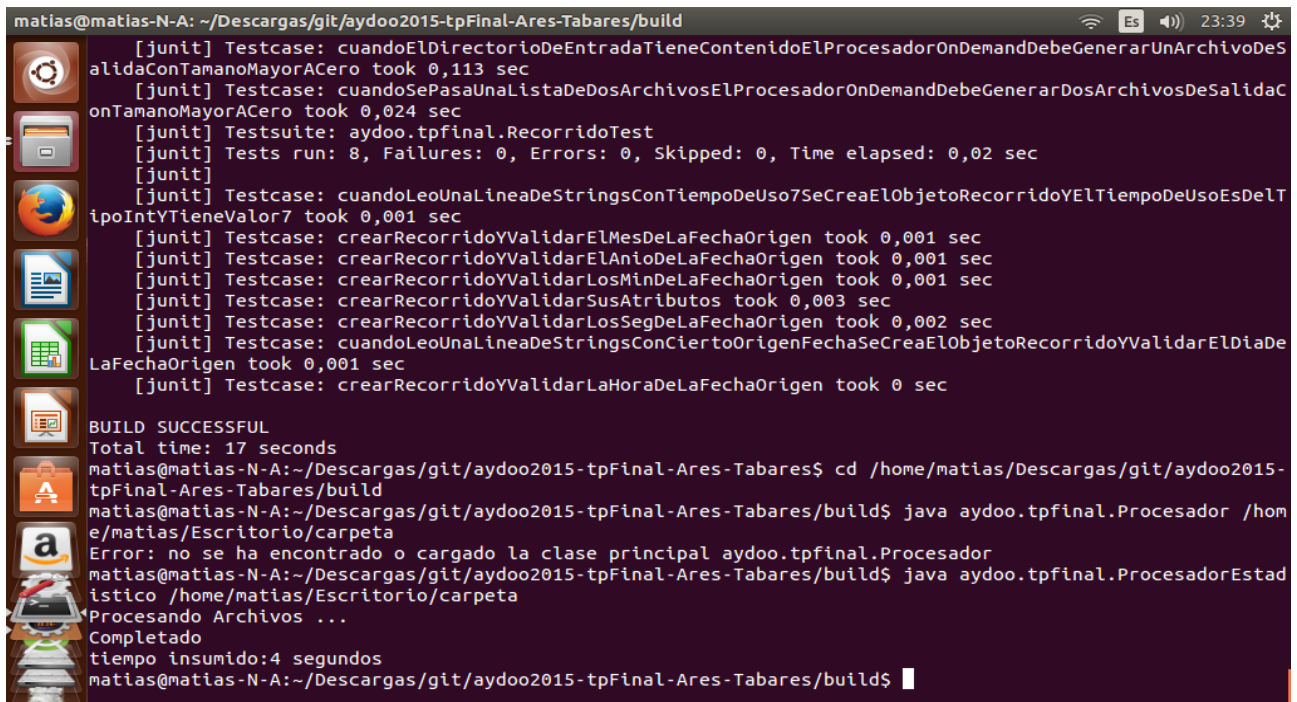
Para evaluar el funcionamiento de la aplicación se utilizó una netbook con las siguientes especificaciones:

1. Procesador: Intel Atom 1.6 Ghz.
2. Memoria: 2 Gb.
3. Sistema operativo: Ubuntu 14.04, ejecutado con problemas de lentitud por escasez de recursos)

## Análisis de la aplicación

Luego de probar la aplicación y revisar el código, la primera optimización de carácter crítico que se evidencia es el uso de la memoria. La aplicación carga a memoria todo el contenido de todos los archivos ZIP del directorio especificado, por tal motivo, produce un procesamiento rápido para archivos ZIP con poco contenido, pero para archivos ZIP de mucho contenido la aplicación no responde. A continuación se evidencia la problemática descrita:

