

Instituto Superior Politécnico “José Antonio Echeverría”

Centro de Estudios de Ingeniería y Sistemas

**Trabajo de Curso**

**Aplicación para la gestión de consumos eléctricos**

**Informe**

Autores: Camilo José del Real Martell

La Habana, Cuba

2011

**Resumen**

Este informe es no es más que un control evaluativo orientado a chequear el trabajo de los autores en la realización de la tarea de fin de semestre de la asignatura Estructuras de Datos 1, con el objetivo fundamental de valorar su desempeño ante el trabajo con estructuras de datos lineales y ficheros. En este informe está plasmada la problemática establecida a los autores así como la solución propuesta por ellos, donde se incluye la descripción de todas las estructuras de datos empleadas en la desarrollo de la aplicación.

**Summary**

This report is not just a control-oriented evaluation to check the work of the authors in carrying out the task to half of the course Data Structures 1, with the primary aim of assessing their performance to work with structures linear data and files. This report is embodied the problematic established authors as well as the solution proposed by them, which includes the description of all data structures used in application development.

Índice

[Introducción 4](#_Toc310058783)

[1. Descripción del negocio 5](#_Toc310058784)

[2. Solución propuesta 6](#_Toc310058785)

[3. Optimización 7](#_Toc310058786)

[3.1 Estructuras de datos 7](#_Toc310058787)

[3.2 Ordenamiento 9](#_Toc310058788)

[4. Persistencia de los datos 10](#_Toc310058789)

[4.1 Estructura de los ficheros 10](#_Toc310058790)

[4.1.1 Sobre los ficheros de los reportes 12](#_Toc310058791)

[Conclusiones 13](#_Toc310058792)

[Recomendaciones 14](#_Toc310058793)

[Bibliografía 15](#_Toc310058794)

# Introducción

El desarrollo de una aplicación para la gestión de consumos eléctricos ha sido designado a los autores de este trabajo, y ante el objetivo de registrar el proceso que conllevó el trabajo de mesa sobre este software, este informe se torna indispensable. Es preciso detallar que el objetivo principal de la aplicación es llevar un control sobre los consumos eléctricos que se van produciendo en los hogares con el fin de que se pueda obtener un gasto monetario real o al menos lo más real posible, ofreciéndole la confianza a los usuarios de cuán alterado o no se encuentran los recibos de cobro eléctrico correspondientes. A lo largo de este informe se exponen como eslabón fundamental a la cadena que es la solución propuesta, las estructuras de datos empleadas en el desarrollo de la aplicación, ya sean estructuras dinámicas de datos de almacenamiento en memoria interna (entiéndase como listas, pilas, colas, árboles, etc.) o estructuras de datos de almacenamiento en memoria externa que no son más que ficheros.

El proceso de creación del software se lleva a cabo empleando tecnología Java; lenguaje de programación y entorno de desarrollo incluido, sobre este último: los desarrolladores hacen uso del entorno de desarrollo integrado Eclipse en su versión 3.5.0 (Galileo), valiéndose de los plug-ins WindowBuilder Pro Eclipse v1.0.0, Eclipse Modeling Framework Project (EMF) v2.4.2, Graphical Editing Framework (GEF) v3.6, estos dos últimos como dependencias de WindowBuilder. Para el modelado de los diagramas de clases UML se hizo uso de la herramienta Enterprise Architect v8.0.856.

1. **Descripción del negocio**

Se ha planteado como problemática a resolver la necesidad de controlar los consumos de corriente eléctrica de un vecindario determinado, con el fin de poder determinar si los recibos de cobro eléctrico son correctos. Para ello es necesario almacenar los registros de consumo diario, donde se solicita que estos sean guardados en un fichero de datos con los siguientes datos: fecha del registro, número del hogar, registro inicial y final de consumo (se entiende que el registro diario de consumo es el final menos el inicial). De un hogar se registra su número, dirección y cantidad de habitantes.

Nota: Para la aplicación se ha tomado como tarifa eléctrica 15 centavos por KW; y los recibos eléctricos la aplicación los obtendrá a través de un fichero de datos.

Se plantean como reportes obtener:

* un fichero de datos con todos los datos de los hogares donde el consumo real estuvo alterado,
* un fichero de texto con las diez casas que más consumo real tuvieron ordenado descendentemente por la cantidad de habitantes.

