Desarrollo de un Programa Informático para la Formulación de Piensos Compuestos

Trabajo de Fin de Grado en Matemáticas

Andrea Aguilar Ibáñez

Directores: Pedro Mateo Collazos
Joaquín Surra Muñoz

Universidad de Zaragoza
24 de junio de 2016

Abstract

This work is framed within the Operations Research, more specifically in Linear Programming, together with Software Development and Compound Feed Formulation.

The main purpose of this work is to develope an informatical software powerful enough destined to monogastric animals' compound feed formulation, mainly for academic purpose. Also, another main purpose is the application of mathematical knowledge in real life.

The work is organized as follows. It consists of a central part, which includes five chapters that we will detail below, and three appendices.

In Chapter 1 it presents a brief introduction about global situation at feed production and what is the aim of feed formulation: to find with least-cost an optimal ingredients combination to achieve nutrients requirements depending on the animal for whom the feed is destinated. Following this, there is a section talking about all the main purposes of the present work. After that, it shows several examples where linear programming has been applied to feed formulation and optimization since its birth. Finally, this chapter names some libraries that have been used to implement the software with Java.

Chapter 2 enumerates the necessary information a high qualified nutritionist needs to find an optimal ingredients blend that provides every animal with suitable and healthy nutrients: nutrient feed concentration, nutritional contribution of each ingredient, nutrient and ingredient limits and ingredients costs. To can imagine easily the meaning of these requirements, an example of chicken feed is shown.

The aim of Chapter 3 is to introduce the mathematical background of the work. It shows the basic contents of linear programming, the way the linear programming problem is drawn up and the solving method of these problems through the Simplex algorithm. Then, it is explained that every linear programming problem has associated another problem, called dual problem. Duality in linear programming allows for studying each nutrient unit effect in formulation global cost. Because the main interest of feed formulation is to minimize production costs, it is also explained here the posibility of varying ingredient costs keeping the present optimal solution doing a post-optimal analysis.

Chapter 4 is the central part of the work. It explains how the software has been developed. First of all, it presents the global structure of the program and all options it includes: a file menu where it is possible to choose create, upload, save and save as a formula, another menu related with linear programming problem generation and optimization, and the post-optimal analysis. Afterwards, if the finded solution seems interesting, the program gives the option of save the results in an Excel report. Also it is an explanation of the way all the implemented classes work.

In the last chapter, Chapter 5, conclusions enumeration of the main part of the work is made. All objectives set out at the beginng of this work were achieved, on one hand to develop an usable and efficient compound feed formulation software, on the other hand to apply already known tools and acquire new knowledge in different areas that are not our main one. In addition, there are also shown several possible improvements in future.

The first one of the appendixes, Appendix A, was written to explain the linear programming problem formulation method used by GLPK library. GLPK uses a different structure to solve problems that the one presented in Chapter 3.

In Appendix B a series of figures is given to see the program interface and how it works. This display helps to understand the explanations given in Chapter 4 for every components.

Due to this work has an important programming component, in the last one of the appendices, Appendix C, is included all the programming code we have been developed to create the software. It is divided in eight sections, one for each implemented class.

Índice general

Abstract III

1. Introducción 1
- 1.1. Objetivos . . . 1
- 1.2. Metología y Herramientas . . . 2

2. Diseño de dietas 5

3. Programación Lineal 7
- 3.1. El Problema de Programación Lineal . . . 7
- 3.2. Dualidad en Programación Lineal . . . 10
- 3.3. Análisis Post-óptimo . . . 12

4. Desarrollo del Programa 13
- 4.1. Panel Ingredientes y Nutrientes . . . 14
- 4.2. Panel Fórmulas . . . 14
- 4.3. Panel Tabla de Ingredientes . . . 15
- 4.4. Opción Fichero . . . 15
- 4.5. Opción Formulación . . . 15
- 4.6. Opción Informes . . . 16
- 4.7. Opción Tabla ingredientes . . . 16
- 4.8. Clases implementadas. Funcionamiento . . . 16
  - 4.8.1. Clase Ventana . . . 16
  - 4.8.2. Clase baseDeDatos . . . 16
  - 4.8.3. Clase datosGlobales . . . 17
  - 4.8.4. Clase Ingrediente y clase Nutriente . . . 17
  - 4.8.5. Clase PPLs . . . 17
  - 4.8.6. Clase GeneraExcel . . . 19
  - 4.8.7. Clase AcercaDe . . . 19

5. Conclusiones y Trabajos Futuros 21
- 5.1. Conclusiones . . . 21
- 5.2. Trabajo Futuro . . . 22

Bibliografía 23

Anexos 25

A. Formato de PPL en GLPK 27

B. Imágenes del Programa 29

C. Código del Programa **35**
C.1. Ventana principal 35
C.2. Clase baseDeDatos 64
C.3. Clase datosGlobales 69
C.4. Clase Ingrediente 70
C.5. Clase Nutriente 71
C.6. Clase PPLs 72
C.7. Clase GeneraExcel 78
C.8. Clase AcercaDe 86

Índice de figuras

2.1. Ingredientes y Nutrientes seleccionados. 6

4.1. Opciones de los menús del programa. 13
4.2. Inicio del programa. 14
4.3. Formato del archivo de fórmulas. 17
4.4. Funciones para la creación del documento Excel. 19

B.1. Inicio del programa. 29
B.2. Tabla de valores nutricionales. 29
B.3. Añadir ingredientes. 30
B.4. Cargar fórmula. 30
B.5. Fórmula cargada. 30
B.6. Opción de generación del problema. 31
B.7. Problema generado. 31
B.8. Resultado de la optimización del problema. 31
B.9. Análisis de sensibilidad del problema. 32
B.10. Introducir partida en Kg para generar Excel. 32
B.11. Abrir Excel creado. 32
B.12. Hoja Fórmula del informe en Excel. 33
B.13. Hoja Análisis de sensibilidad del informe en Excel. 33
B.14. Hoja Distribución de nutrientes del informe en Excel. 33

Capítulo 1

Introducción

En la actualidad mundial el aumento de la población, el desarrollo económico y el incremento del poder adquisitivo de las personas hacen que el consumo y la demanda de alimentos de origen animal sea cada vez mayor. Mientras tanto, la producción y la disponibilidad de las materias primas empleadas en la alimentación de los distintos animales se ve comprometida, agravando la situación. Además, si se tiene en cuenta que en la Unión Europea el gasto en la alimentación de los animales representa entre el 50 % y el 93 % del coste total de producción [4], esta debe ser cada día más eficiente para hacer rentables las explotaciones ganaderas. La producción de piensos es un elemento fundamental en el sistema de producción ganadero intensivo español, especialmente en la ganadería de producción de carne. Dado que de los costes de los piensos el 80 % corresponde a las materias primas [13] o ingredientes empleados, siendo estos un factor no controlado por las fábricas de piensos ni por los ganaderos, un modo de reducir dichos costes sería la optimización en la formulación de los piensos. De por sí la reducción de los costes resulta complicada, y además de eso el proceso industrial de fabricación de piensos debe adaptarse a una economía de escala ya que requiere un gran volumen de dinero en la compra de materias primas, gastos energéticos y de amortización de las instalaciones para como resultado obtener un pequeño valor añadido, por lo que es necesario mover grandes volúmenes de producción y además en un ambiente de gran competitividad y exigencias legales [16].

La formulación de piensos consiste básicamente en encontrar, a coste mínimo, una combinación óptima de materias primas e ingredientes, determinada por los propios contenidos en nutrientes de estos y sus precios, que además sea capaz de satisfacer o colmar las necesidades o requerimientos de nutrientes de los animales, según especie, fase productiva (gestación, lactación, crecimiento, engorde, ...), etc. [16], empleándose para ello programas informáticos de formulación basados en la metodología de la programación lineal.

Finalmente, hay que resaltar que los programas informáticos comerciales destinados a la formulación de piensos son excesivamente caros e inalcanzables para la mayoría de pequeñas empresas, ganaderías y, como es nuestro caso, para fines docentes.

1.1. Objetivos

El objetivo principal del presente trabajo es el de desarrollar un programa informático destinado a la formulación de piensos compuestos de animales monogástricos (cerdos y aves) suficientemente potente y versatil para poder ser utilizado para fines docentes, en pequeñas empresas, ganaderías, etc., ya que como se ha comentado anteriormente los programas informáticos comerciales resultan inaccesibles. El interés primero de dicho software es su utilización con fines académicos para docencia práctica de formulación y fabricación de piensos de la asignatura 'Sistemas de Producción Animal' correspondiente al Máster Universitario en Ingeniería Agronómica por la Universidad de Zaragoza de la Escuela Poli-

técnica Superior de Huesca, aunque sería también el comienzo de una posterior ampliación y mejora para la utilización por parte de otros colectivos que no pueden acceder a los programas comerciales. La formulación del pienso se realizará en función del coste económico (mínimo coste) a la cual se impondrán restricciones (máximos o mínimos) de ingredientes y nutrientes. La adquisición de los ingredientes se realiza desde una base de datos con la información de los valores nutritivos de las diversas materias primeras disponibles y elaborada por la Fundación Española para el Desarrollo de la Nutrición Animal (FEDNA) [2].

Como objetivo secundario pero también importante del trabajo está el hecho de afrontar un problema real, en el que se van a tener que aunar distintos conocimientos adquiridos en el grado: programación lineal (Investigación Operativa), programación en Java (Informática II) y modelado de problemas (Investigación Operativa, Grafos y Combinatoria, Modelización Matemática, etc.) para poder lograr el objetivo principal.

Los objetivos operativos del software requieren por un lado utilizar la modelización mediante problemas de programación lineal para formular el problema de la dieta y por otro utilizar el algoritmo Simplex para la obtención de la solución óptima (si existe) de la dieta propuesta. Además de esto, se va un poco más allá planteando el análisis post-óptimo de los precios de los ingredientes así como la valoración de los nutrientes, elementos que requieren de la teoría de la dualidad en programación lineal.

Otro objetivo importante es que el programa se puede utilizar dedicando un tiempo mínimo de aprendizaje por parte de los usuarios, sobre todo pensando en su uso en docencia donde el tiempo es importante y no tendría sentido dedicar mucho tiempo a aprender cómo manejarlo.

En resumen, los objetivos que se plantearon a la hora de realizar este trabajo fueron implementar un software de fácil uso, modelar problemas de dietas mediante técnicas de programación lineal, resolver dichos problemas y hacer un análisis post-óptimo de los resultados obtenidos.

1.2. Metología y Herramientas

Formulación

Como se ha comentado al inicio del capítulo, el costo del pienso es el factor que más determina la rentabilidad de la cría de animales. Se han utilizado con diferente éxito varios modelos matemáticos con la intención de economizar la formulación de dietas. Entre todos los métodos, la programación lineal (PL) se ha usado eficazmente durante muchos años para minimizar el costo de la fórmula o economizar la formulación de mezclas concentradas. A pesar de que este método presenta ciertas carencias ya que, por ejemplo, no puede incluir la variabilidad de los nutrientes ni la del costo de los ingredientes del pienso, se utiliza habitualmente como una herramienta para el diseño de fórmulas [5].

Las primeras referencias de utilización de la PL en la optimización de dietas en ganadería se atribuyen a Waugh (1951) [20], y tuvo un amplio desarrollo a partir de los años 60 del siglo pasado con los trabajos de Heady y Dillon (1961) [7] optimizando funciones en producción animal en general o el de Heady et al. (1963) en ganado vacuno de carne, como también el de Heady et al. (1964) [8] en la producción de leche. Desde entonces, la PL ha sido utilizada ampliamente en el modelado de problemas de alimentación de varias especies ganaderas (Chakeredza et al. (2008) [3]; O'Connor et al. (1989) [11]) y acuicultura (Barbieri, Cuzon 1980 [1]). Vande Haar y Black (1991) [19] describieron que los modelos de PL pueden resolver complicados conjuntos de requerimientos de nutrientes dando una fórmula equilibrada. Sklan y Dariel (1993) [14] desarrollaron un programa nutricional para ganado de alta producción lechera con el fin de alcanzar niveles de producción de leche eficientes y rentables.

Tedeschi et al. (2004) [17] aplicó en un modelo el concepto de usar nutrientes eficientes en relación con la rentabilidad de las granjas lecheras. Guevara (2004) desarrolló una hoja de cálculo de programación lineal basada en el análisis de costes y validó su eficacia en ordeño y operaciones personalizadas para vaquillas [5].

Tanto la PL como disciplina matemática en particular, como la investigación operativa en general, han experimentado un gran avance debido a las mejoras en las tecnologías y de los ordenadores. Las dietas de costo mínimo pueden formularse mediante representación gráfica, resolviendo las ecuaciones manualmente, en hojas de cálculo o mediante programas de ordenador desarrollados en base a este principio.

La idea básica en la formulación de piensos compuestos mediante PL es minimizar el coste al mismo tiempo que se proporcionan los nutrientes adecuados de acuerdo con la demanda del animal. El modelo de PL para formulación de piensos tiene la estructura siguiente, con n variables de decisión y m restricciones:

$$\begin{aligned} &\text{minimizar / maximizar} \quad Z = \sum_{j=1}^{n} c_j x_j \\ &\text{sujeto a} \quad \sum_{j=1}^{n} a_{ij} x_{ij} \quad (\leq, =, \geq) \quad b_i, \quad i = 1, \ldots, m \\ &\qquad\qquad x_j \geqslant 0, \ j = 1, \ldots, n, \end{aligned}$$

donde Z es el coste total de la fórmula, c_i el coste por kilogramo de materia seca del j-ésimo ingrediente (son los coeficientes de las variables de decisión en la función objetivo), x_j es la cantidad del ingrediente j y la variable decisión, a_{ij} es la cantidad del i-ésimo nutriente en el ingrediente j y b_i es la cantidad necesaria del nutriente i en la fórmula.

A continuación se muestra un sencillo ejemplo en el que se plantea el uso de los ingredientes "cebada 2C 11.3 PB" y "maíz nacional"; de los nutrientes "materia seca" y "energía en aves", seguido de su solución óptima.

$$\begin{array}{lrrcrcrcl} \text{Minimizar} & Z = & 0{,}168x_1 & + & 0{,}178x_2 & & & & \\ \text{Sujeto a} & & x_1 & + & x_2 & = & 1 & & \\ & & 90{,}2x_1 & + & 86{,}2x_2 & \geq & 85 & & \\ & & 90{,}2x_1 & + & 86{,}2x_2 & \leq & 100 & & \\ & & 2800x_1 & + & 3280x_2 & \geq & 5000 & & \\ & & 2800x_1 & + & 3280x_2 & \leq & 3000 & & \\ & & & & & & & & \\ & & x_1 \geq 0, & & & & x_1 \leq 1{,}0 & & \\ & & x_2 \geq 0{,}2, & & & & x_2 \leq 1{,}0. & & \end{array}$$

Se requiere que entre un 0 % y un 100 % de la composición del pienso sea cebada y que entre un 20 % y un 100 % sea maíz. Asimismo, la composición de nutrientes requerirá entre un 85 % y un 100 % de materia seca, y entre 3000 Kcal/Kg y 5000 Kcal/Kg de energía.

La solución óptima del problema tiene un costo de 0,173€ por kilogramo de pienso, utilizándose un 58,333 % de cebada y un 41,667 % de maíz, y logrando una composición de 88,533 % de materia seca y 3000 Kcal/Kg de energía.

Además de la obtención de la fórmula óptima que se ajusta a los requerimientos establecidos, a través de la teoría de la dualidad de la PL se puede realizar el análisis de sensibilidad de las soluciones,

es decir bajo qué condiciones la solución obtenida seguiría siendo la mejor; o también permite valorar implícitamente los nutrientes, o sea cómo influiría en el precio de la fórmula el incrementar o reducir los requerimientos de un cierto nutriente.

La formulación obtenida se mantendría óptima si el precio de la cebada se mantiene por debajo de 0,18€ y si el precio del maíz se mantiene por encima de 0,17€.

Librerías

Para el desarrollo del programa se ha utilizado el lenguaje de programación Java, y como entorno de desarrollo NetBeans. La decisión de hacer uso de Java se debe a que es un lenguaje de programación muy extendido y multiplataforma. Documentación sobre Java y NetBeans puede encontrarse en [12] y [10]. Tanto la base de datos como la clase para acceder a esta fue proporcionada por los directores del Trabajo Fin de Grado. Tres son las librerías a destacar:

- **SQLite JDBC**: Librería bajo licencia Apache que se utiliza para acceder a la base de datos FEDNA2015 creada con el motor de bases de datos de dominio público SQLite. Dicha librería permite la ejecución desde Java de las distintas consultas a la base de datos [15].
- **GLPK (GNU Linear Programming Kit)**: Es una librería que proporciona herramientas para resolver problemas de programación lineal y programación entera mixta de grandes dimensiones y otros problemas similares. Está implementada en C pero existen diversos enlaces en otros lenguajes (Java, C#, F#, Visual Basic, etc.) para ejecutar estas librerías. En el programa se utiliza "GLPK for Java", que proporciona una "API" para realizar las llamadas desde programas .java. Esta librería se distribuye bajo licencia GNU [6].
- **POI**: Desarrollada por APACHE, es una API en Java para acceder a documentos de Microsoft. En el programa se utilizará la parte de la librería encargada de generar documentos Excel. Esta librería se distribuye bajo licencia Apache [18].