Procesamiento de recorridos-2010.zip

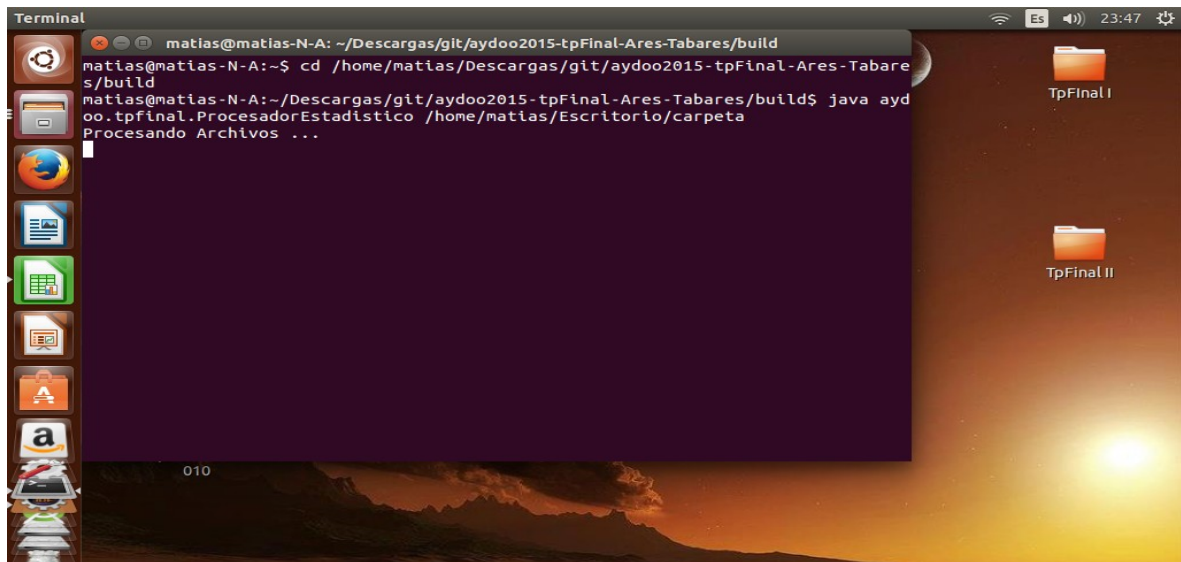


```
matias@matias-N-A: ~/Descargas/git/aydoo2015-tpFinal-Ares-Tabares/build
[junit] Testcase: cuandoElDirectorioDeEntradaTieneContenidoElProcesadorOnDemandDebeGenerarUnArchivoDeSalidaConTamanoMayorACero took 0,113 sec
[junit] Testcase: cuandoSePasaUnaListaDeDosArchivosElProcesadorOnDemandDebeGenerarDosArchivosDeSalidaConTamanoMayorACero took 0,024 sec
[junit] Testsuite: aydoo.tpfinal.RecorridoTest
[junit] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,02 sec
[junit]
[junit] Testcase: cuandoLeoUnaLineaDeStringsConTiempoDeUso7SeCreaElObjetoRecorridoYElTiempoDeUsoEsDelTipoIntYTieneValor7 took 0,001 sec
[junit] Testcase: crearRecorridoYValidarElMesDeLaFechaOrigen took 0,001 sec
[junit] Testcase: crearRecorridoYValidarELAnioDeLaFechaOrigen took 0,001 sec
[junit] Testcase: crearRecorridoYValidarLosMinDeLaFechaOrigen took 0,001 sec
[junit] Testcase: crearRecorridoYValidarSusAtributos took 0,003 sec
[junit] Testcase: crearRecorridoYValidarLosSegDeLaFechaOrigen took 0,002 sec
[junit] Testcase: cuandoLeoUnaLineaDeStringsConCiertaOrigenFechaSeCreaElObjetoRecorridoYValidarElDiaDeLaFechaOrigen took 0,001 sec
[junit] Testcase: crearRecorridoYValidarLaHoraDeLaFechaOrigen took 0 sec

BUILD SUCCESSFUL
Total time: 17 seconds
matias@matias-N-A:~/Descargas/git/aydoo2015-tpFinal-Ares-Tabares$ cd /home/matias/D
matias@matias-N-A:~/Descargas/git/aydoo2015-tpFinal-Ares-Tabares/build$ java aydoo.tpfinal.Procesador /home/matias/Escritorio/carpetas
Error: no se ha encontrado o cargado la clase principal aydoo.tpfinal.Procesador
matias@matias-N-A:~/Descargas/git/aydoo2015-tpFinal-Ares-Tabares/build$ java aydoo.tpfinal.ProcesadorEstado /home/matias/Escritorio/carpetas
Procesando Archivos ...
Completado
tiempo insumido:4 segundos
matias@matias-N-A:~/Descargas/git/aydoo2015-tpFinal-Ares-Tabares/build$
```