1. **Solución propuesta**

En la problemática fueron detectadas las siguientes entidades:

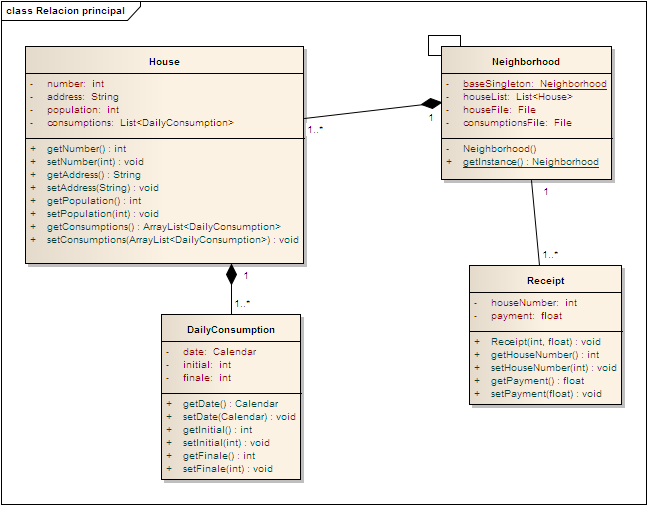
* Hogar: número, dirección y cantidad de habitantes
* Consumo diario: fecha, registro inicial y final de consumo
* Recibo: número de la vivienda, saldo monetario a pagar

Hasta este punto se tiene que los hogares conocen sus consumos diarios, por lo que le correspondería al hogar tener todo el listado de consumos diarios.

* Entidad controladora: listado de los hogares, y para la persistencia de datos contendría dos ficheros para guardar los hogares y los registros de consumos diarios de cada hogar

Es la entidad controladora quien recibe los recibos y sabe qué hacer con ellos.

Tras este análisis se obtuvo el siguiente diagrama UML de clases.



1. **Optimización**

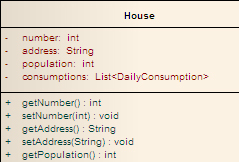
Este epígrafe está dedicado exclusivamente a las técnicas y métodos que se emplearán en el desarrollo de la aplicación, donde intervienen las estructuras de datos que se designaron para las listas y los métodos de ordenamiento.

* 1. **Estructuras de datos**

Apoyándonos en el diagrama de clases, comenzaremos el análisis.

La primera estructura de datos detectada es la lista de los registros de consumos diarios.

Los consumos diarios, para un mes, no excederán los 31 valores y en promedio se encontrarán en 30, por lo que una implementación secuencial no dejaría de ser óptima, a demás de las facilidades que ofrecería el poder obtener en tiempo constante los datos de un día en específico.



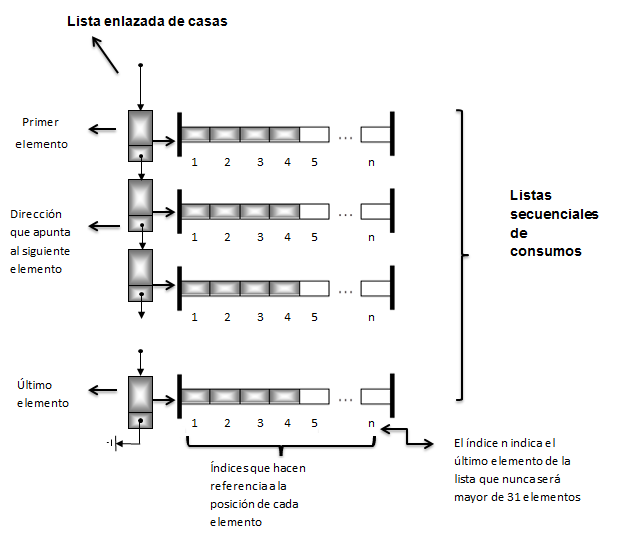
La segunda lista se localiza en la entidad controladora.

Esta lista almacena el conjunto de todos los hogares; la cantidad de hogares es un valor desconocido que puede alcanzar altos niveles, por lo que, una implementación enlazada sería la indicada.