Capítulo 2

Diseño de dietas

Para llevar a cabo una adecuada formulación de un pienso, se busca la combinación óptima de ingredientes que proporcione un determinado aporte de nutrientes el cual permita un rendimiento productivo máximo, sin afectar al bienestar o a la salud de los animales, lo que evidentemente es competencia de un experto nutricionista. Para ello es necesario disponer de la siguiente información:

- La concentración de nutrientes que debe contener el pienso, que dependerá de los requerimientos de los animales a los cuales va destinado. Entre las necesidades se incluyen las de Energía (Kcal/Kg) o las de los aminoácidos esenciales: lisina, metionina, etc. (las de proteína son menos importantes ya que se formula en función de los aminoácidos); también las de algunos ácidos grasos insaturados (esenciales) o de macrominerales como calcio y fósforo y otros microminerales, expresados todos en porcentajes.
- El valor nutritivo de las materias primas e ingredientes. Por lo general en las fábricas de piensos se emplean las Tablas FEDNA [2], las mismas que utilizará el software desarrollado, a no ser que se disponga de un análisis específico, cosa poco frecuente.
- Las restricciones o limitaciones de inclusión de ciertas materias primas, establecidas por la experiencia del nutricionista y motivadas por diversas causas, bien nutricionales: ya sean ingredientes poco digestibles para animales de primeras edades (pollitos, lechones, etc.); bien de tipo tecnológicas: por contener demasiada grasa o fibra que dificultarían la granulación posterior del pienso; o de tipo comercial: aspectos tan subjetivos como las preferencias del ganadero (color u olor del pienso) o propietarios de mascotas (forma, colores, etc.).
- El precio de las materias primas, ya que tiene un impacto fundamental en el coste final del pienso y su valor resulta muy variable pues depende de factores ajenos a las fábricas de piensos o ganaderos, y determinará su inclusión o no y el porcentaje de inclusión en la fórmula final.

Se considera a continuación como ejemplo la dieta del fichero `gallineas.die`, el mismo considerado en el Anexo B para mostrar los avances del programa en imágenes. Esta dieta cuenta con la selección inicial de 20 ingredientes como materias primas y 13 nutrientes requeridos para un pienso adecuado para gallinas.

Ingredientes	Mínimo	Máximo	Precio
CEBADA 2C 11.3	0	100	0,168
MAIZ NACIONAL	0	100	0,178
MAIZ FRANCES	0	100	0,171
TRIGO BLANDO 12.9 PB	0	100	0,171
GLUTEN FEED MAIZ 21%	0	100	0,190
MELAZA REMOLACHA	0	100	0,135
HNA. COLZA 00 SOLVENTES	0	100	0,250
HNA.GIRASOL 34/	0	100	0,230
HNA.SOJA 47	0	100	0,382
ALFALFA GRANULADA 16,5% PB	0	100	0,150
PULPA REMOLACHA	0	100	0,173
MANTECA	0	100	0,625
GRASA MEZCLA	0	100	0,595
AC. SOJA	0	100	0,702
AC. PALMA	0	100	0,697
CARBONATO CALCICO	0	100	0,135
FOSFATO BICALCICO ANH.	0	100	0,660
CLORURO SODICO MARINO 98	0	100	0,250
DL METIONINA	0	100	3,550
L-LISINA HCL	0	100	2,00

Nutrientes	Mínimo	Máximo
MATERIA SECA (%)	85	100
EMA_AVES Kcal/Kg	2740	5000
Ca (%)	3,7	4,10
Pdisp. (%)	0,33	0,36
C18\:2 (%)	1,20	100
LYS (%)	0,71	100
MET (%)	0,31	100
M+C (%)	0,61	100
THR (%)	0,5	100
TRP (%)	0,16	100
ILE (%)	0,59	100
Na (%)	0,14	100
K (%)	0,45	0,90

Figura 2.1: Ingredientes y Nutrientes seleccionados.

En estas tablas pueden verse las especificaciones que limitan la concentración mínima y máxima de cada nutriente que debe tener el pienso para gallinas. Como se había enunciado antes, se especifican entre otros, la energía necesaria en el caso de aves, los aportes de lisina y metionina, calcio, etc.

También están presentes las limitaciones de inclusión de las materias primas. En este caso todas están entre 0 y 100, así como sus precios correspondientes. Dado que ninguna de estas materias primas tiene limitación mínima, habrá alguna de ellas que no se incluirá en la dieta para conseguir el pienso óptimo, como se puede ver en la figura B.8 del Anexo B al optimizar el problema.

En el siguiente capítulo se presentarán brevemente los principales elementos de la programación lineal necesarios para el desarrollo y obtención de dietas óptimas.

Capítulo 3

Programación Lineal

Un problema de programación lineal trata de maximizar o minimizar una función lineal de las variables de decisión y en él cada restricción se expresa como una ecuación o inecuación lineal [9].

En general, los problemas de optimización lineal incluyen restricciones de igualdad y desigualdad. Además, pueden incluir variables que están restringidas a ser no negativas, no positivas o que no están restringidas en signo.

3.1. El Problema de Programación Lineal

Sin pérdida de generalidad se asumirá que el problema de programación lineal (PPL) estará en forma estándar, tal y como se muestra en (3.1). Es decir, un problema de mínimo en el que todas las restricciones son de igualdad y en el que todas las variables son de tipo mayor o igual que cero[1].

$$\begin{array}{lccccccccc} \min & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & & \\ \text{sujeto a} & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ & & & & & \dots & & & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{2mn}x_n & = & b_m \end{array} \tag{3.1}$$

$$x_j \geq 0,\ j = 1,\dots,n\ .$$

Definición 3.1. El recinto de factibilidad S de un PPL es un poliedro, y se representará como

$$S = \left\{ x_j \in \mathbb{R}^n : \sum_{i=1}^{m} a_{ij}x_j = b_i,\ x_j \geq 0,\ j = 1,\dots n \right\}.$$

Un PPL será factible si la región de factibilidad S es no vacía.

Las especiales características de un PPL hacen que, si su región de factibilidad es no vacía y acotada, es decir si lo son sus variables de decisión $x_j,\ j = 1,\dots,n$ la solución óptima global del problema se encuentre siempre en lo que se denomina un "punto extremo" del recinto. En el ejemplo de la sección anterior los puntos extremos se corresponden con las esquinas del recinto de factibilidad. Esta característica se explota en el algoritmo que se utilizará para resolver el problema, denominado algoritmo del Simplex[2], y que se dedica a ir calculando sucesivos puntos extremos del recinto (esquinas del poliedro)

[1] Esta suposición no supone ninguna pérdida de generalidad ya que cualquier problema puede plantearse en forma estándar [9].

[2] El desarrollo del algoritmo del Simplex se debe al matemático estadounidense G. Dantzig, en 1947.

hasta localizar la esquina que corresponde a la solución óptima.

A continuación se enunciarán los principales resultados que sustentan las bases de la programación lineal y de sus algoritmos de resolución.

Con objeto de simplificar el desarrollo y hacerlo más compacto vamos a utilizar notación matricial para representar el problema. Para ello se definen los siguientes vectores:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} y \quad \mathbf{c} = (c_1 \ \cdots \ c_n),$$

y la matriz

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}.$$

Los vectores se nombran como vector de variables de decisión, vector de recursos y vector de costos[3], respectivamente y la matriz A se denomina matriz de coeficientes tecnológicos.

Con estos elementos la formulación del PPL en forma estándar queda tal y como se muestra en (3.2).

$$\begin{array}{lrcl} \text{minimizar} & \mathbf{cx} & & \\ \text{sujeto a} & A\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & \mathbf{0}. \end{array} \tag{3.2}$$

Como se ha comentado previamente, la solución óptima de un PPL, si existe, se encontrará en un punto extremo. Ahora se van a definir y posteriormente caracterizar los puntos extremos y direcciones extremas, elementos necesarios para poder establecer el teorema que caracterizará las soluciones óptimas de los PPL.

Definición 3.2. Un punto $\mathbf{x} \in S \subset \mathbb{R}^n$ convexo se dice que es un punto extremo de S si $\mathbf{x} = \lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2$ con $\mathbf{x}^1, \mathbf{x}^2 \in S$ y $\lambda \in (0,1)$ implica que $\mathbf{x} = \mathbf{x}^1 = \mathbf{x}^2$. Es decir, $\mathbf{x}$ no se puede poner como una combinación lineal convexa de otros dos puntos del conjunto S.

Definición 3.3. Un vector $\mathbf{d} \in \mathbb{R}^n$ se dice que es una dirección extrema de S si $\mathbf{d} = \lambda_1\mathbf{d}^1 + \lambda_2\mathbf{d}^2$ con $\mathbf{d}^1, \mathbf{d}^2$ direcciones de S y $\lambda_1, \lambda_2 > 0$ implica que $\mathbf{d}^1 = \alpha\mathbf{d}^2$, para algún $\alpha > 0$. Es decir, $\mathbf{d}$ no se puede poner como una combinación lineal positiva de otras dos direcciones del conjunto.

Tras las definiciones anteriores se presentan los teoremas que caracterizan tanto los puntos como las direcciones extremas de un poliedro, que corresponden al recinto dentro del cual se buscan las soluciones de un PPL.

Teorema 3.1. *Sea $S = \{\mathbf{x} : A\mathbf{x} = \mathbf{b} \geq \mathbf{0}\}$, siendo A una matriz $m \times n$ de rango m y $\mathbf{b}$ un vector $m \times 1$. Un punto $\mathbf{x}$ es un punto extremo de S si y sólo si A puede descomponerse como $A = [B,N]$, donde B es una matriz $m \times m$ de rango m tal que $B^{-1}\mathbf{b} \geq \mathbf{0}$ y*

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} B^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix}.$$

[3] El vector de costos se define intencionadamente como un vector fila.

Teorema 3.2. *Sea $S = \{\mathbf{x} : A\mathbf{x} = \mathbf{b} \geq \mathbf{0}\}$, siendo A una matriz $m \times n$ de rango m y $\mathbf{b}$ un vector $m \times 1$. Un vector $\mathbf{d}$ es una dirección extrema de S si y sólo si A puede descomponerse en $A = [B,N]$, donde B es una matriz $m \times m$ de rango m tal que $B^{-1}A_j \leq \mathbf{0}$ para alguna columna A_j de N y $\mathbf{d}$ es un múltiplo positivo de*

$$\hat{\mathbf{d}} = \begin{pmatrix} \hat{\mathbf{d}}_B \\ \hat{\mathbf{d}}_N \end{pmatrix} = \begin{pmatrix} -B^{-1}A_j \\ \mathbf{e_j} \end{pmatrix},$$

donde e_j es un vector de n-m ceros, excepto un 1 en la j-ésima componente.

A la vista del teorema 3.1, para calcular los puntos extremos de un recinto basta con ir tomando sucesivas factorizaciones de la matriz $A = [B,N]$ y comprobando si se cumplen las condiciones de este. Además, como consecuencia de dicha factorización el número de puntos extremos siempre es finito. Una situación similar se presenta para la obtención de las direcciones extremas de un poliedro. Una consecuencia directa de la factorización de los puntos extremos es que a lo sumo m variables toman valor no nulo $\mathbf{x}_B$ y el resto, obligatoriamente son cero, $\mathbf{x}_N = \mathbf{0}$. Dado un poliedro, a las soluciones que verifican esa condición se les denomina soluciones básicas, tal y como se enuncia a continuación.

Definición 3.4. Dado un poliedro definido en las condiciones de los teoremas anteriores y un punto $\mathbf{x}$ que verifica las restricciones $A\mathbf{x} = \mathbf{b}$, este se denominará solución básica si a lo sumo tiene m valores no nulos y los vectores columna de la matriz A asociados a las componentes no nulas son linealmente independientes. Las variables que no están obligadas a ser nulas se denominan variables básicas.

Por tanto los puntos extremos de un poliedro son soluciones básicas que además son factibles por lo que se denominan soluciones factibles básicas, SFB.

El teorema que se enuncia a continuación es el centro del desarrollo de la PL y es el que determina cuándo existe solución de un PPL y dónde se encuentra.

Teorema 3.3. *Sea el PPL (3.2) en el que el rango de la matriz A es m, su número de filas, y en el que se asume que la región de factibilidad es no vacía. Sean $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ los puntos extremos del recinto y $\mathbf{d}^1, \mathbf{d}^2, \ldots, \mathbf{d}^h$ sus direcciones extremas, si existen.*
Una condición necesaria y suficiente para que el problema tenga solución óptima es que

$$\mathbf{c}\mathbf{d}^i \geq 0, \quad i = 1, \ldots, h.$$

Además, si el problema tiene solución óptima, existe un punto extremo que es solución global del problema.

Utilizando este teorema, el algoritmo del Simplex procede calculando puntos extremos sucesivos en los que se mejora estrictamente *(ver nota 4 de la página 10)* el valor de la función objetivo. De esta forma, si no se detecta que el problema es no acotado ($\mathbf{c}\mathbf{d}^i < 0$ para un cierto i), en un número finito de iteraciones debe alcanzar el óptimo ya que el número de puntos extremos está acotado.

El esquema del funcionamiento del algoritmo del Simplex se muestra a continuación:

1. Se calcula una punto extremo inicial $\mathbf{x}$ con una cierta factorización de la matriz $A = [B,N]$ en la que por simplicidad se asume que la matriz B corresponde a las m primeras columnas de A y N al resto.

2. Se comprueba si la solución es óptima, para ello basta con calcular:

$$c_j - z_j = c_j - \mathbf{c}_B B^{-1} A_j, \forall j. \tag{3.3}$$

Si $c_j - z_j \geq 0 \; \forall j$ el punto extremo o SFB actual es óptima[4].
En otro caso, se va a hacer que una de las variables que valían cero, no básicas, pase a tomar valor y una de las que tomaban valor, variable básica, pase a ser cero. Para ello se selecciona el índice k tal que $c_k - z_k = \min_j \{c_j - z_j\} < 0$.

3. La columna de la variable k correspondiente a N pasa a la matriz B y de la matriz B se quita la columna que ocupa la posición r tal que

$$\frac{\overline{b}_r}{Y_{rk}} = \min_{1 \leq i \leq m} \left\{ \frac{\overline{b}_i}{Y_{ik}} : Y_{ik} > 0 \right\} \geq 0 \tag{3.4}$$

donde el vector Y_k se calcula a partir de la columna k-ésima de A de acuerdo a:

$$Y_k = B^{-1} A_k. \tag{3.5}$$

Si en el cálculo (3.5) todas las componentes son menores o iguales que cero, $Y_k \leq \mathbf{0}$ entonces el problema es no acotado y se puede encontrar una dirección a través de la cual llevar la función objetivo a $-\infty$ (costo marginal negativo, teorema 3.2); el algoritmo se detiene.

4. Una vez determinadas la variable entrante y saliente, se refactoriza la matriz A utilizando la nueva B y la nueva N y el proceso se repite a partir del punto 2.

La forma de seleccionar los sucesivos puntos extremos garantiza que la función objetivo no empeora y en la mayoría de las situaciones mejora[5], por tanto el algoritmo finaliza en un número finito de iteraciones.

3.2. Dualidad en Programación Lineal

El siguiente elemento importante en el desarrollo de la teoría de la programación lineal es el establecimiento de la dualidad. Asociado a cada problema de programación lineal se puede definir otro problema, también de tipo lineal, denominado problema dual, el cual tiene importantes relaciones y propiedades respecto al problema original, denominado primal, que pueden ser de gran beneficio para la toma de decisiones.

Aunque el problema dual asociado a un problema primal puede definirse para cualquier PPL en este resumen se plantea únicamente en términos del PPL en forma estándar (3.2), su problema dual asociado se muestra en (3.6).

Definición 3.5. Considerando como problema primal el problema en forma estándar de mínimo (3.2), el problema dual asociado toma la forma:

$$\begin{array}{lll} \text{maximizar} & \mathbf{b}'\mathbf{y} & \\ \text{sujeto a} & A'\mathbf{y} \leq \mathbf{c}' & \\ & \mathbf{y} \text{ no restringido,} & \end{array} \tag{3.6}$$

donde $\mathbf{y}$ de dimensión $m \times 1$ es el vector de variables duales, la nueva matriz de coeficientes tecnológicos es la matriz traspuesta de la original, el vector de costos del problema original pasa a ser el vector de recursos y viceversa, el vector de recursos pasa a ser el vector de costos. Como consecuencia de la definición de los problemas duales, el dual del problema dual vuelve a ser el problema primal.