Como puede verse al final de la foto, el tiempo insumido fue de 4 segundos. Sin embargo, al intentar procesar un archivo mas pesado, la aplicación no responde:

Procesamiento recorridos-2012.zip

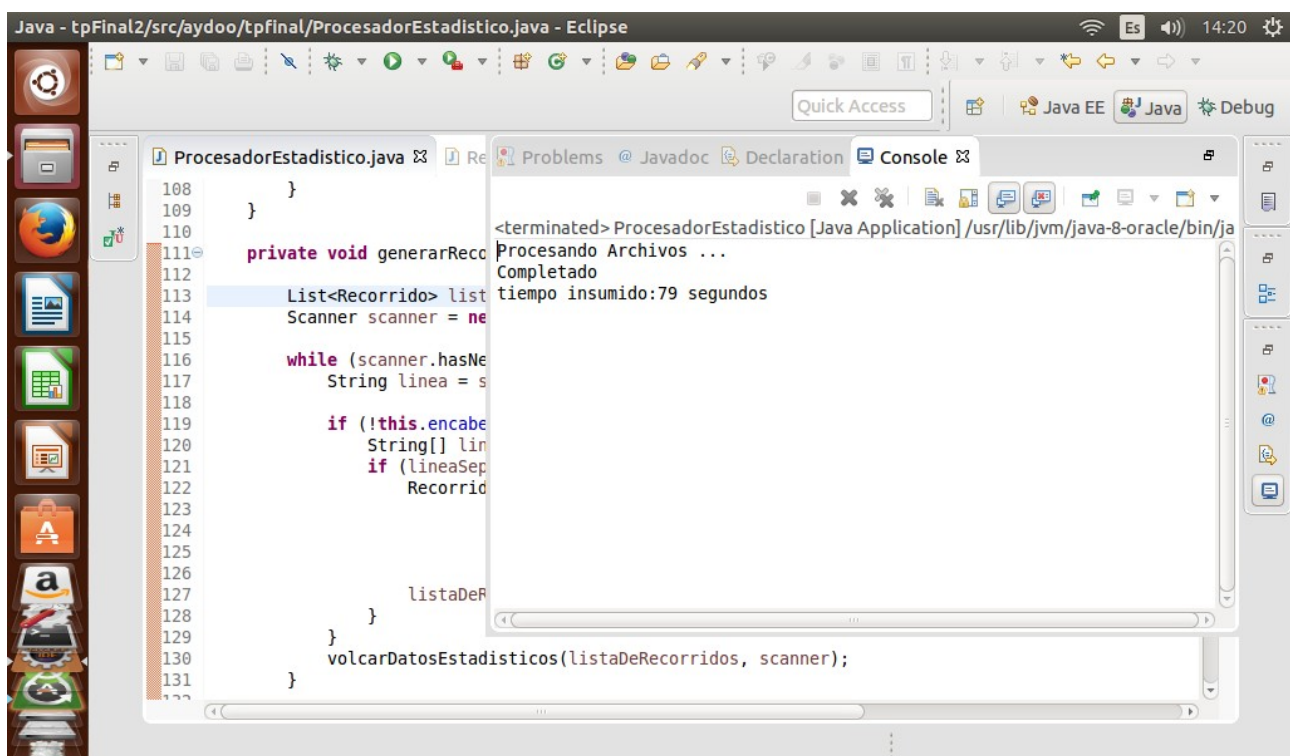


### Solución a emplear:

La estrategia será procesar los archivos por partes, subiendo segmentos de los mismos para procesarlos en memoria, evitando de esta manera llenarla. De acá en adelante, realizaremos las pruebas con el archivo “recorridos-2012.zip”, para medir siempre sobre el mismo archivo.