Si para esta última lista se tiene en cuenta la necesidad de ordenarla (para satisfacer el segundo reporte indicado) se podría concluir que una implementación secuencial ofrecería un mayor rendimiento pues puede ser alto el costo que implica obtener una lista secuencial de una enlazada para después ordenarla, pero una tarea de ordenamiento no es algo que se requiera comúnmente y las labores de insertar y eliminar tienden a ser más frecuentes, y afectarían enormemente el rendimiento y velocidad de la aplicación al tener esta que ejecutar corrimientos en la lista para poder aumentarla o disminuirla.

Hasta el momento se han analizado dos listas que al detallar el diagrama UML se puede determinar que juntas conforman una gran estructura con relación bidimensional no lineal que conocemos como multilista o lista de listas figurada de la manera siguiente.



Existe una tercera lista que estaría compuesta por los recibos de cobro de consumo eléctrico; esta es un lista que se obtiene de un fichero que se le proporciona a la aplicación, el cual posee, además de los dato de cada recibo, la cantidad de ellos, uno para cada hogar, bien podría emplearse usar una lista enlazada, como es el caso con los hogares, o también una lista secuencial pues para estos datos sí se posee información precisa sobre las cantidades; para la aplicación se determinó emplear un lista implementada secuencialmente.

* 1. **Ordenamiento**

El tema del ordenamiento ha sido, es y será siempre un aspecto complejo, como tantos otros problemas algorítmicos, que depende fundamentalmente de las capacidades físicas del computador y de la cantidad de datos a ordenar. Si se tienen en cuenta, por ejemplo, los gráficos de orden O(n), O(lg), O(nlgn), O(), O() y O(n!), se aprecia claramente que para pequeñas cantidades es preferible emplear algoritmos con tiempos de ejecución O(n²) por ejemplo, con aras de no emplear un algoritmo muy complejo, pero para grandes cantidades la ventaja, por valores muy superiores, la ganan los algoritmos de orden O(nlgn), es por ello que, ante los datos que manipula la aplicación, hogares principalmente (que serán estos los que deben ser ordenados), y que un vecindario contendrá alrededor de 500 hogares, quizás un poco más, un algoritmo de orden O(nlgn) puede cumplir la tarea de ordenamiento en muy buen tiempo; ha sido elegido para la realización de esta labor el algoritmo de ordenamiento por fusión. Podría emplearse el algoritmo de ordenamiento rápido ( O(nlgn) ), aún más eficiente gracias a constantes multiplicativas con menores valores en su fórmula como tal de cálculo de tiempo, pero se eligió desarrollar la aplicación con un algoritmo más claro de entender que con otro más rápido y con una complejidad de implementación mayor.

1. **Persistencia de los datos**

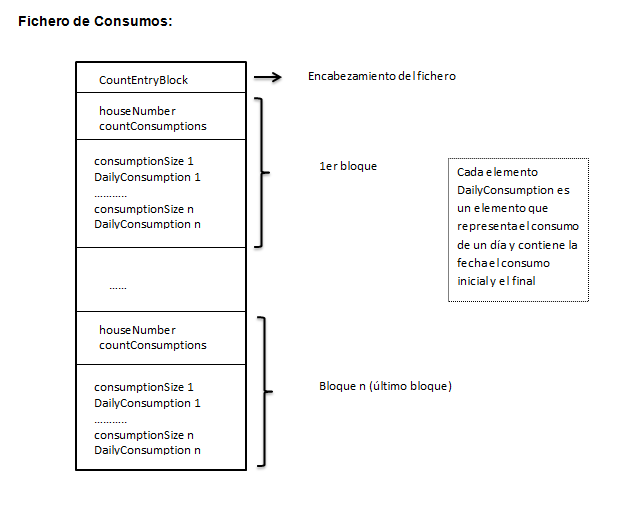
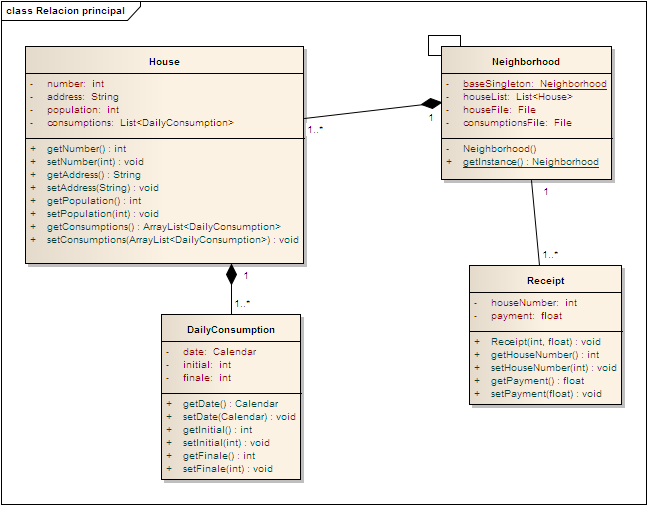
Para el almacenamiento de la información se hará empleo de ficheros binarios que le proporcionarán a la aplicación, cuando inicie, los datos salvados en una sesión anterior, así como la posibilidad de guardarlos cuando esta sea cerrada, o cuando el usuario lo desee. Los datos no serán trabajados directamente desde el fichero pues este es un método lento en comparación a la manipulación directa desde la memoria interna o de acceso aleatorio.