[4] Las cantidades $c_j - z_j$ se denominan *costos marginales* de las variables x_j, por construcción estos son 0 para las variables básicas. Para las variables no básicas representan la variación de la función objetivo por cada unidad que aumente x_j.

[5] El algoritmo del Simplex puede tener problemas de convergencia si aparece un fenómeno denominado degeneración, si bien aunque pueda ralentizarlo un poco, rara vez le provoca divergencia. De hecho los problemas en los que la degeneración provoca la no convergencia del algoritmo se construyen artificialmente.

Como se ha comentado previamente, la resolución de uno de estos problemas aporta información sobre el otro, en particular en el teorema 3.4 se muestra la relación entre las soluciones de ambos.

Teorema 3.4. *Dado un par de problemas primal y su dual asociado, una y sólo una de las afirmaciones siguientes es cierta:*

1. *Ambos problemas tienen soluciones óptimas finitas y los valores de las funciones objetivo en el óptimo coinciden.*
2. *Si un problema es no acotado, su dual es no factible.*
3. *Si un problema es no factible, su dual o es no factible o es no acotado.*

Teniendo en consideración el problema primal (3.2) y su dual (3.6) y suponiendo que (3.2) tiene solución óptima finita, entonces la solución de (3.6) se obtiene directamente de esta tal y como se ve en el teorema 3.5.

Teorema 3.5. *Sea* $\mathbf{x}^*$ *una punto extremo que corresponde a una solución óptima del PPL (3.2) y sea B la matriz de la factorización correspondiente, entonces* $\mathbf{y}' = \mathbf{c}_B B^{-1}$ *es una solución óptima del problema dual (3.6).*

Con este resultado ya se puede presentar el que se utiliza en la construcción del programa para hacer la valoración de los nutrientes, es decir, qué repercusión tiene en el costo global de la fórmula cada unidad de nutriente (recordar que estos forman parte de los ingredientes y estos, los ingredientes, son los que tienen asignado un costo de forma explícita).

Sean $\mathbf{x}^*$ e $\mathbf{y}^*$ soluciones óptimas de (3.2) y (3.6), respectivamente. Considérese, sin pérdida de generalidad, la siguiente modificación del vector de recursos en la que se varía el requerimiento del primero de ellos,

$$\bar{\mathbf{b}} = \mathbf{b} + (\Delta b_1, 0, \ldots, 0)', \tag{3.7}$$

donde Δb_1 es suficientemente pequeño de forma tal que usando la factorización de la solución óptima actual se cumple[6]

$$\bar{\mathbf{x}}_B = B^{-1}\,\bar{\mathbf{b}} = B^{-1}(\mathbf{b} + (\Delta b_1, 0, \ldots, 0)') \geq \mathbf{0}. \tag{3.8}$$

Ahora se consideran los siguientes problemas primal (3.9) y su dual (3.10):

$$\begin{array}{lrcl} \text{minimizar} & \mathbf{cx} & & \\ \text{sujeto a} & A\mathbf{x} & = & \bar{\mathbf{b}} \\ & \mathbf{x} & \geq & \mathbf{0} \end{array} \tag{3.9}$$

$$\begin{array}{lrcl} \text{maximizar} & \bar{\mathbf{b}}'\mathbf{y} & & \\ \text{sujeto a} & A'\mathbf{y} & \leq & \mathbf{c}' \\ & \mathbf{y} \text{ no restringido.} & & \end{array} \tag{3.10}$$

Por construcción la solución óptima de (3.9) viene dada por (3.8) ya que los costos marginales $c_j - z_j = c_j - c_B B^{-1} A_j$ no han cambiado. La solución del problema dual (3.10) de acuerdo al teorema 3.5 es

$$\bar{\mathbf{y}}' = \mathbf{c}_B B^{-1} = \mathbf{y}. \tag{3.11}$$

Juntando los elementos anteriores con el hecho de que el problema primal y dual en el óptimo tienen el mismo valor de la función objetivo se puede concluir que:

$$Z(\bar{\mathbf{x}}) = Z(\mathbf{x}^*) + y_1 \Delta b_1. \tag{3.12}$$

Es decir, al variar el valor del primer recurso, b_1, en una cantidad Δb_1 suficientemente pequeña, el valor de la función objetivo del problema se modifica en el valor de la primera variable dual en el óptimo multiplicada por la variación realizada, $y_1 \Delta b_1$. Es por esto que el valor de las variables duales en el óptimo se puede interpretar como el *precio sombra* (implícito) del recurso correspondiente a dicha variable dual.

[6] Se supone que no existe degeneración, es decir, que $\mathbf{x}_B = B^{-1}\mathbf{b} > \mathbf{0}$.

3.3. Análisis Post-óptimo

Para finalizar este pequeño resumen de los elementos de programación lineal se presenta el análisis post-optimo de costos.

Se considera el PPL en forma estándar 3.2 y su solución óptima $\mathbf{x}^*$ con una cierta factorización $[B,N]$ de la matriz A. Al ser óptima se verificará

$$c_j - \mathbf{c}_B B^{-1} A_j = c_j - z_j \geq 0, \quad j = 1,\ldots,n.$$

Se considera ahora una modificación paramétrica del vector de costos. Sin pérdida de generalidad se modifica el primero de ellos:

$$\begin{array}{ll} \text{minimizar} & (\mathbf{c} + (\lambda, 0, \ldots, 0))\mathbf{x} \\ \text{sujeto a} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{3.13}$$

con $\lambda \in \mathbb{R}$. Si se considera la misma factorización, su solución asociada se mantendrá óptima siempre y cuando los nuevos costos marginales $\bar{c}_j - \bar{z}_j$ sigan siendo no negativos. Dichos valores ahora dependen de λ y su valor dependerá de si la variable x_1 a la que se le modifica el costo es una variable correspondiente a la matriz B o a la matriz N.

Si x_1 corresponde a una variable cuya columna está en N, variable no básica, entonces únicamente su costo marginal se modificará tras la parametrización, quedando en la forma:

$$(c_1 + \lambda) - \mathbf{c}_B B^{-1} A_j = \lambda + (c_1 - z_1)$$

y la solución seguirá siendo óptima siempre y cuando dicha expresión sea no negativa.

En el caso en el que x_1 corresponde a una variable cuya columna está en B, variable básica[7], su variación modifica todos los costos marginales y será necesario expresarlos todos en función de λ e imponer que todas las expresiones se mantengan no negativas, es decir:

$$c_j - \bar{\mathbf{c}}_B B^{-1} A_j = (c_j - z_j) + \begin{pmatrix} \lambda \\ 0 \\ \vdots \\ 0 \end{pmatrix} B^{-1} A_j \geq 0, \ \forall A_j \in N. \tag{3.14}$$

De esta forma se obtendrá el intervalo de variación de λ para el cual el valor de los costos marginales se mantiene no negativo y en consecuencia la solución de partida, la obtenida antes de plantear la modificación, sigue siendo óptima.

[7] Se supone además, sin pérdida de generalidad, que su columna es la primera de la matriz B.

Capítulo 4

Desarrollo del Programa

Este capítulo se centra en el programa, cuál es su estructura, de qué funciones dispone y cómo se ha desarrollado.

Tras arrancar el programa, se ve que este consta de tres paneles distintos: “Ingredientes y Nutrientes”, “Fórmulas” y “Tabla de ingredientes”. También presenta un menú principal con cuatro menús desplegables: “Fichero”, “Formulación”, “Informes” y “Tabla de ingredientes”, con diferentes opciones cada uno. A la derecha del menú principal se tiene un quinto menú, “Acerca de”, que abre una ventana emergente cuando clicamos sobre él, donde se encuentra una pequeña descripción de la realización del programa. La disposición de estos elementos puede verse en en la figura 4.2.

El esquema siguiente muestra las diferentes opciones que se pueden encontrar en cada uno de los menús desplegables del menú principal del programa:

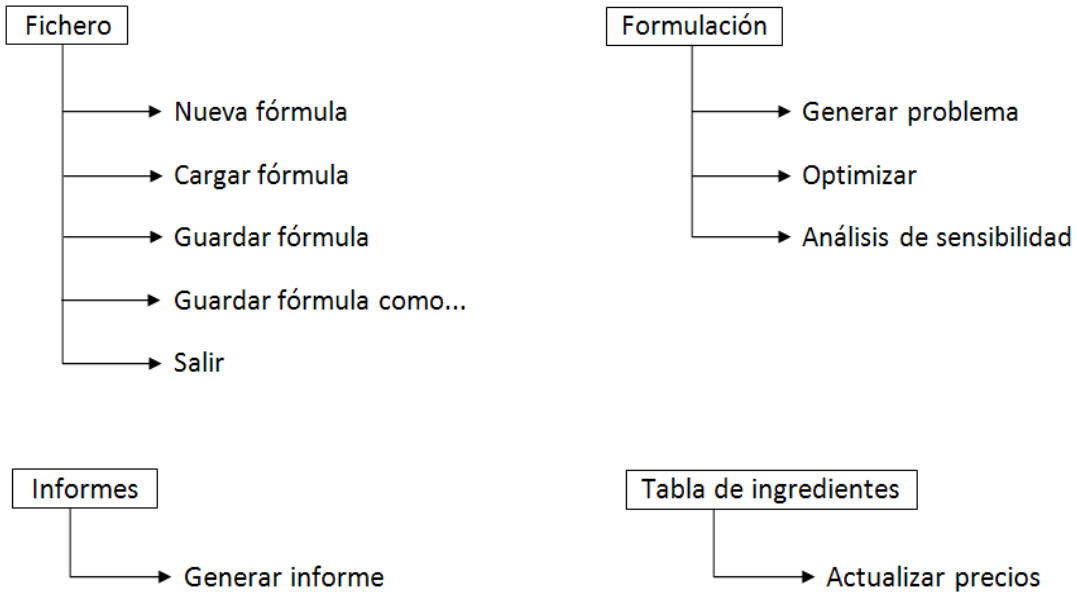


Figura 4.1: Opciones de los menús del programa.

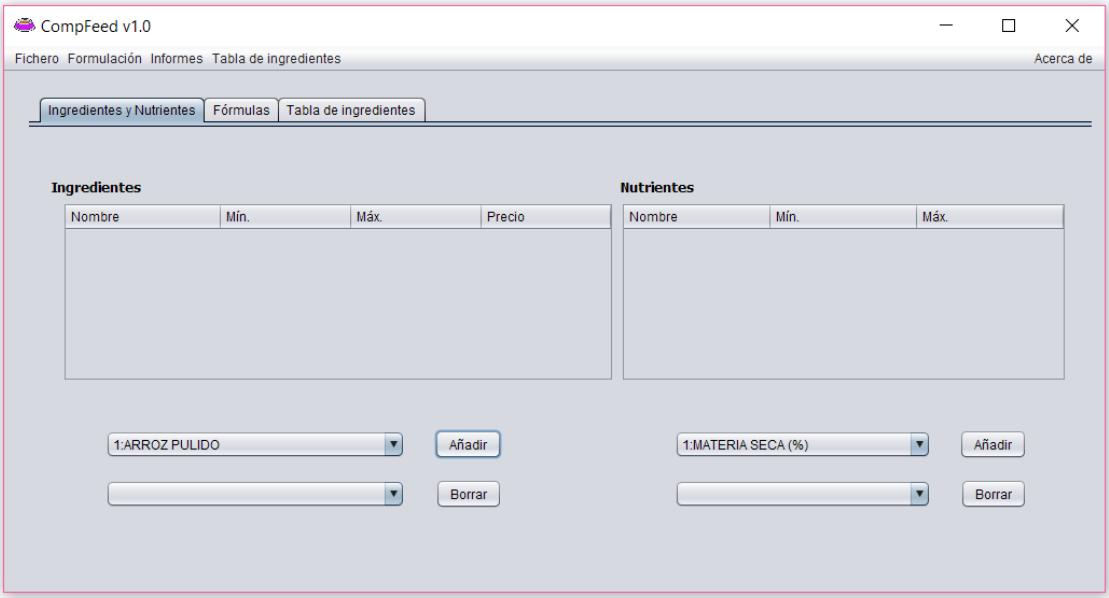


Figura 4.2: Inicio del programa.

4.1. Panel Ingredientes y Nutrientes

Es el panel que se muestra por defecto al iniciar el programa. Consta de dos tablas, una para añadir los ingredientes que se quieren considerar para la creación de nuestra dieta, y otra para añadir los nutrientes necesarios para la misma. Cada tabla tiene asociados dos menús desplegables. El primero de ellos almacena todos los ingredientes de la base de datos y permite añadir a la tabla asociada el ingrediente que se selecciona. Una vez se añade a la tabla, este ingrediente desaparece del primer menú y pasa a almacenarse en el segundo, desde el que se podrá seleccionar más adelante para borrarlo de la tabla si interesase. Si se selecciona borrar un ingrediente, este desaparecería tanto de la tabla como del menú de borrar, y volvería a aparecer en el menú de añadir. Los menús asociados a la tabla de nutrientes funcionan de forma análoga.

La tabla de ingredientes está formada por cuatro columnas, una para el nombre del ingrediente, otra para su contenido mínimo, otra para su contenido máximo y una última para su precio. La tabla de nutrientes está formada solo por tres columnas, una para el nombre del nutriente, otra para su contenido mínimo y otra para su contenido máximo.

Cuando se añade un ingrediente o nutriente a las tablas, el resto de columnas se rellenan con unos datos iniciales que deberán modificarse en función de las restricciones que haya que considerar para la fórmula que se va a crear.

4.2. Panel Fórmulas

Este panel incorpora un área de texto donde se van imprimiendo todos los resultados de la resolución de la fórmula. Primero, si se le indica, aparecerá el problema de programación lineal generado, después los resultados obtenidos tras su optimización y por último las conclusiones del análisis post-óptimo. Más adelante se explicarán más a fondo los resultados que van mostrándose en esta pestaña.

4.3. Panel Tabla de Ingredientes

El tercer panel incluye la base de datos. Es una tabla con todos los ingredientes disponibles y las composiciones nutricionales correspondientes. Los datos aquí disponibles son solo de lectura, no pueden modificarse, aunque sí puede variarse la posición de las columnas para facilitar la localización y visualización de los datos que sean de interés. Desde aquí es posible añadir ingredientes y nutrientes a sus correspondientes tablas simplemente haciendo doble clic sobre el nombre del ingrediente o nutriente que se quiera añadir.

4.4. Opción Fichero

Esta primera opción, como puede verse en el esquema gráfico del principio del capítulo, incluye otras cinco opciones. La primera de todas, "Nueva fórmula", se encuentra inicialmente deshabilitada. Se encontrará operativa únicamente cuando alguna de las tablas no esté vacía. Permite guardar la fórmula que está editándose actualmente antes de vaciar ambas tablas para poder crear una nueva o simplemente descartarla y comenzar la formulación desde cero.

La opción "Cargar fórmula" permite cargar alguna dieta formulada anteriormente y que se encuentre guardada en un fichero .die. Al seleccionar el fichero que se desee abrir, las tablas de ingredientes y nutriente se llenarán automáticamente con los datos correspondientes y ya será posible trabajar sobre esa fórmula.

Mediante la opción "Guardar fórmula", si se está trabajando sobre un fichero ya existente cargado previamente, se guardará la nueva fórmula en el mismo fichero sobrescribiendo los datos anteriores. Si por el contrario, se está creando una nueva fórmula desde cero, esta opción funciona de la misma manera que la opción "Guardar fórmula como..."; lo que ambas hacen es abrir una ventana emergente que da opción de guardar la fórmula donde se seleccione y de darle un nombre al nuevo fichero. A diferencia de "Guardar fórmula", la opción de "Guardar fórmula como..." crea un nuevo fichero con el nombre que se elija, ya se esté creando una fórmula desde el principio o trabajando sobre un fichero ya existente.

4.5. Opción Formulación

Esta segunda opción de menú incluye tres opciones, todas deshabilitadas al inicio del programa. La primera opción, "Generar problema", pasa a estar disponible cuando al menos una de las tablas contiene algún elemento. Cuando se selecciona generar el problema, aparece la opción de mostrar o no la formulación en el área de texto del panel "Fórmulas". Se incluye esta posibilidad de elección ya que matemáticamente puede interesar cómo se construyen las diferentes restricciones del problema para la posterior aplicación del algoritmo del Simplex.

Una vez generado el problema, se activa la opción "Optimizar". Al seleccionar esta opción, si no se encuentra una solución óptima, aparecerá una ventana de aviso indicando que el problema es no factible. En caso contrario, se añade al área de texto el costo de la solución, es decir el precio por kilogramo de pienso. A continuación se muestran los ingredientes utilizados y el tanto por ciento que cada uno de ellos aporta a la composición total. Después de esto se muestran los ingredientes que no han sido utilizados para la formulación del pienso deseado, si es que ha habido alguno que se ha descartado; y por último aparecen los aportes de nutrientes.