### 1 ° iteración: mejora en el uso de memoria:

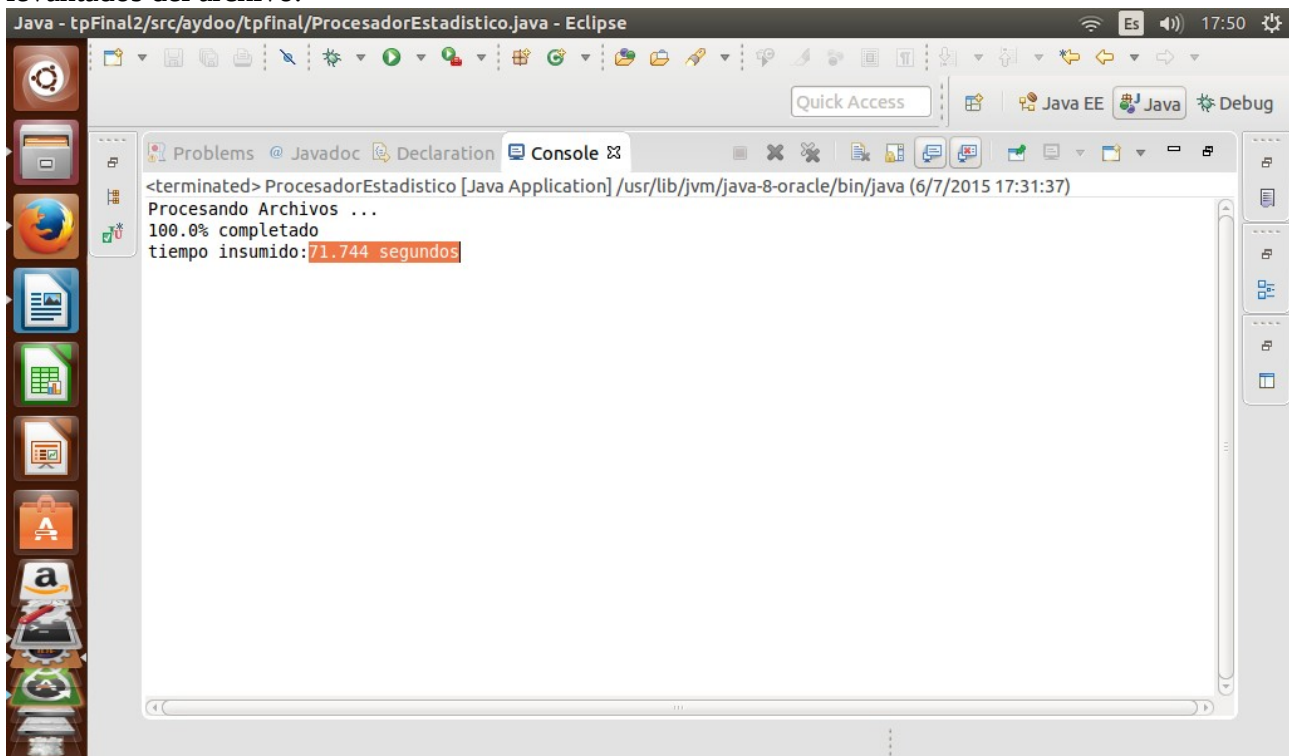
Luego de refactorizar el código y de adaptarlo para poder trabajar los archivos por segmentos, volvimos a correr la aplicación y se logró procesar un archivo grande (recorridos-2012). Sin embargo, el tiempo de procesamiento es alto, 79 segundos para ser exactos:



Intentaremos mejorar el desempeño revisando el código nuevamente.

## 2° iteración: mejora en el tiempo de procesamiento:

En dicha iteración logramos un aumento en el desempeño de la aplicación, disminuyendo el tiempo de procesamiento en 8 segundos. Recordar que siempre se evalúa sobre “recorridos-2012.zip”. Para realizar esto, cambiamos la estructura de datos utilizada en algunas clases encargadas de guardar historiales. Pasamos de usar un TreeMap a un HashMap. Además, pasamos de usar un ArrayList a un LinkedList en la lista encargada de almacenar los registros levantados del archivo.