Volviendo a un punto visto en el epígrafe 2, la entidad controladora contiene dos ficheros para almacenar los datos de las casas y los datos de los registros; el trabajo se puede desenvolver con la creación de un solo fichero que englobe bloques representando hogares y registros dentro de los bloques de hogares que representen los consumos diarios, pero era una premisa del trabajo que se crease un fichero solamente para el almacenamiento de los consumos, es desde este punto donde surge la necesidad de crear otro fichero para la persistencia de los datos de los hogares. Con respecto al fichero de los recibos, este surge ante el hecho de que los datos que almacena serán empleados en pocas ocasiones, siendo una pérdida de recursos el mantener esta información constantemente cargada en la memoria RAM, es más rentable mantener estos datos guardados fuera del ámbito de la aplicación, además, el uso de un fichero para archivar los recibos de cobro favorece el traspaso de esta información a la aplicación.

* 1. **Estructura de los ficheros**

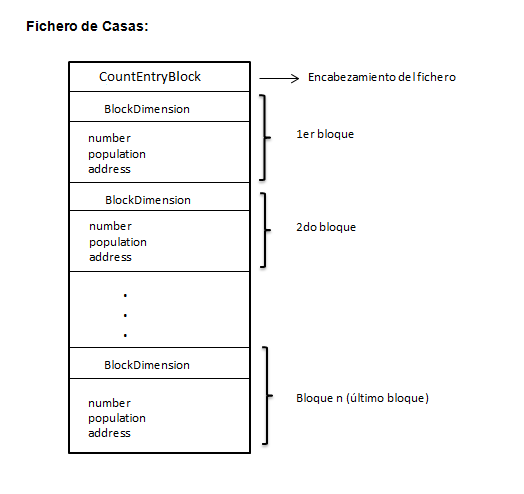
Comencemos con el fichero detectado en el enunciado de la problemática; este cuenta con un encabezado indicando la cantidad de bloques los cuales identifican los consumos que se tienen hasta el momento de cada hogar, seguido se tienen los bloques cuyo encabezado presenta el número del hogar al que identifica y la cantidad de registros de consumos que contiene el bloque, lo restante en el bloque son los registros de consumos.

Los registros de consumos diarios son otra estructura que se encuentra figurada en el diagrama de clases UML

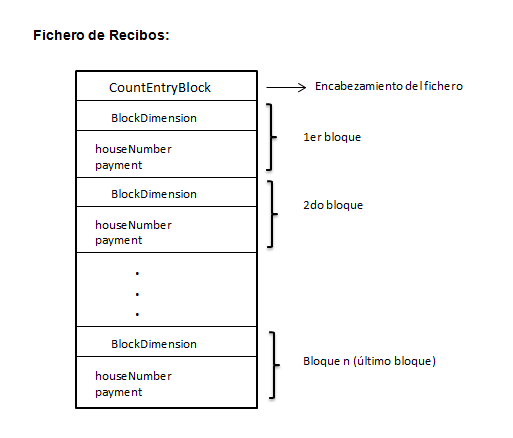


El siguiente fichero surge de la necesidad de hacer persistir en memoria todos los datos introducidos en aplicación.

La estructura de los ficheros continúa siendo muy similar, almacenamos un encabezado con la cantidad de bloques que serían las casas, y en cada bloque los registros de cada casa con sus datos correspondientes.



Y por último el fichero designado al almacenamiento de los recibos de pago.