La opción "Análisis de sensibilidad" se encuentra operativa una vez que el problema ha sido optimizado y muestra en el área de texto el análisis post-óptimo. Primero muestra los diferentes rangos en los que pueden variar los precios de cada ingrediente sin que se modifique la formulación actual, y después muestra el incremento del costo que supondría incluir una unidad adicional de cierto nutriente,

si este satisface su requerimiento mínimo, o disminuir en una unidad cierto nutriente, si este satisface su requerimiento máximo. En el caso de que la cantidad que se incluye del nutriente no sea la mínima ni la máxima exigidas, el precio no se modificaría y obviamente no se muestra.

4.6. Opción Informes

Después de haber realizado todo lo anterior, existe la posibilidad de guardar los resultados obtenidos en un documento Excel. Cuando se selecciona "Generar informe", el programa permite exportar los datos a un documento nuevo de Excel o a uno ya existente sobrescribiendo los datos que tuviese anteriormente. El informe consta de tres hojas: "Fórmula", "Análisis de sensibilidad" y "Distribución de nutrientes". En la primera se muestra el costo de la solución, la partida en kilogramos seleccionada, los ingredientes seleccionados con sus respectivas restricciones de mínimo y máximo, la cantidad que aportan en porcentaje y también en kilogramos. También se muestra la composición lograda, es decir, los nutrientes utilizados con sus respectivas restricciones de mínimo y máximo junto con la cantidad que habrá presente de cada uno en nuestro pienso. En la segunda se muestra el análisis de sensibilidad de ingredientes: su precio y el rango de variación de este en el que la solución del problema sigue siendo óptima; y de nutrientes: la cantidad y el incremento en el precio que supondría la modificación del contenido en una unidad. La tercera contiene la información de la pestaña "Ingredientes y Nutrientes".

4.7. Opción Tabla ingredientes

Cuenta con una única opción, "Actualizar precios" que permite actualizar la base de datos desde el propio programa. Conecta con la base de datos y almacena los precios que se han introducido para cada ingrediente. Asimismo se almacenan también en la tabla de la pestaña "Tabla de ingredientes".

4.8. Clases implementadas. Funcionamiento

El programa desarrollado consta de una clase principal, "Ventana", y otras siete clases: "baseDeDatos", "datosGlobales", "Ingrediente", "Nutriente", "PPLs", "GeneraExcel" y "AcercaDe". En lo que sigue se va a hablar de la implementación de cada una de ellas y qué es lo que hacen.

4.8.1. Clase Ventana

Esta es la clase principal. En el constructor de la clase Ventana se inicializan y construyen todos los elementos que constituyen la interfaz gráfica: la barra de menú principal con los diferentes menús que contiene, los paneles sobre los que se crean las tablas con los correspondientes menús desplegables para la selección de ingredientes y nutrientes, los botones para añadirlos y eliminarlos de dichas tablas, y el área de texto donde se van imprimiendo los resultados obtenidos.

Se conecta con la base de datos y se inicializan unos arrays estáticos donde se irán almacenando los nombres de los diferentes ingredientes y nutrientes que componen la tabla FEDNA, así como una matriz con los datos nutricionales de cada ingrediente. Después, de estas listas se vuelcan los datos a las diferentes componentes del programa: los nombres de los ingredientes y nutrientes a sus correspondientes menús desplegables para posteriormente poder seleccionarlos, y toda la información nutricional a la tabla que se encuentra en la pestaña "Tabla de ingredientes".

4.8.2. Clase baseDeDatos

Esta clase permite la conexión con la base de datos utilizada, extraída de la tabla FEDNA e incluida en la carpeta del programa. Esta conexión se lleva a cabo, como ya se ha dicho, mediante la librería

SQLite JDBC. De la base de datos se obtiene el número total de ingredientes y nutrientes, las listas de los mismos con sus correspondientes códigos escritos precediendo a los nombres y en el caso de los nutrientes seguidos entre paréntesis de la unidad en la que se miden, además de la matriz con todos los datos de composición de nutrientes.

4.8.3. Clase datosGlobales

En esta clase es donde se crean e inicializan los elementos estáticos de forma que sean accesibles a través de la clase y puedan utilizarse como variables globales accesibles desde cualquier parte del programa. Estos elementos son los vectores que almacenan las listas de ingredientes y nutrientes, la matriz que almacena las componentes nutricionales de todos los ingredientes, la cadena que almacena el nombre del fichero que se crea para guardar una fórmula de dieta y una variable booleana (true/flase) que se utiliza para saber si el fichero que quiere guardarse ya existe y en ese caso sobrescribirlo o crear uno nuevo. También se crean unas listas estáticas de tipo “Ingrediente” y “Nutriente” donde se almacenan los ingredientes/nutrientes y sus datos asociados de las restricciones que se imponen al crear la fórmula.

4.8.4. Clase Ingrediente y clase Nutriente

Estas clases recogen los datos que se almacenarán en los arrays estáticos correspondientes a cada una. En el caso de la clase Ingrediente se almacenan como cadenas de caracteres el código del ingrediente, el nombre, la restricción de mínimo, la de máximo y el precio. En la clase Nutriente se almacenan de igual modo el código del nutriente, el nombre, la restricción de mínimo y la de máximo. Cuando se seleccione la opción de guardar, de estos arrays se irá seleccionando y guardando cada componente una a continuación de la otra, separadas por una doble barra, para guardarlas en un archivo de propiedades generado mediante la clase `util.Properties`.

```
NombreDeDieta=C\:\\Users\\aaguilar\\gallineas.die
NumIngredientes=20
NumNutrientes=13
Ingrediente1=4\:CEBADA 2C 11.3 PB//0//100//0.168//1
Ingrediente2=7\:MAIZ NACIONAL//0//100//0.178//1
...
Nutriente1=1\:MATERIA SECA (%)//85//100
Nutriente2=50\:EMA_AVES Kcal/Kg//2740//5000
...
```

Figura 4.3: Formato del archivo de fórmulas.

4.8.5. Clase PPLs

Esta clase es la encargada de optimizar el PPL y hacer el análisis post-óptimo. Presenta tres funciones principales: `resuelveProblema, escribeSolucion, análisiDeSensibilidad`. Aquí destaca la utilización de las rutinas de la librería GLPK y vamos a hablar de las más destacadas [6]. El constructor de la clase toma como argumentos la tabla de ingredientes, la de nutrientes y el área de texto donde se imprimen los resultados del problema.

Antes de implementar estas funciones, se crea el PPL, que inicialmente está vacío, mediante la rutina `GLPK.glp_create_prob` y a continuación se establecen las variables del mismo, sus restricciones y la función objetivo.

Se establece el número de columnas / filas del problema con la rutina `GLPK.glp_add_cols` / `GLPK.glp_add_rows`. Tendrá tantas columnas / filas como variables estructurales / auxiliares haya[1]. También se establece el tipo de las variables con `GLPK.glp_set_col_kind`, pudiendo ser estas de tipo continuo, entero o binario. En el presente trabajo serán de tipo continuo.

Precediendo a las restricciones introducidas para cada variable está la restricción de que todas las cantidades de ingredientes sumen 1, que se considera el total de la mezcla a la hora de la resolución. La variable auxiliar asociada a esta restricción será de tipo fijo, lo que se establece a través de la rutina `GLPK.glp_set_row_bnds`. Para las variables estructurales utilizamos la rutina análoga, `GLPK.glp_set_col_bnds`, que establecerá sólo si tienen cotas inferiores o superiores. Por último se crea una matriz donde se almacenan las restricciones del problema, con `GLPK.glp_set_mat_row`.

En relación a la función objetivo, lo que se hace es establecer si el problema es de mínimo o de máximo con la rutina `GLPK.glp_set_obj_dir` (de mínimo en nuetro caso) y también el coeficiente de las variables estructurales en dicha función: `GLPK.glp_set_obj_coef`.

La función `resuelveProblema` inicializa los parámetros de control y mediante el uso de la rutina `GLPK.glp_simplex` recupera los datos del problema especificado, lo resuelve por el método Simplex y almacena los resultados en el problema. Tras la resolución, para determinar si ha encontrado una solución óptima, se consulta el estado del problema mediante `GLPK.glp_get_status(lp)==GLP_OPT` (siendo `lp` nuestro PPL). En otro caso se considera que es no factible[2].

Con la función `escribeSolucion` el programa imprime en el área de texto del panel "Fórmulas" todos los resultados que se obtienen en la resolución del problema. Primero imprime el costo de la solución óptima, los ingredientes utilizados, los no utilizados y la composición de nutrientes lograda. Para los ingredientes utilizados y no utilizados se cuenta con la rutina que devuelve el valor primal de las variables estructurales, `GLPK.glp_get_col_prim`; si el valor es distinto de cero, entonces las variables son básicas y se corresponden con los ingredientes que se utilizan. Para la composición de nutrientes se imprimen sus cotas inferiores y superiores además del valor que toman, que será el valor primal de las variables auxiliares.

Por último, la función `análisisDeSensibilidad` indica los rangos en que pueden variar los precios de las variables estructurales, es decir los ingredientes, y que la solución óptima obtenida se mantenga. Cabe destacar tres casos:

- Si la variable es básica, utilizamos la rutina `GLPK.glp_analyze_coef`, que es la que analiza el efecto de variar el coeficiente en la función objetivo de una variable básica. Podrá variar entre dos coeficientes que nos devuelve esta misma rutina.
- Si la variable es no básica y toma el valor de su cota inferior, el rango de variación será entre el valor en la función objetivo de dicha variable menos el costo marginal de la misma, `GLPK.glp_get_obj_coef-GLPK.glp_get_col_dual`, e infinito.
- Si la variable es no básica y toma el valor de su cota superior, el rango de variación de dicha variable será entre menos infinito y el valor en la función objetivo menos el costo marginal de la misma.

[1] Las variables estructurales se corresponden con los ingredientes de la fórmula y las variables auxiliares con las ecuaciones que establecen la relación de los ingredientes con su composición de nutrientes. En el Anexo A se describe el formato de los PPL en GLPK.

[2] Por construcción el problema no puede ser no acotado.

También indica lo que supondría para el costo del pienso la variación en una unidad de un determinado nutriente. Aquí distinguimos dos casos: si la variable es no básica y toma el valor de su cota inferior, se calcula el incremento en el costo que supondría una unidad adicional de la cantidad indicada, y si toma el valor de su cota superior, se calcula el incremento en el costo que supondría una unidad menos de la cantidad indicada. En ambos casos se indica cuál sería el incremento en el precio mediante la rutina `GLPK.glp_get_row_dual`, que es la que devuelve el valor dual de las variables auxiliares.

4.8.6. Clase GeneraExcel

Esta clase es la encargada de generar un informe en un archivo Excel donde se almacenen todos los resultados obtenidos a lo largo de la resolución del PPL. El constructor de la clase toma como argumentos la tabla de ingredientes, la de nutrientes y la clase PPLs.

Tras resolver el PPL mediante el algoritmo del Simplex, se crean las tres hojas que se han comentado en la sección 4.6 y en cada una de ellas se van creando las sucesivas filas y celdas correspondientes a cada fila en las que van a almacenarse todos los datos. Las funciones principales utilizadas son las que se muestran en la figura 4.4:

```
Sheet hoja = libro.createSheet("Nombre hoja"); // Crea una hoja con el nombre
    deseado en el documento Excel
fila = hoja.createRow(i); // Crea la fila i-ésima en la hoja actual
fila=hoja.getRow(i); // Selecciona la fila i-ésima en la hoja actual
celda = fila.createCell(j); // Crea la celda j-ésima en la fila actual (i)
celda.setCellValue("Valor"); // Establece el valor deseado a la celda actual
```

Figura 4.4: Funciones para la creación del documento Excel.

4.8.7. Clase AcercaDe

En esta última clase se crea una nueva ventana formada por un panel que contiene un área de texto donde aparece información sobre la realización del programa.

Capítulo 5

Conclusiones y Trabajos Futuros

Finalmente, en este último capítulo, vamos a comentar las conclusiones obtenidas con la realización de este trabajo y posibles cuestiones pendientes que podrían desarrollarse en un futuro para mejorar y completar la implementación del programa.

5.1. Conclusiones

Los objetivos que nos planteamos al inicio de este trabajo se han logrado satisfactoriamente. Vamos a enumerar estos objetivos y a comentar los resultados y conclusiones obtenidos:

1. Desarrollar un programa informático destinado a la formulación de piensos compuestos de animales monogástricos.

Tal y como se pretendía, hemos conseguido desarrollar una aplicación informática con suficiente potencia para poder llevar a cabo las formulaciones requeridas. Con la ayuda del segundo director, experto nutricionista, se ha ido comprobando que la implementación era la adecuada y que proporcionaba soluciones válidas.

2. Afrontar un problema real, en el que se van a tener que aunar distintos conocimientos adquiridos en el grado para poder lograr el objetivo anterior.

En la realización de este trabajo tenía un gran peso que pudiese hacerse en torno a una situación real para realzar la importancia de las aplicaciones de las matemáticas y para que contase con un componente más original.

3. Realizar un software de fácil uso para que se pueda utilizar dedicando un tiempo mínimo de aprendizaje por parte de los usuarios, sobre todo pensando en su uso en docencia.

Tanto la interfaz del programa como su funcionamiento hacen que este sea fácil de usar y no requiera mucho tiempo de aprendizaje. No son necesarios conocimientos matemáticos ni de programación para su utilización, tan solo conocimientos sobre formulación de piensos a la hora de introducir las restricciones de ingredientes y nutrientes para que la formulación que vaya a obtenerse sea adecuada para el animal en cuestión.

En el Anexo B hemos incluido una serie de imágenes sobre la resolución de un problema para mostrar el funcionamiento del programa.

La conclusión final de este trabajo es que un software como el desarrollado para la formulación de piensos compuestos permite acercar una plataforma antes prácticamente inaccesible por su alto coste económico a un público del ámbito educativo.

Durante el desarrollo del Trabajo Fin de Grado, además de las nuevas habilidades y tecnologías que hemos aprendido, así como conocimientos sobre generación de dietas, como se dijo en el capítulo introductorio se han aplicado diversos conocimientos adquiridos a lo largo del grado. Sobre todo los relacionados con la metodología de programación y con la programación lineal.

A título personal, este trabajo me ha resultado muy interesante y motivador dado que está diseñado para ser utilizado como herramienta docente en la Universidad y he podido ver en primera persona las aplicaciones y utilidad que pueden tener las matemáticas.

5.2. Trabajo Futuro

El trabajo realizado a lo largo de este proyecto deja las puertas abiertas a posibles ampliaciones futuras. Estas ampliaciones consisten básicamente en la mejora de la funcionalidad del programa. Algunas de las ideas podrían ser:

- Creación de nuevas restricciones, por ejemplo ratios de nutrientes, proporciones, incompatibilidades, calidad, etc.
- Incorporación de nuevas bases de datos con diferentes valores nutritivos y materias primeras de los considerados en las tablas FEDNA.
- Inclusión de un menú de configuración del programa, en el que poder cambiar colores, tipos de letra, etc.
- Posibles extensiones a otros paradigmas como pueden ser problemas multi-objetivo en los que se trata de optimizar más de un criterio simultáneamente, por ejemplo, minimizar el costo de la fórmula y simultáneamente maximizar el aporte de Kcal/Kg.

Bibliografía

[1] M.A. BARBIERI, G.CUZON, *Improved nutrient specification for linear programming of penaeid rations*, *Aquaculture*, 19: 313–323, 1980.

[2] C. DE BLAS, G.G. MATEOS, P. GARCÍA-REBOLLAR, *Tablas FEDNA de composición y valor nutritivo de alimentos para la fabricación de piensos compuestos*, Fundación Española para el Desarrollo de la Nutrición Animal, 2016. `http://www.fundacionfedna.org/ingredientes-para-piensos`.

[3] S. CHAKEREDZA, F.K. AKINNIFESI, O.C. AJAYI, G. SILESHI, S. MNGOMBA, F.M.T. GONDWE, *A simple method of formulating least-cost diets for smallholder dairy production in sub-Saharan Africa*, African J. Biotechnol., 7(16): 2925, 2008.

[4] EUROPEAN FEED MANUFACTURERS' FEDERATION, FEFAC. *Feed & Food*. Statistical Yearbook, 2004.

[5] S. GHOSH, J. GHOSH, D.T. PAL, R. GUPTA. *Current Concepts of Feed Formulation for Livestock using Mathematical Modeling*. Animal Nutrition and Feed Technology, 2014.

[6] GNU LINEAR PROGRAMMING KIT. *Reference Manual for GLPK Version 4.58*. Draft, 2016.

[7] E.O. HEADY, J.L. DILLON. *Agricultural production functions*. Iowa State University Press, Ames, Iowa, 1961.

[8] E.O. HEADY, N.L. JACOBSON, A.E. FREEMAN, J.P. MADDEN, *Milk production functions incorporating variables for cow characteristics and environment*, J. Farm Econ., 46(1): 1–19, 1964.

[9] F.S. HILLIER, G.J. LIEBERMAN, *Introduction to Operations Research*, 8ª edición, McCrawHill, Boston, 3–4, 24–160, 2005.