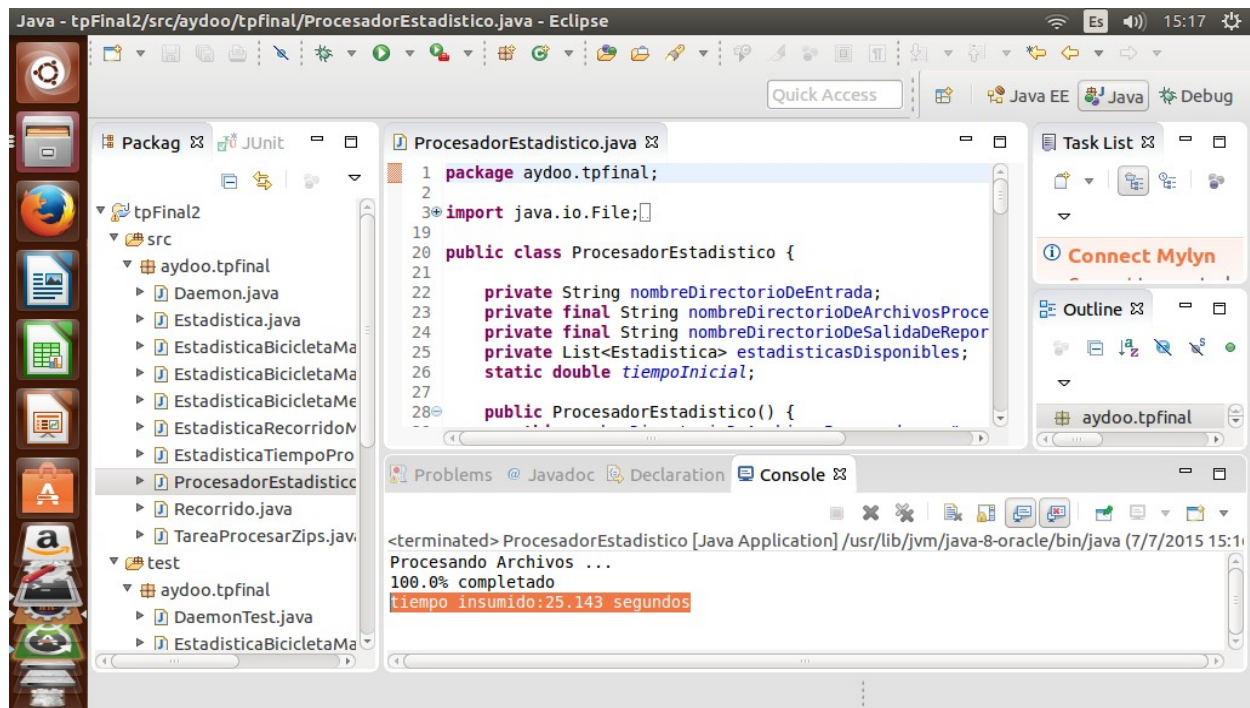
El siguiente objetivo será tratar de disminuir aun mas el tiempo de procesamiento.

## 3° iteración: mejora en el tiempo de procesamiento:

Eliminamos atributos del objeto “Recorrido”, que pertenecían al dominio pero que no eran necesarios para la implementación de la lógica del negocio. Dichos atributos eran:

1. Fecha de origen.
2. Fecha de destino.
3. Id de usuario.

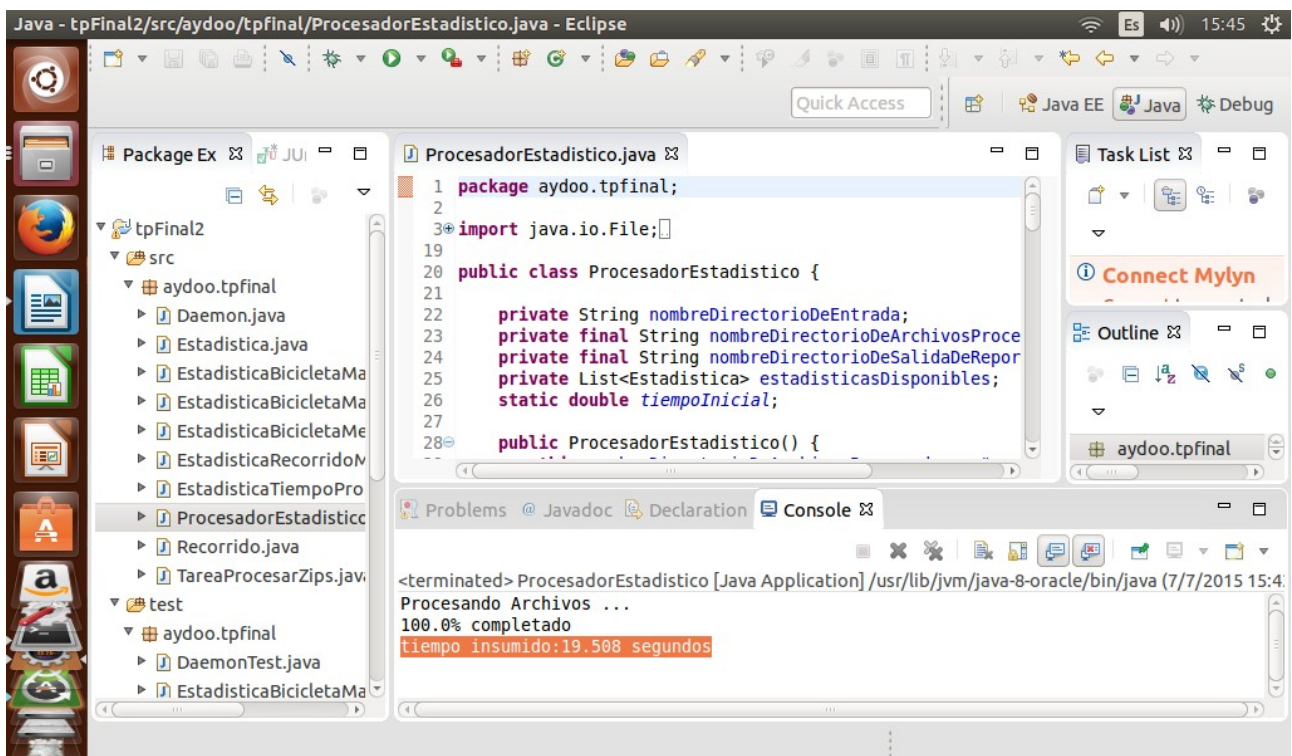
Al hacer esto, disminuimos el tiempo de procesamiento en 46 segundos.