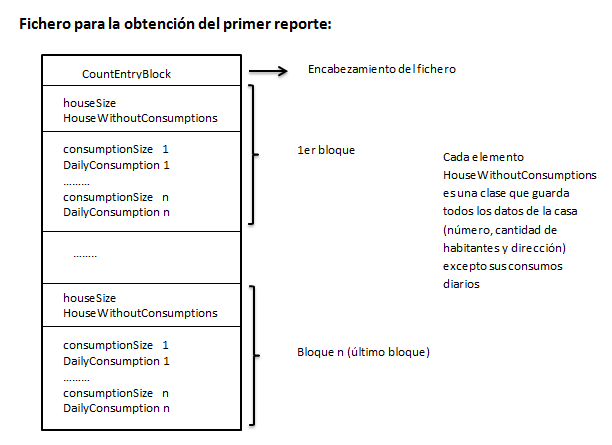
El encabezado del fichero cuenta con la cantidad de recibos que contiene, un recibo es identificado por un bloque o registro que contiene el número de la casa a la que pertenece y el pago monetario de dicho hogar.

* + 1. **Sobre los ficheros de los reportes**

Son necesarios generar dos ficheros en los reportes, uno de texto que no lleva mucho que entender, pero hay otro del que se puede comentar.

*Obtener un fichero de datos con todos los datos de los hogares donde el consumo real estuvo alterado.*

¿Este fichero contendrá la misma estructura del que ya habíamos declarado para el almacenamiento de los hogares? Pues no; para este se ha declarado la siguiente estructura, no tan diferente, pero sí más general, donde los registros de consumos diarios si son almacenados dentro de bloques del fichero que identifican un hogar.



# Conclusiones

El desarrollo de la aplicación está a pasos firmes y no precisamente lentos, los autores o desarrolladores se han planificado bien el tiempo, por lo que la aplicación podrá ser entregada en tiempo cumpliendo con todos los requisitos.

La orientación de esta tarea ha sido favorable ante el estudio realizado por los autores en materia de algoritmización de métodos de ordenamiento y la puesta en práctica de almacenamiento persistente de datos a través del uso de ficheros, concluyendo que:

* No siempre un algoritmo pequeño es la mejor solución.
* Los buenos resultados se logran del estudio, y el estudio de realiza desde distintas fuentes.
* Siempre es útil buscar nuevos algoritmos que resuelvan o mejoren problemas existentes.
* La implantación de reglas de trabajo son muy efectivas, en especial cuando el régimen de estudio es demasiado alto y es necesario aprovechar el tiempo.

# Recomendaciones

La problemática orientada presentaba una solución trivial, esto implica que pueden ser ampliadas las situaciones a enfrentarse los desarrolladores, que podrían ser: emplear la tarifa real de cobro del pago de la electricidad así como crear el mecanismo para poder cambiarla, incluir más reportes referentes a los consumos incorrectos que es el objetivo fundamental a solucionar en el problema. Se podrían también crear cuentas de usuario para que los datos solo puedan ser manejados por personal autorizado. En resumen, las nuevas mejoras dependerían primordialmente de las necesidades que “el cliente” solicite.

# Bibliografía

1. Colectivo de autores. http://chuwiki.chuidiang.org/index.php?title=Lectura\_y\_Escritura\_de\_Ficheros\_en\_Java

Visitado 11/11/2011

1. Colectivo de autores. Java 2 SE 6 Documentation.
2. Heileman, Gregory L. *“Estructuras de datos, algoritmos y programación orientada a objetos”.* Editorial Félix Varela, 2003.
   * Epígrafe 2.4 Un ejemplo elaborado: ordenación
3. Weiss, Mark Allen. *“Estructuras de datos en java. Compatible con Java 2”.* Editorial Félix Varela, 2008.
   * Epígrafe 2.6 Entrada y Salida
   * Epígrafe 5.6 Problema de la búsqueda estática
   * Epígrafe 6.4 Listas enlazadas
4. Lafore, Robert. *“Data structures and algorithms in Java”.* Editorial Sams, 1998.
   * Chapter 3 Simple sorting
   * Chapter 7 Advanced sorting
5. Drake, Peter. *“Data Structures and Algorithms in Java”.* Prentice Hall, 2005.
   * Chapter 8 Searching and sorting