[10] NETBEANS, *NetBeans IDE*, `https://netbeans.org/features/index.html`.

[11] J.D. O'CONNOR, C.J. SNIFFEN, D.G. FOX, R.A. MILIGAN, *Least cost dairy cattle ration formulation model based on the degradable protein system*, J. Dairy Sci., 72: 2733–2745, 1989.

[12] ORACLE, *Java Documentation*, `http://docs.oracle.com/javase/8/`.

[13] J. DE SAJA, *Aportación de la industria de la alimentación animal*, Jornada Informativa MAGRAMA - Ministerio de Agricultura, Alimentación y Medio Ambiente, Madrid, 2011.

[14] D. SKLAN, I. DARIEL, *Diet planning for humans using mixed-integer linear programming*, Br. J. Nutr., 70: 27, 1993.

[15] SQLITE, `https://www.sqlite.org/`.

[16] J.C. SURRA, *Apuntes de fromulación y tecnologías de la fabricación de piensos*, Servicio de reprografía EPSH - Universidad de Zaragoza, 2016.

[17] L.O. TEDESCHI, D.G. FOX, L.E. CHASE, AND S.J. WANG, *Whole-herd optimization with the Cornell Net Carbohydrate and Protein System: Predicting feed biological values for diet optimization with linear programming*, Poult. Sci., 83: 147–151, 2004.

[18] THE APACHE SOFTWARE FUNDATION, *Apache POI - the Java API for Microsoft Documents*, `https://poi.apache.org/`.

[19] M.J. VANDEHAAR, J.R. BLACK, *Ration formulation using linear programming*, Vet. Clin. North Am. Food Anim. Pract., 7(2): 541–556, 1991.

[20] F.V. WAUGH, *The Minimum-Cost Dairy Feed (An Application of "Linear Programming"*, Journal of Farm Economics, 3(3): 299–300, 1951.

Anexos

Anexo A

Formato de PPL en GLPK

En este anexo vamos a explicar la forma en que GLPK formula el problema de programación lineal para su resolución. La estructura es la siguiente:

$$
\begin{array}{lrcllllll}
\text{minimizar} & Z & = & c_1x_{m+1} & + & c_2x_{m+2} & + \ \ldots + & c_nx_{m+n} & + \ c_0 \\
\text{sujeto a} & x_1 & = & a_{11}x_{m+1} & + & a_{12}x_{m+2} & + \ \ldots + & a_{1n}x_{m+n} & \\
 & x_2 & = & a_{21}x_{m+1} & + & a_{22}x_{m+2} & + \ \ldots + & a_{2n}x_{m+n} & \\
 & & & & & & \ldots & & \\
 & x_m & = & a_{m1}x_{m+1} & + & a_{m2}x_{m+2} & + \ \ldots + & a_{2mn}x_{m+n} &
\end{array}
$$

$$
\begin{array}{rcccl}
l_1 & \leq & x_1 & \leq & u_1 \\
l_2 & \leq & x_2 & \leq & u_2 \\
 & & \ldots & & \\
l_{m+n} & \leq & x_{m+n} & \leq & u_{m+n}
\end{array}
$$

donde las variables x_1, x_2,..., x_m se denominan variables auxiliares o también llamadas filas, y las variables x_{m+1}, x_{m+2},..., x_{m+n} son las variables estructurales o columnas, es decir las dadas en el problema. Todas las variables cuentan con cotas inferiores: l_1, l_2,..., l_{m+n} y cotas superiores: u_1, u_2,..., u_{m+n}, siendo posible que estas sean respectivamente $-\infty$, $+\infty$, o que sean iguales y en ese caso la variable sea fija. Igual que se definió en el Capítulo 3, Z es la función objetivo, c_1, c_2,..., c_n los coeficientes de la misma y c_0 su término constante. $a_{11}, a_{12}, \ldots, a_{mn}$ son los coeficientes de restricción.

Para comprender mejor esta forma de definir el problema, vamos a considerar un ejemplo sencillo. En este ejemplo hemos seleccionado los ingredientes maíz nacional y harina de soja 44, que se corresponden con las variables estructurales x_3 y x_4, y los nutrientes materia seca y ema_aves, cuyas restricciones se corresponden con las variables auxiliares x_1 y x_2:

$$
\begin{array}{lrcllll}
\text{minimizar} & Z & = & 0{,}14831x_3 & + & 0{,}2737x_4 \\
\text{sujeto a} & 1 & = & x_3 & + & x_4 \\
 & x_1 & = & 86{,}2x_3 & + & 88x_4 \\
 & x_2 & = & 3280x_3 & + & 2200x_4
\end{array}
$$

$$
\begin{array}{rcccl}
85 & \leq & x_1 & \leq & 100 \\
3000 & \leq & x_2 & \leq & 5000 \\
0 & \leq & x_3 & \leq & 1 \\
0{,}2 & \leq & x_4 & \leq & 1.
\end{array}
$$

A continuación resta a las variables auxiliares unas nuevas variables asociadas e iguala a las cotas inferiores correspondientes a cada variable auxiliar, es decir:

$$\begin{array}{rcrcrcl} 86{,}2x_3 & + & 88x_4 & - & \tilde{x_1} & = & 85 \\ 3280x_3 & + & 2200x_4 & - & \tilde{x_2} & = & 3000 \end{array}$$

y así el problema a resolver es el siguiente:

$$\begin{array}{rcrcrcrcl} \text{minimizar} & Z & = & 0{,}14831x_3 & + & 0{,}2737x_4 & & & \\ \text{sujeto a} & 1 & = & x_3 & + & x_4 & & & \\ & \tilde{x_1} & = & 86{,}2x_3 & + & 88x_4 & - & 85 & \\ & \tilde{x_2} & = & 3280x_3 & + & 2200x_4 & - & 3000 & \end{array}$$

$$\begin{array}{rcrcl} 0 & \leq & \tilde{x_1} & \leq & 15 \\ 0 & \leq & \tilde{x_2} & \leq & 2000 \\ 0 & \leq & x_3 & \leq & 1 \\ 0{,}2 & \leq & x_4 & \leq & 1. \end{array}$$

Anexo B

Imágenes del Programa

En este anexo incluiremos una serie de imágenes que muestren el programa en cada paso que se va realizando al crear una fórmula y los resultados finales. Se ha tomado como ejemplo el fichero `gallineas.die`.

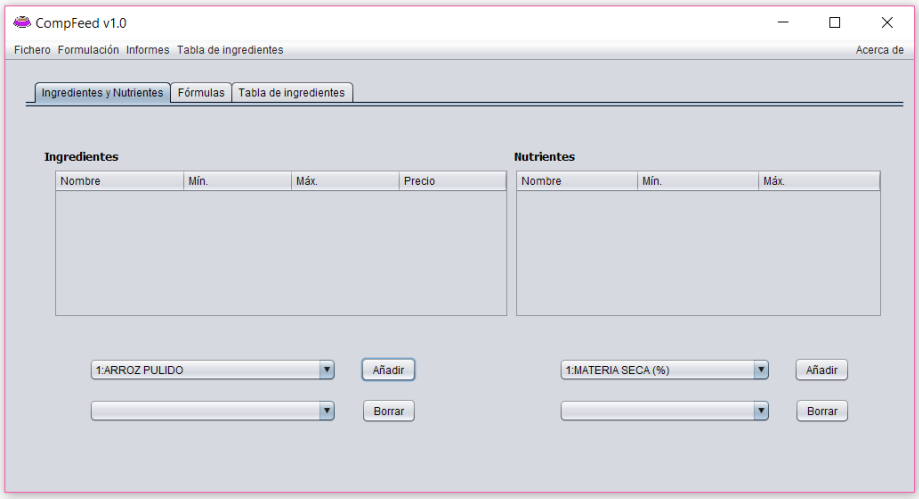


Figura B.1: Inicio del programa.

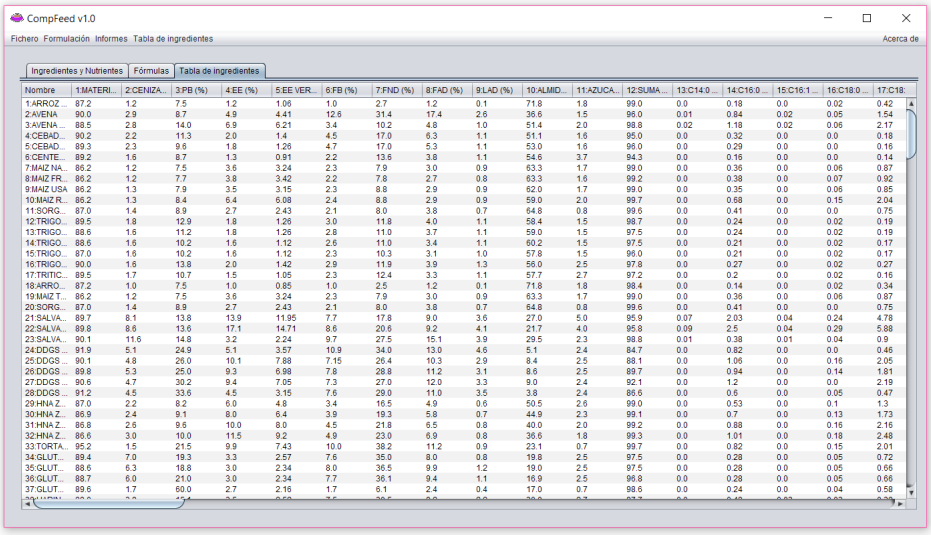


Figura B.2: Tabla de valores nutricionales.

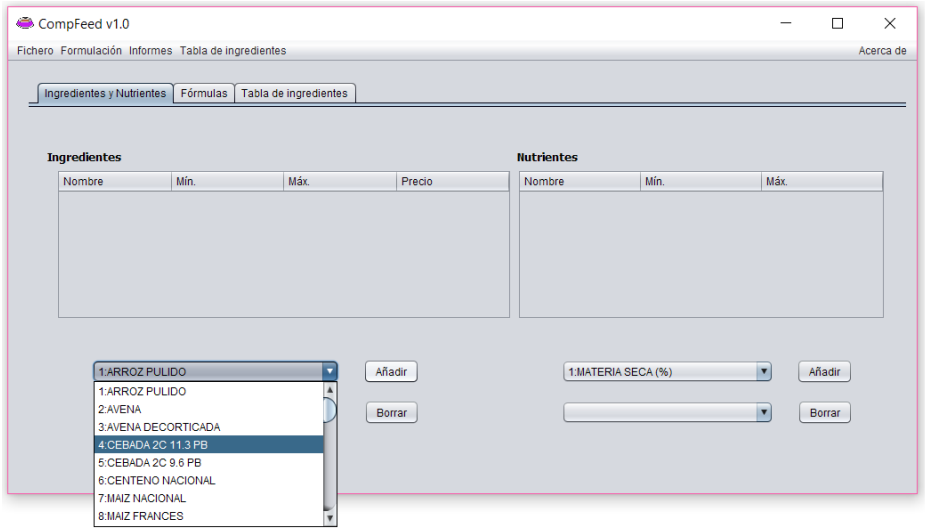


Figura B.3: Añadir ingredientes.

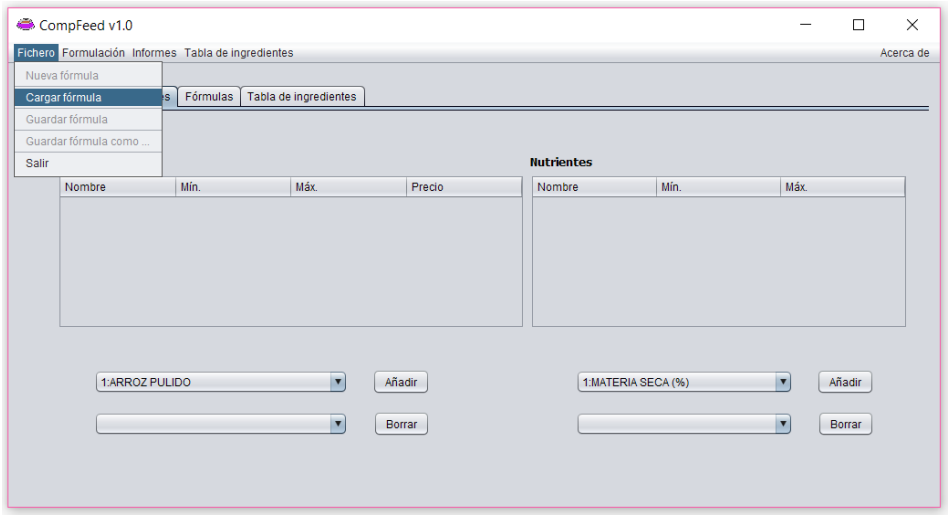


Figura B.4: Cargar fórmula.

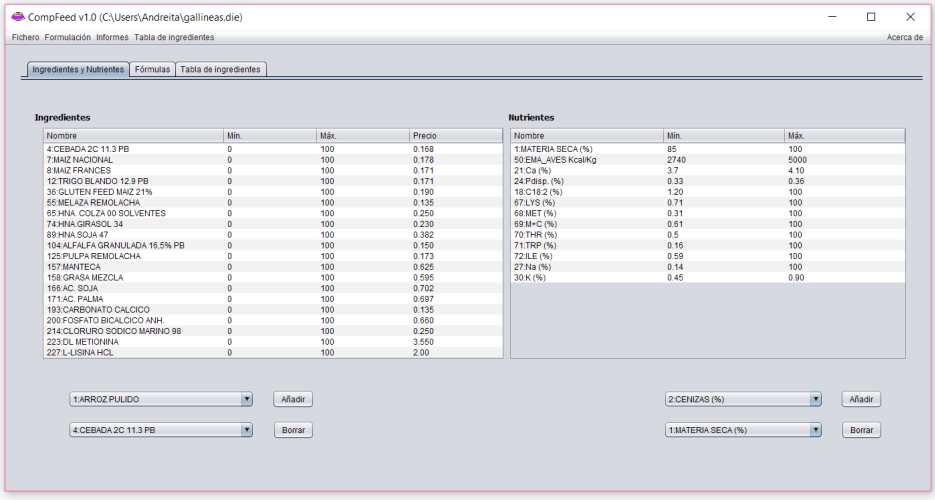


Figura B.5: Fórmula cargada.

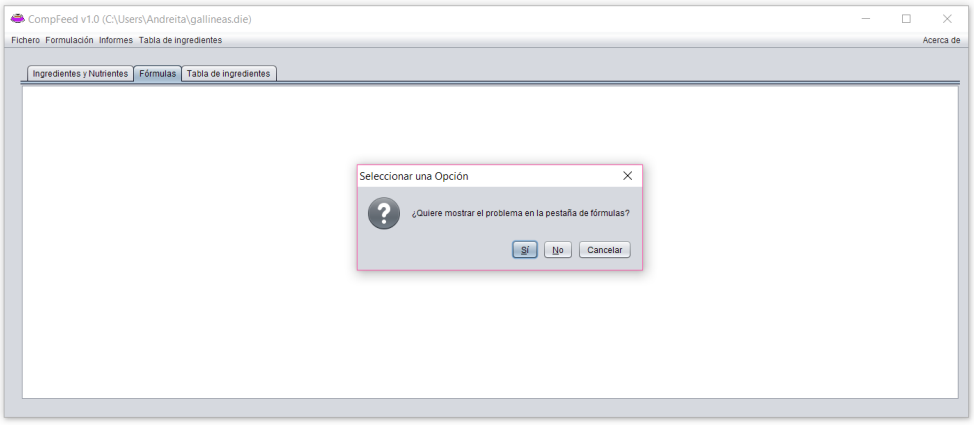


Figura B.6: Opción de generación del problema.

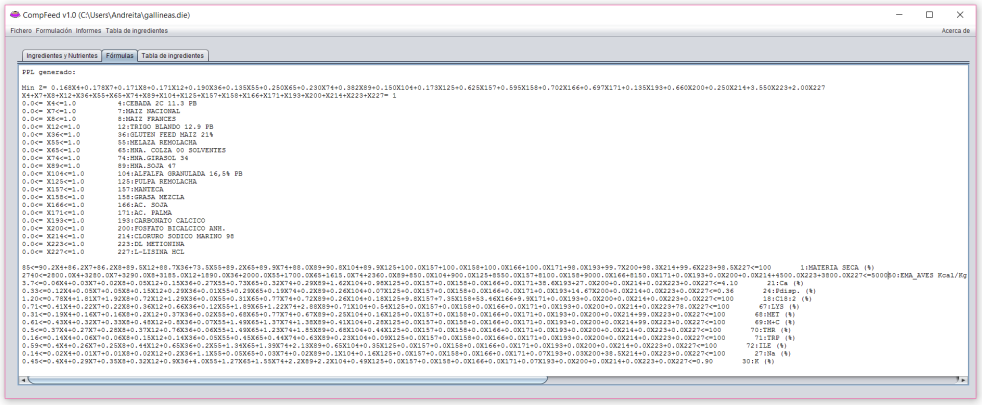


Figura B.7: Problema generado.