El tiempo de procesamiento paso de 71 segundos a 25 segundos.

#### 4º iteración: mejora en el tiempo de procesamiento:

Modificamos los algoritmos que obtienen las estadísticas. Con esto disminuimos el tiempo de procesamiento en 6 segundos. Pasó de 25 segundos a 19 segundos.





## **5° iteración: mejora en la funcionalidad:**

Se corrigió un bug que no dejaba levantar la aplicación si no había Zips ya cargados en el directorio a trabajar.

## **Agregado de funcionalidad.**

El agregado de la funcionalidad extra fue un proceso sencillo, ya que el diseño creado para las estadísticas consiste en una clase distinta para cada estadística, que heredan de una clase abstracta llamada “Estadística”. Solo fue necesario crear una clase nueva para la estadística pedida como funcionalidad extra, implementarla, y agregarla a la lista que contiene las demás Estadísticas dentro de la clase ProcesadorEstadistico.

## **Resumen de los cambios realizados, en el orden escrito:**

1. Se realizaron cambios en el algoritmo para optimizar el uso de la memoria.
2. Se realizaron cambios en el algoritmo para optimizar el desempeño.
3. Se corrigió bug que solo permitía levantar la aplicación con Zips precargados.
4. Se agregó la funcionalidad pedida.
5. Se refactorizó el código.

## **Lo que se aprendió:**

Pablo Matías Mariani:

En este proceso aprendí la importancia (de carácter vital) de un buen diseño, y cómo puede complicarse la tarea si este carece de buenas prácticas. Por otro lado, pude poner en práctica por primera vez la tarea de mejorar algunos atributos de calidad, como es el uso de la memoria y el desempeño. Dicha tarea me permitió asentar mas los conocimientos respecto a este tema. En lo personal, fue la materia que mas me gustó cursar, combinó un montón de cosas positivas: fue la que mas aprendí y, junto a Análisis y diseño estructurado, opino que es una de las mas importantes de la carrera.

Leonel Campos:

He percibido en gran medida los beneficios, en términos de experiencia y aprendizaje, que se nos haya puesto como consigna extender la funcionalidad a un proyecto realizado por otro equipo. He podido apreciar la importancia de contar con una documentación, como un diagrama de clases, que facilite la comprensión del diseño y la solución que se ha seguido, además de experimentar cómo cuestiones tan sutiles de implementación, como la herramienta o estructura de datos que se va a usar en un determinado lugar, tienen efectos tan grandes en la performance de la aplicación, reduciendo, en este caso, enormemente el tiempo de procesamiento.

Aprendí la importancia que implica contar con un buen diseño que represente prolijamente la realidad que se intenta modelar, ya que esto nos brindó la flexibilidad suficiente para que proyecto pueda escalar en funcionalidad fácilmente.