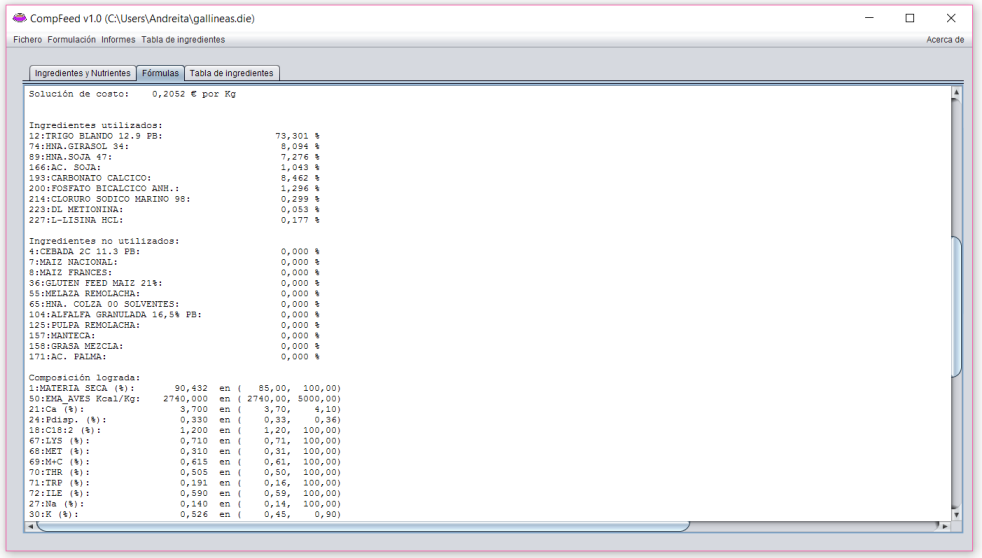


Figura B.8: Resultado de la optimización del problema.

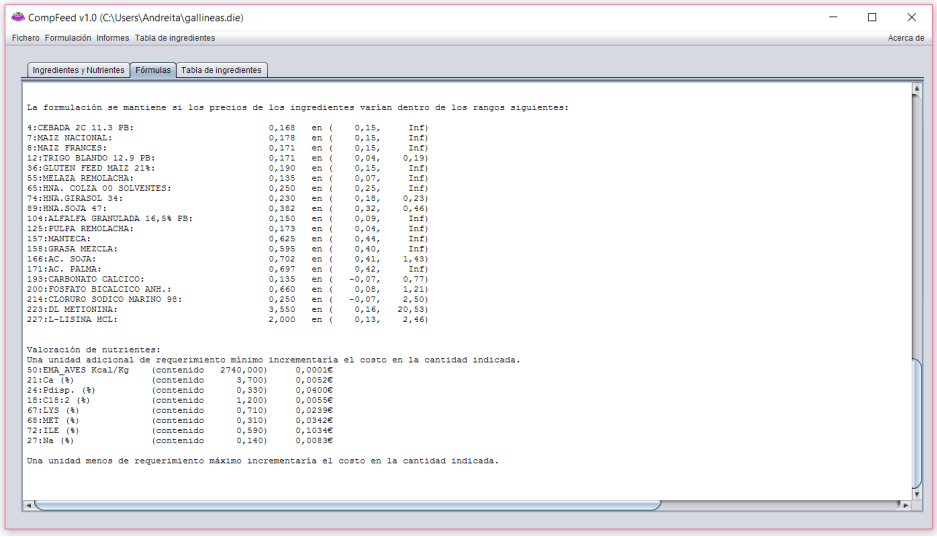


Figura B.9: Análisis de sensibilidad del problema.

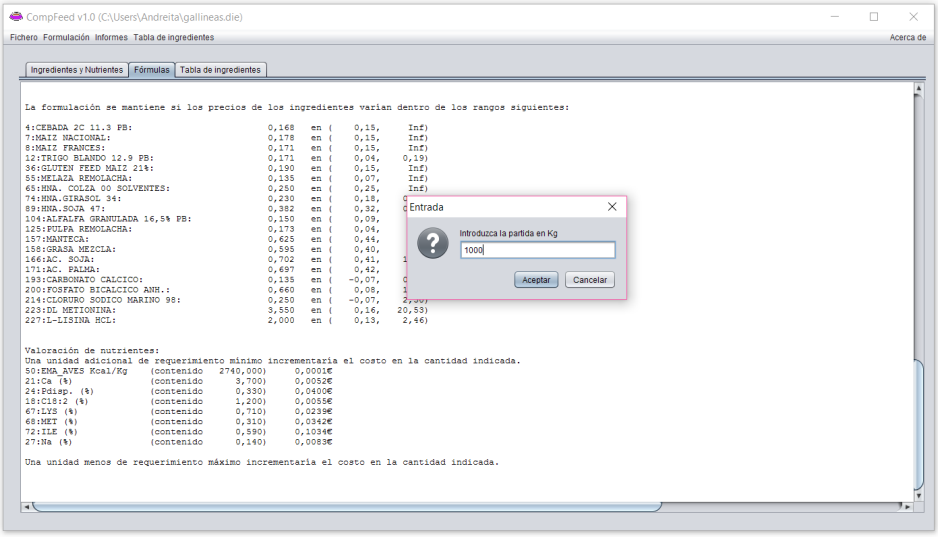


Figura B.10: Introducir partida en Kg para generar Excel.

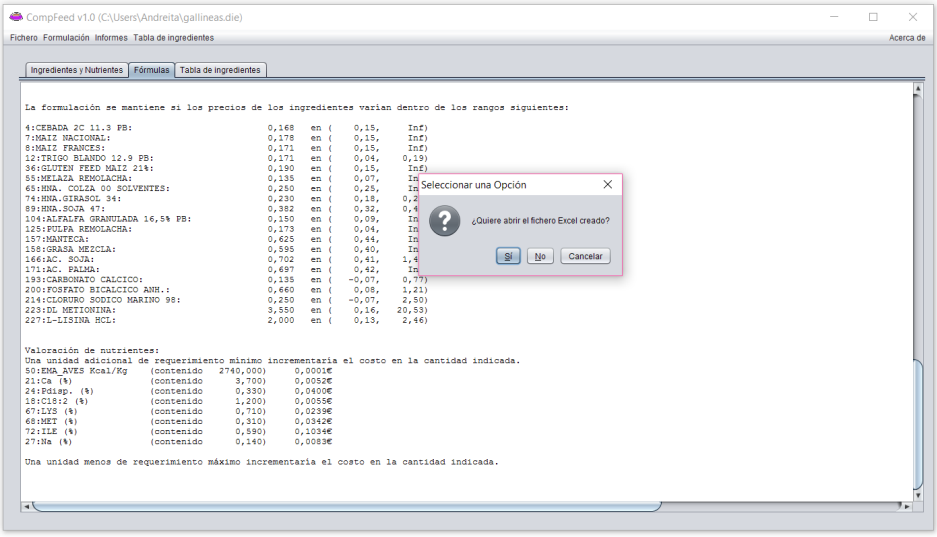
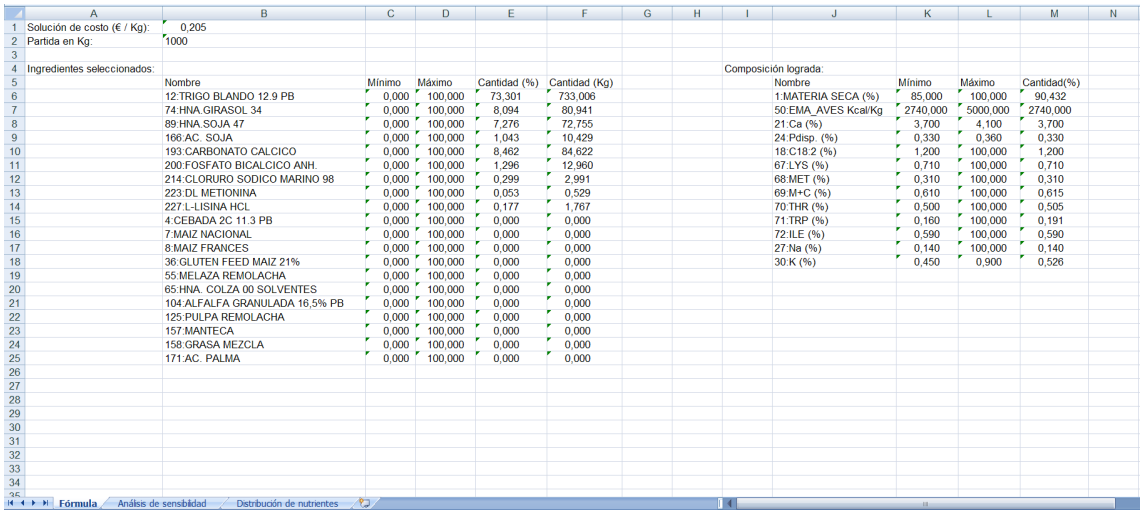
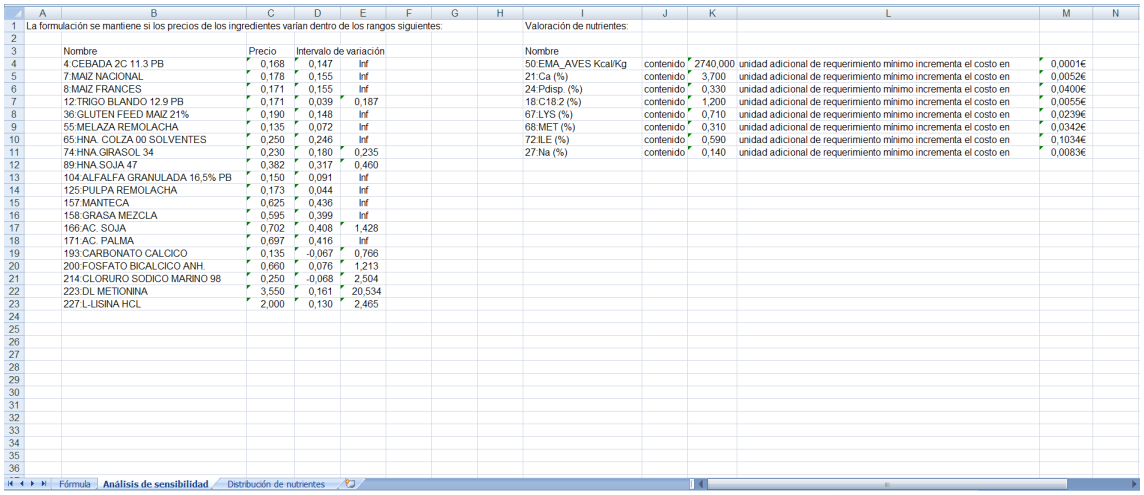


Figura B.11: Abrir Excel creado.



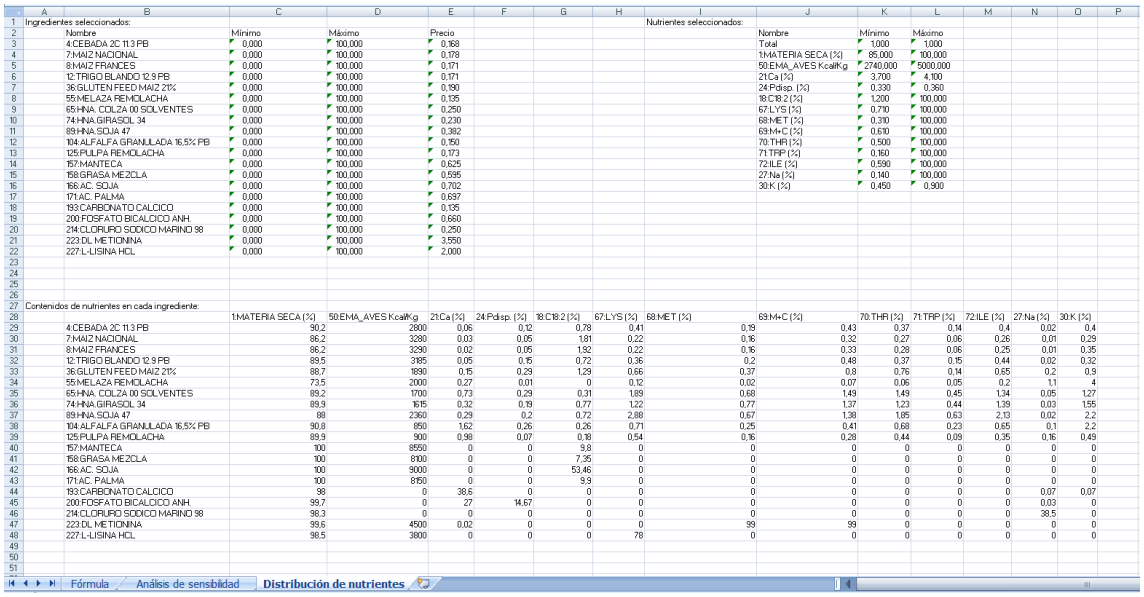
	A	B	C	D	E	F	I	J	K	L	M
1	Solución de costo (€ / Kg):	0,205									
2	Partida en Kg:	1000									
3											
4	Ingredientes seleccionados:						Composición lograda:				
5		Nombre	Mínimo	Máximo	Cantidad (%)	Cantidad (Kg)		Nombre	Mínimo	Máximo	Cantidad(%)
6		12:TRIGO BLANDO 12.9 PB	0,000	100,000	73,301	733,006		1:MATERIA SECA (%)	85,000	100,000	90,432
7		74:HNA.GIRASOL 34	0,000	100,000	8,094	80,941		50:EMA_AVES Kcal/Kg	2740,000	5000,000	2740,000
8		89:HNA.SOJA 47	0,000	100,000	7,276	72,755		21:Ca (%)	3,700	4,100	3,700
9		166:AC. SOJA	0,000	100,000	1,043	10,429		24:Pdisp. (%)	0,330	0,360	0,330
10		193:CARBONATO CALCICO	0,000	100,000	8,462	84,622		18:C18:2 (%)	1,200	100,000	1,200
11		200:FOSFATO BICALCICO ANH.	0,000	100,000	1,296	12,960		67:LYS (%)	0,710	100,000	0,710
12		214:CLORURO SODICO MARINO 98	0,000	100,000	0,299	2,991		68:MET (%)	0,310	100,000	0,310
13		223:DL METIONINA	0,000	100,000	0,053	0,529		69:M+C (%)	0,610	100,000	0,615
14		227:L-LISINA HCL	0,000	100,000	0,177	1,767		70:THR (%)	0,500	100,000	0,505
15		4:CEBADA 2C 11.3 PB	0,000	100,000	0,000	0,000		71:TRP (%)	0,160	100,000	0,191
16		7:MAIZ NACIONAL	0,000	100,000	0,000	0,000		72:ILE (%)	0,590	100,000	0,590
17		8:MAIZ FRANCES	0,000	100,000	0,000	0,000		27:Na (%)	0,140	100,000	0,140
18		36:GLUTEN FEED MAIZ 21%	0,000	100,000	0,000	0,000		30:K (%)	0,450	0,900	0,526
19		55:MELAZA REMOLACHA	0,000	100,000	0,000	0,000					
20		65:HNA. COLZA 00 SOLVENTES	0,000	100,000	0,000	0,000					
21		104:ALFALFA GRANULADA 16,5% PB	0,000	100,000	0,000	0,000					
22		125:PULPA REMOLACHA	0,000	100,000	0,000	0,000					
23		157:MANTECA	0,000	100,000	0,000	0,000					
24		158:GRASA MEZCLA	0,000	100,000	0,000	0,000					
25		171:AC. PALMA	0,000	100,000	0,000	0,000					

Figura B.12: Hoja Fórmula del informe en Excel.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	La formulación se mantiene si los precios de los ingredientes varían dentro de los rangos siguientes:								Valoración de nutrientes:					
2														
3		Nombre	Precio	Intervalo de variación					Nombre					
4		4:CEBADA 2C 11.3 PB	0,168	0,147	Inf				50:EMA_AVES Kcal/Kg	contenido	2740,000	unidad adicional de requerimiento mínimo incrementa el costo en	0,0001€	
5		7:MAIZ NACIONAL	0,178	0,155	Inf				21:Ca (%)	contenido	3,700	unidad adicional de requerimiento mínimo incrementa el costo en	0,0052€	
6		8:MAIZ FRANCES	0,171	0,155	Inf				24:Pdisp. (%)	contenido	0,330	unidad adicional de requerimiento mínimo incrementa el costo en	0,0400€	
7		12:TRIGO BLANDO 12.9 PB	0,171	0,039	0,187				18:C18:2 (%)	contenido	1,200	unidad adicional de requerimiento mínimo incrementa el costo en	0,0055€	
8		36:GLUTEN FEED MAIZ 21%	0,190	0,148	Inf				67:LYS (%)	contenido	0,710	unidad adicional de requerimiento mínimo incrementa el costo en	0,0239€	
9		55:MELAZA REMOLACHA	0,135	0,072	Inf				68:MET (%)	contenido	0,310	unidad adicional de requerimiento mínimo incrementa el costo en	0,0342€	
10		65:HNA. COLZA 00 SOLVENTES	0,250	0,246	Inf				72:ILE (%)	contenido	0,590	unidad adicional de requerimiento mínimo incrementa el costo en	0,1034€	
11		74:HNA.GIRASOL 34	0,230	0,180	0,235				27:Na (%)	contenido	0,140	unidad adicional de requerimiento mínimo incrementa el costo en	0,0083€	
12		89:HNA.SOJA 47	0,382	0,317	0,460									
13		104:ALFALFA GRANULADA 16,5% PB	0,150	0,091	Inf									
14		125:PULPA REMOLACHA	0,173	0,044	Inf									
15		157:MANTECA	0,625	0,436	Inf									
16		158:GRASA MEZCLA	0,595	0,399	Inf									
17		166:AC. SOJA	0,702	0,408	1,428									
18		171:AC. PALMA	0,697	0,416	Inf									
19		193:CARBONATO CALCICO	0,135	-0,067	0,766									
20		200:FOSFATO BICALCICO ANH.	0,660	0,076	1,213									
21		214:CLORURO SODICO MARINO 98	0,250	-0,068	2,504									
22		223:DL METIONINA	3,550	0,161	20,534									
23		227:L-LISINA HCL	2,000	0,130	2,465									

Figura B.13: Hoja Análisis de sensibilidad del informe en Excel.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Ingredientes seleccionados:								Nutrientes seleccionados:							
2		Nombre	Mínimo	Máximo	Precio					Nombre	Mínimo	Máximo				
3		4:CEBADA 2C 11.3 PB	0,000	100,000	0,168					Total	1,000	1,000				
4		7:MAIZ NACIONAL	0,000	100,000	0,178					1:MATERIA SECA (%)	85,000	100,000				
5		8:MAIZ FRANCES	0,000	100,000	0,171					50:EMA_AVES Kcal/Kg	2740,000	5000,000				
6		12:TRIGO BLANDO 12.9 PB	0,000	100,000	0,171					21:Ca (%)	3,700	4,100				
7		36:GLUTEN FEED MAIZ 21%	0,000	100,000	0,190					24:Pdisp. (%)	0,330	0,360				
8		55:MELAZA REMOLACHA	0,000	100,000	0,135					18:C18:2 (%)	1,200	100,000				
9		65:HNA. COLZA 00 SOLVENTES	0,000	100,000	0,250					67:LYS (%)	0,710	100,000				
10		74:HNA.GIRASOL 34	0,000	100,000	0,230					68:MET (%)	0,310	100,000				
11		89:HNA.SOJA 47	0,000	100,000	0,382					69:M+C (%)	0,610	100,000				
12		104:ALFALFA GRANULADA 16,5% PB	0,000	100,000	0,150					70:THR (%)	0,500	100,000				
13		125:PULPA REMOLACHA	0,000	100,000	0,173					71:TRP (%)	0,160	100,000				
14		157:MANTECA	0,000	100,000	0,625					72:ILE (%)	0,590	100,000				
15		158:GRASA MEZCLA	0,000	100,000	0,595					27:Na (%)	0,140	100,000				
16		166:AC. SOJA	0,000	100,000	0,702					30:K (%)	0,450	0,900				
17		171:AC. PALMA	0,000	100,000	0,697											
18		193:CARBONATO CALCICO	0,000	100,000	0,135											
19		200:FOSFATO BICALCICO ANH.	0,000	100,000	0,660											
20		214:CLORURO SODICO MARINO 98	0,000	100,000	0,250											
21		223:DL METIONINA	0,000	100,000	3,550											
22		227:L-LISINA HCL	0,000	100,000	2,000											
23																
24																
25																
26																
27	Contenidos de nutrientes en cada ingrediente:															
28			1:MATERIA SECA (%)	50:EMA_AVES Kcal/Kg	21:Ca (%)	24:Pdisp. (%)	18:C18:2 (%)	67:LYS (%)	68:MET (%)	69:M+C (%)	70:THR (%)	71:TRP (%)	72:ILE (%)	27:Na (%)	30:K (%)		
29		4:CEBADA 2C 11.3 PB	90,2	2800	0,06	0,12	0,78	0,41	0,19	0,43	0,37	0,14	0,4	0,02	0,4		
30		7:MAIZ NACIONAL	86,2	3280	0,03	0,05	1,81	0,22	0,16	0,32	0,27	0,06	0,26	0,01	0,29		
31		8:MAIZ FRANCES	86,2	3290	0,02	0,05	1,92	0,22	0,16	0,33	0,28	0,06	0,25	0,01	0,35		
32		12:TRIGO BLANDO 12.9 PB	89,5	3185	0,05	0,15	0,72	0,36	0,2	0,48	0,37	0,15	0,44	0,02	0,32		
33		36:GLUTEN FEED MAIZ 21%	88,7	1880	0,15	0,29	1,29	0,66	0,37	0,8	0,76	0,14	0,65	0,2	0,9		
34		55:MELAZA REMOLACHA	73,5	2000	0,27	0,01	0	0,12	0,02	0,07	0,06	0,05	0,2	1,1	4		
35		65:HNA. COLZA 00 SOLVENTES	89,2	1700	0,73	0,29	0,31	1,89	0,68	1,49	1,49	0,45	1,34	0,05	1,27		
36		74:HNA.GIRASOL 34	89,9	1615	0,32	0,19	0,77	1,22	0,77	1,37	1,23	0,44	1,39	0,03	1,55		
37		89:HNA.SOJA 47	88	2360	0,29	0,2	0,72	2,88	0,67	1,38	1,85	0,63	2,13	0,02	2,2		
38		104:ALFALFA GRANULADA 16,5% PB	90,8	850	1,62	0,26	0,26	0,71	0,25	0,41	0,68	0,23	0,65	0,1	2,2		
39		125:PULPA REMOLACHA	89,9	900	0,98	0,07	0,18	0,54	0,16	0,28	0,44	0,09	0,35	0,16	0,49		
40		157:MANTECA	100	8550	0	0	9,8	0	0	0	0	0	0	0	0		
41		158:GRASA MEZCLA	100	8100	0	0	7,35	0	0	0	0	0	0	0	0		
42		166:AC. SOJA	100	9000	0	0	53,46	0	0	0	0	0	0	0	0		
43		171:AC. PALMA	100	8150	0	0	9,9	0	0	0	0	0	0	0	0		
44		193:CARBONATO CALCICO	98	0	38,6	0	0	0	0	0	0	0	0	0,07	0,07		
45		200:FOSFATO BICALCICO ANH.	99,7	0	27	14,67	0	0	0	0	0	0	0	0,03	0		
46		214:CLORURO SODICO MARINO 98	98,3	0	0	0	0	0	0	0	0	0	0	38,5	0		
47		223:DL METIONINA	99,6	4500	0,02	0	0	0	99	99	0	0	0	0	0		
48		227:L-LISINA HCL	98,5	3800	0	0	0	78	0	0	0	0	0	0	0		
49																	
50																	
51																	

Figura B.14: Hoja Distribución de nutrientes del informe en Excel.

Anexo C

Código del Programa

C.1. Ventana principal

```
package programadietas;

/**
*
* @author aaguilar, pmateo
*/

import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Image;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.swing.JTable;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Box;
import javax.swing.DefaultCellEditor;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JFileChooser;
import javax.swing.JMenu;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import javax.swing.WindowConstants;
import javax.swing.event.TableModelEvent;
import javax.swing.event.TableModelListener;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import static programadietas.datosGlobales.A;
import static programadietas.datosGlobales.c;
```

```
import static programadietas.datosGlobales.l;
import static programadietas.datosGlobales.u;
import static programadietas.datosGlobales.bL;
import static programadietas.datosGlobales.bU;
import static programadietas.datosGlobales.nombreFichero;



public class Ventana extends javax.swing.JFrame {
   /**
   * Creates new form Ventana
   */


   public Ventana() {

      //Se crean todos los componentes de la interfaz gráfica
      initComponents();

      //Se conecta con la base de datos
      baseDeDatos tabla = new baseDeDatos();
      String ruta = null;
      tabla.conectar(ruta);

      //Se crean vectores y matrices estáticas para almacenar los datos
      datosGlobales.listaNutrientes=tabla.obtenListaNutrientes();
      datosGlobales.listaIngredientes = tabla.obtenListaIngredientes();
      datosGlobales.listaUnidades=tabla.obtenListaUnidadesNutrientes();
      datosGlobales.tablaAlimentosSS= tabla.obtenMatrizAString();
      datosGlobales.tablaAlimentos=tabla.obtenMatrizA();

      //Se incluyen los datos de la BD en el jTable3 (pestaña base de datos)
      jTable3.setModel(new DefaultTableModel(datosGlobales.tablaAlimentosSS,
      datosGlobales.listaNutrientes));
      tabla.cierraConexion();

      //Se vuelcan la lista de ingredientes y nutrientes en sus desplegables
      //correspondientes
      jComboBox1.setModel(
      new DefaultComboBoxModel(datosGlobales.listaIngredientes));
      jComboBox3.setModel(
      new DefaultComboBoxModel(datosGlobales.listaNutrientes));
      jComboBox3.removeItemAt(0);
      jComboBox3.removeItemAt(jComboBox3.getItemCount()-1);

      //Al arranque del programa estos menús están desactivados
      jMenuItem2.setEnabled(false);
      jMenuItem3.setEnabled(false);
      jMenuItem4.setEnabled(false);
      jMenuItem5.setEnabled(false);
      jMenuItem9.setEnabled(false);
      jMenuItem7.setEnabled(false);
      jMenuItem10.setEnabled(false);

      //Incluimos un icono al programa, así no aparece el predeterminado de Java
      Image icon = Toolkit.getDefaultToolkit().getImage(
      getClass().getResource("pienso.png"));
      setIconImage(icon);
```

```
        //Creación de la ventana Acerca de
        JMenu jMenu6 = new javax.swing.JMenu();
        jMenu6.setText("Acerca de");
        jMenu6.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jMenu6MouseClicked(evt);
            }
        });

        jMenu6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenu6ActionPerformed(evt);
            }

            private void jMenu6ActionPerformed(ActionEvent evt) {
                throw new UnsupportedOperationException("Not supported yet."); //To change
                    body of generated methods, choose Tools | Templates.
            }
        });
        MenuPrincipal.add(jMenu6);

    }
    /**
    * This method is called from within the constructor to initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is always
    * regenerated by the Form Editor.
    */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel1 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jTable1.setRowSelectionAllowed(false);
        jTable1.setColumnSelectionAllowed(false);
        jScrollPane2 = new javax.swing.JScrollPane();
        jTable2 = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jComboBox1 = new javax.swing.JComboBox();
        jComboBox2 = new javax.swing.JComboBox();
        jComboBox3 = new javax.swing.JComboBox();
        jComboBox4 = new javax.swing.JComboBox();
        jPanel2 = new javax.swing.JPanel();
        jScrollPane4 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jPanel3 = new javax.swing.JPanel();
        jScrollPane3 = new javax.swing.JScrollPane();
        jTable3 = new javax.swing.JTable(){
            public boolean isCellEditable(int rowIndex, int colIndex) {
                return false; //Disallow the editing of any cell
```

```
   }
};
MenuPrincipal = new javax.swing.JMenuBar();
jMenu1 = new javax.swing.JMenu();
jMenuItem10 = new javax.swing.JMenuItem();
jSeparator6 = new javax.swing.JPopupMenu.Separator();
jMenuItem1 = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JPopupMenu.Separator();
jMenuItem2 = new javax.swing.JMenuItem();
jSeparator2 = new javax.swing.JPopupMenu.Separator();
jMenuItem3 = new javax.swing.JMenuItem();
jSeparator4 = new javax.swing.JPopupMenu.Separator();
jMenuItem8 = new javax.swing.JMenuItem();
jMenu2 = new javax.swing.JMenu();
jMenuItem9 = new javax.swing.JMenuItem();
jSeparator3 = new javax.swing.JPopupMenu.Separator();
jMenuItem4 = new javax.swing.JMenuItem();
jSeparator5 = new javax.swing.JPopupMenu.Separator();
jMenuItem5 = new javax.swing.JMenuItem();
jMenu5 = new javax.swing.JMenu();
jMenuItem7 = new javax.swing.JMenuItem();
jMenu3 = new javax.swing.JMenu();
jMenuItem6 = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("CompFeed v1.0");

jPanel1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

jScrollPane1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

jTable1.setModel(new javax.swing.table.DefaultTableModel(
new Object [][] {

},
new String [] {
   "Nombre", "Mín.", "Máx.", "Precio"
}
) {
   Class[] types = new Class [] {
      java.lang.String.class, java.lang.String.class, java.lang.String.class,
          java.lang.String.class
   };
   boolean[] canEdit = new boolean [] {
      false, true, true, true
   };

   public Class getColumnClass(int columnIndex) {
      return types [columnIndex];
   }

   public boolean isCellEditable(int rowIndex, int columnIndex) {
      return canEdit [columnIndex];
   }
});
jTable1.setSurrendersFocusOnKeystroke(true);
jTable1.getTableHeader().setReorderingAllowed(false);
jTable1.addFocusListener(new java.awt.event.FocusAdapter() {
```

```
        public void focusLost(java.awt.event.FocusEvent evt) {
            jTable1FocusLost(evt);
        }
    });
    jScrollPane1.setViewportView(jTable1);
    jTable1.getModel().addTableModelListener(new TableModelListener() {

        public void tableChanged(TableModelEvent evento) {
            System.out.println("Algun dato ha cambiado, algo habrá que hacer");
            jMenuItem4.setEnabled(false);
            jMenuItem5.setEnabled(false);
            jMenuItem7.setEnabled(false);
        }
    });

    jTable2.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {
        "Nombre", "Mín.", "Máx."
    }
    ) {
        boolean[] canEdit = new boolean [] {
            false, true, true
        };

        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });
    jTable2.setRowSelectionAllowed(false);
    jTable2.getTableHeader().setReorderingAllowed(false);
    jTable2.addFocusListener(new java.awt.event.FocusAdapter() {
        public void focusLost(java.awt.event.FocusEvent evt) {
            jTable2FocusLost(evt);
        }
    });
    jScrollPane2.setViewportView(jTable2);
    jTable2.getModel().addTableModelListener(new TableModelListener() {

        public void tableChanged(TableModelEvent evento) {
            System.out.println("Algun dato ha cambiado, algo habrá que hacer");
            jMenuItem4.setEnabled(false);
            jMenuItem5.setEnabled(false);
            jMenuItem7.setEnabled(false);
        }
    });

    jLabel1.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
    jLabel1.setText("Ingredientes");

    jLabel2.setFont(new java.awt.Font("Tahoma", 1, 13)); // NOI18N
    jLabel2.setText("Nutrientes");

    jButton1.setText("Añadir");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            jButton1ActionPerformed(evt);
        }
    });

    jButton2.setText("Borrar");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    jButton3.setText("Añadir");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });

    jButton4.setText("Borrar");
    jButton4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton4ActionPerformed(evt);
        }
    });

    jComboBox1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });

    jComboBox2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox2ActionPerformed(evt);
        }
    });

    jComboBox3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox3ActionPerformed(evt);
        }
    });

    jComboBox4.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox4ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
    .addGap(21, 21, 21)
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
    .addGap(50, 50, 50)
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
    false)
.addComponent(jComboBox2, 0, 278, Short.MAX_VALUE)
.addComponent(jComboBox1, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
.addGap(27, 27, 27)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jButton1)
.addComponent(jButton2, javax.swing.GroupLayout.Alignment.TRAILING))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
    false)
.addComponent(jComboBox3, 0, 238, Short.MAX_VALUE)
.addComponent(jComboBox4, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
.addGap(26, 26, 26)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
.addComponent(jButton4)
.addComponent(jButton3))
.addGap(60, 60, 60))
.addGroup(jPanel1Layout.createSequentialGroup()
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 160,
    javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(0, 362, Short.MAX_VALUE))
.addGroup(jPanel1Layout.createSequentialGroup()
.addGap(10, 10, 10)
.addComponent(jScrollPane1)))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 120,
    javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 414,
    Short.MAX_VALUE))
.addGap(23, 23, 23))))
);
jPanel1Layout.setVerticalGroup(
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(jPanel1Layout.createSequentialGroup()
.addGap(48, 48, 48)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jLabel1)
.addComponent(jLabel2))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 169,
    Short.MAX_VALUE)
.addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
    Short.MAX_VALUE))
.addGap(44, 44, 44)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton1)
.addComponent(jButton3)
.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(18, 18, 18)
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jButton2)
.addComponent(jButton4)
.addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
.addGap(53, 53, 53))
);

jTabbedPane1.addTab("Ingredientes y Nutrientes", jPanel1);

jPanel2.setLayout(new java.awt.BorderLayout());

jTextArea1.setColumns(20);
jTextArea1.setFont(new java.awt.Font("Courier New", 0, 13)); // NOI18N
jTextArea1.setRows(5);
jTextArea1.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mousePressed(java.awt.event.MouseEvent evt) {
      jTextArea1MousePressed(evt);
   }
});
jScrollPane4.setViewportView(jTextArea1);

jPanel2.add(jScrollPane4, java.awt.BorderLayout.CENTER);

jTabbedPane1.addTab("Fórmulas", null, jPanel2, "");

jPanel3.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
jPanel3.addContainerListener(new java.awt.event.ContainerAdapter() {
   public void componentAdded(java.awt.event.ContainerEvent evt) {
      jPanel3ComponentAdded(evt);
   }
});
jPanel3.addAncestorListener(new javax.swing.event.AncestorListener() {
   public void ancestorMoved(javax.swing.event.AncestorEvent evt) {
   }
   public void ancestorAdded(javax.swing.event.AncestorEvent evt) {
      jPanel3AncestorAdded(evt);
   }
   public void ancestorRemoved(javax.swing.event.AncestorEvent evt) {
   }
});

jScrollPane3.setToolTipText("");
jScrollPane3.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

jTable3.setModel(new javax.swing.table.DefaultTableModel(
new Object [][] {
   {},
   {},
   {},
```

```
        {}
    },
    new String [] {

    }
    ));
    jTable3.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_OFF);
    jTable3.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mousePressed(java.awt.event.MouseEvent evt) {
            jTable3MousePressed(evt);
        }
    });
    jTable3.getTableHeader().addMouseListener(new java.awt.event.MouseAdapter() {
        public void mousePressed(java.awt.event.MouseEvent evt) {

            if (evt.getClickCount() == 2){
                System.out.println("Hemos pulsado cabecera twice");
                int col = jTable3.columnAtPoint(evt.getPoint());
                if(col>=1){
                    String dato = jTable3.getColumnName(col);
                    System.out.println("Column index selected " + col + " " + dato);
                    for(int i=0;i<jComboBox4.getItemCount();i++) {
                        if(jComboBox4.getItemAt(i).equals(dato))return;
                    }
                    System.out.println("Nutriente aun no esta introducido, lo
                        introducimos");
                    DefaultTableModel modelo = (DefaultTableModel) jTable2.getModel();

                    jMenuItem2.setEnabled(true);
                    jMenuItem3.setEnabled(true);
                    jMenuItem9.setEnabled(true);
                    Object [] fila=new String[3];
                    fila[0]= dato;
                    fila[1]="0";
                    fila[2]="100";
                    modelo.addRow(fila);
                    jComboBox3.removeItem(dato);
                    jComboBox4.addItem(dato);
                    jTable2.setModel(modelo);
                    System.out.println("Número de nutrientes "+jTable2.getRowCount());
                    for (int i=0; i< jTable2.getRowCount(); i++){
                        String datos = jTable2.getValueAt(i,0)+"";
                        String min = jTable2.getValueAt(i,1)+"";
                        String max = jTable2.getValueAt(i,2)+"";
                        Nutriente nutriente = new Nutriente("",datos,min,max);
                        String nutri = nutriente.Nombre();
                        String minim = nutriente.Min();
                        String maxim = nutriente.Max();
                        System.out.println(nutri+"::"+minim+"::"+maxim);
                    }
                    System.out.println("\n");
                }
            }

        }
    });
    jScrollPane3.setViewportView(jTable3);
```

```
javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane3, javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.DEFAULT_SIZE, 986, Short.MAX_VALUE)
);
jPanel3Layout.setVerticalGroup(
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane3, javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.DEFAULT_SIZE, 405, Short.MAX_VALUE)
);

jTabbedPane1.addTab("Tabla de ingredientes", jPanel3);

MenuPrincipal.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
MenuPrincipal.setInheritsPopupMenu(true);
MenuPrincipal.addComponentListener(new java.awt.event.ComponentAdapter() {
   public void componentResized(java.awt.event.ComponentEvent evt) {
      MenuPrincipalComponentResized(evt);
   }
});

jMenu1.setText("Fichero");
jMenu1.setName(""); // NOI18N

jMenuItem10.setText("Nueva fórmula");
jMenuItem10.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem10ActionPerformed(evt);
   }
});
jMenu1.add(jMenuItem10);
jMenu1.add(jSeparator6);

jMenuItem1.setText("Cargar fórmula");
jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem1ActionPerformed(evt);
   }
});
jMenu1.add(jMenuItem1);
jMenu1.add(jSeparator1);

jMenuItem2.setText("Guardar fórmula");
jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem2ActionPerformed(evt);
   }
});
jMenu1.add(jMenuItem2);
jMenu1.add(jSeparator2);

jMenuItem3.setText("Guardar fórmula como ...");
jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem3ActionPerformed(evt);
   }
```

```
});
jMenu1.add(jMenuItem3);
jMenu1.add(jSeparator4);

jMenuItem8.setText("Salir");
jMenuItem8.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem8ActionPerformed(evt);
   }
});
jMenu1.add(jMenuItem8);

MenuPrincipal.add(jMenu1);
jMenu1.getAccessibleContext().setAccessibleDescription("");

jMenu2.setText("Formulación");

jMenuItem9.setText("Generar problema");
jMenuItem9.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem9ActionPerformed(evt);
   }
});
jMenu2.add(jMenuItem9);
jMenu2.add(jSeparator3);

jMenuItem4.setText("Optimizar");
jMenuItem4.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem4ActionPerformed(evt);
   }
});
jMenu2.add(jMenuItem4);
jMenu2.add(jSeparator5);

jMenuItem5.setText("Análisis de sensibilidad");
jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem5ActionPerformed(evt);
   }
});
jMenu2.add(jMenuItem5);

MenuPrincipal.add(jMenu2);

jMenu5.setText("Informes");

jMenuItem7.setText("Generar informe");
jMenuItem7.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jMenuItem7ActionPerformed(evt);
   }
});
jMenu5.add(jMenuItem7);

MenuPrincipal.add(jMenu5);

jMenu3.setText("Tabla de ingredientes");
```

```
        jMenuItem6.setText("Actualizar precios");
        jMenuItem6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem6ActionPerformed(evt);
            }
        });
        jMenu3.add(jMenuItem6);

        MenuPrincipal.add(jMenu3);

        MenuPrincipal.add(Box.createHorizontalGlue());

        setJMenuBar(MenuPrincipal);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addGap(23, 23, 23)
        .addComponent(jTabbedPane1)
        .addGap(12, 12, 12))
        );
        layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addGap(22, 22, 22)
        .addComponent(jTabbedPane1)
        .addGap(24, 24, 24))
        );
        pack();
    }// </editor-fold>


    private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem7.setEnabled(true);
        JTable tabla1 = this.jTable1;
        JTable tabla2 = this.jTable2;
        JTextArea texto = this.jTextArea1;
        PPLs opt = new PPLs(tabla1,tabla2,texto);
        this.jTextArea1.append("\n");
        opt.resuelveProblema();
        opt.análisisDeSensibilidad();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        String dato = this.jComboBox1.getSelectedItem().toString();

        DefaultTableModel modelo = (DefaultTableModel) jTable1.getModel();
        jMenuItem2.setEnabled(true);
        jMenuItem3.setEnabled(true);
        jMenuItem9.setEnabled(true);
        jMenuItem10.setEnabled(true);

        String [] fila = new String[4];
        fila[0]= dato;
```

```
        fila[1]="0";
        fila[2]="100";
        fila[3]=datosGlobales.tablaAlimentosSS[-1+Integer.parseInt(dato.split(":")[0])]
                [datosGlobales.tablaAlimentos[0].length-1];

        modelo.addRow(fila);

        this.jComboBox1.removeItem(dato);

        jComboBox2.addItem(dato);

        jTable1.setModel(modelo);


        System.out.println("Número de ingredientes "+jTable1.getRowCount());

        FontMetrics fm =jTable1.getFontMetrics(jTable1.getFont());
        int imax=0;
        for (int i=0; i< jTable1.getRowCount(); i++){
           if( fm.stringWidth(""+jTable1.getValueAt(i,0))>imax)
           imax=fm.stringWidth(""+jTable1.getValueAt(i,0));
        }
        jTable1.getColumnModel().getColumn(0).setPreferredWidth(imax);
        System.out.println(imax);

        for (int i=0; i< jTable1.getRowCount(); i++){
           String datos = jTable1.getValueAt(i,0)+"";
           String min = jTable1.getValueAt(i,1)+"";
           String max = jTable1.getValueAt(i,2)+"";
           String precio = jTable1.getValueAt(i,3)+"";

           Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
           String ingred = ingrediente.Nombre();
           String minim = ingrediente.Min();
           String maxim = ingrediente.Max();
           String prec = ingrediente.Precio();
           System.out.println(ingred+"::"+minim+"::"+maxim+"::"+prec);
        }

        System.out.println("\n");

    }


    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

        try{

           String b = this.jComboBox2.getSelectedItem().toString();
           int a = this.jComboBox2.getSelectedIndex();

           int confirmar=JOptionPane.showConfirmDialog(null,
           "¿Está seguro de que desea Eliminar el registro? ");


           if(JOptionPane.OK_OPTION==confirmar) {
```

```
            model.removeRow(a);
            jComboBox2.removeItemAt(a);
            jComboBox1.addItem(b);
            JOptionPane.showMessageDialog(null,"Registro Eliminado" );


            System.out.println("Número de ingredientes "+jTable1.getRowCount());

            for (int i=0; i< jTable1.getRowCount(); i++){
               String datos = jTable1.getValueAt(i,0)+"";
               String min = jTable1.getValueAt(i,1)+"";
               String max = jTable1.getValueAt(i,2)+"";
               String precio = jTable1.getValueAt(i,3)+"";

               Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
               String ingred = ingrediente.Nombre();
               String minim = ingrediente.Min();
               String maxim = ingrediente.Max();
               String prec = ingrediente.Precio();
               System.out.println(ingred+"::"+minim+"::"+maxim+"::"+prec);
            }

            System.out.println("\n");

         }

         if (jTable1.getRowCount()==0 && jTable2.getRowCount()==0){
            jMenuItem2.setEnabled(false);
            jMenuItem3.setEnabled(false);
            jMenuItem9.setEnabled(false);
         }

      } catch (NullPointerException e) {

      JOptionPane.showMessageDialog(null,
      "Debe añadir primero un ingrediente a la tabla" );
   }
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

   DefaultTableModel model = (DefaultTableModel) jTable2.getModel();

   try{

      String b = this.jComboBox4.getSelectedItem().toString();

      int a = this.jComboBox4.getSelectedIndex();

      int confirmar=JOptionPane.showConfirmDialog(null,
      "¿Está seguro de que desea Eliminar el registro? ");

      if(JOptionPane.OK_OPTION==confirmar) {

         model.removeRow(a);
         jComboBox4.removeItemAt(a);
         JOptionPane.showMessageDialog(null,"Registro Eliminado" );
```

```
        this.jComboBox3.addItem(b);

        System.out.println("Número de nutrientes "+jTable2.getRowCount());


        for (int i=0; i< jTable2.getRowCount(); i++){

           String datos = jTable2.getValueAt(i,0)+"";
           String min = jTable2.getValueAt(i,1)+"";
           String max = jTable2.getValueAt(i,2)+"";

           Nutriente nutriente = new Nutriente("",datos,min,max);
           String nutri = nutriente.Nombre();
           String minim = nutriente.Min();
           String maxim = nutriente.Max();
           System.out.println(nutri+"::"+minim+"::"+maxim);
        }
        System.out.println("\n");
     }

     if (jTable1.getRowCount()==0 && jTable2.getRowCount()==0){
        jMenuItem2.setEnabled(false);
        jMenuItem3.setEnabled(false);
        jMenuItem9.setEnabled(false);
     }

   } catch(NullPointerException e) {

   JOptionPane.showMessageDialog(null,
   "Debe añadir primero un ingrediente a la tabla" );
}
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
   String dato = this.jComboBox3.getSelectedItem().toString();

   DefaultTableModel modelo = (DefaultTableModel) jTable2.getModel();

   jMenuItem2.setEnabled(true);
   jMenuItem3.setEnabled(true);
   jMenuItem9.setEnabled(true);
   jMenuItem10.setEnabled(true);

   Object [] fila=new String[3];

   fila[0]= dato;
   fila[1]="0";
   fila[2]="100";

   modelo.addRow(fila);
   this.jComboBox3.removeItem(dato);
   jComboBox4.addItem(dato);
   jTable2.setModel(modelo);

   FontMetrics fm =jTable2.getFontMetrics(jTable2.getFont());
   int imax=0;
   for (int i=0; i< jTable2.getRowCount(); i++){
      if( fm.stringWidth(""+jTable2.getValueAt(i,0))>imax)
```

```
      imax=fm.stringWidth(""+jTable2.getValueAt(i,0));
   }
   jTable2.getColumnModel().getColumn(0).setPreferredWidth(imax);



   System.out.println("Número de nutrientes "+jTable2.getRowCount());



   for (int i=0; i< jTable2.getRowCount(); i++){

      String datos = jTable2.getValueAt(i,0)+"";
      String min = jTable2.getValueAt(i,1)+"";
      String max = jTable2.getValueAt(i,2)+"";

      Nutriente nutriente = new Nutriente("",datos,min,max);
      String nutri = nutriente.Nombre();
      String minim = nutriente.Min();
      String maxim = nutriente.Max();
      System.out.println(nutri+"::"+minim+"::"+maxim);
   }

   System.out.println("\n");

}


private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {

   String ingr="";
   String nutr="";

   Properties prop = new Properties();
   prop.setProperty("NumIngredientes", jTable1.getRowCount()+"");
   prop.setProperty("NumNutrientes", jTable2.getRowCount()+"");

   for (int i=0; i< jTable1.getRowCount(); i++){
      String datos = jTable1.getValueAt(i,0)+"";
      String min = jTable1.getValueAt(i,1)+"";
      String max = jTable1.getValueAt(i,2)+"";
      String precio = jTable1.getValueAt(i,3)+"";

      Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
      String ingred = ingrediente.Nombre();
      String minim = ingrediente.Min();
      String maxim = ingrediente.Max();
      String prec = ingrediente.Precio();

      ingr=ingred+"//"+minim+"//"+maxim+"//"+prec;
      prop.setProperty("Ingrediente"+(i+1),ingr);
   }


   for (int i=0; i< jTable2.getRowCount(); i++){

      String datos = jTable2.getValueAt(i,0)+"";
      String min = jTable2.getValueAt(i,1)+"";
```

```
      String max = jTable2.getValueAt(i,2)+"";

      Nutriente nutriente = new Nutriente("",datos,min,max);
      String nutri = nutriente.Nombre();
      String minim = nutriente.Min();
      String maxim = nutriente.Max();

      nutr=nutri+"//"+minim+"//"+maxim;
      prop.setProperty("Nutriente"+(i+1),nutr);
   }



   if (datosGlobales.cargarguardar==true){

      prop.setProperty("NombreDeDieta", datosGlobales.nombreFichero);
      //System.out.println(prop);
      String nombre = prop.getProperty("NombreDeDieta");
      //System.out.println(nombre);

      File fichero = new File(nombre);

      try {
         OutputStream out;
         out = new FileOutputStream( fichero );
         prop.store(out, ingr);
      } catch (IOException ex) {
      Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
   }

   System.out.println("El fichero " + fichero + " existe");

}

else{

   JFileChooser chooser = new JFileChooser(new File(System.getProperty("user.home")));
   FileNameExtensionFilter filter = new FileNameExtensionFilter("Ficheros de dieta",
       "die");
   chooser.setFileFilter(filter);
   int setVal = chooser.showSaveDialog(null);
   if(setVal == JFileChooser.APPROVE_OPTION) {
      try{
         System.out.println("Has elegido guardar este archivo: " +
             chooser.getSelectedFile().getAbsolutePath()+".die");


         prop.setProperty("NombreDeDieta",chooser.getSelectedFile().getAbsolutePath()+".die");
         this.setTitle("CompFeed v1.0
             "+"("+chooser.getSelectedFile().getAbsolutePath()+".die"+")");

         try {
            File f = new File(chooser.getSelectedFile().getAbsolutePath()+".die");
            OutputStream out;
            out = new FileOutputStream( f );
            prop.store(out, ingr);
         } catch (IOException ex) {
```

```
            Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
        }

        datosGlobales.nombreFichero = chooser.getSelectedFile().getAbsolutePath();
        System.out.println(datosGlobales.nombreFichero);

    }catch(Exception e){
    System.out.println(e.getMessage());
}
}
}
}

private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

    JFileChooser chooser = new JFileChooser(new File(System.getProperty("user.home")));
    FileNameExtensionFilter filter = new FileNameExtensionFilter("Ficheros de dieta",
        "die");
    chooser.setFileFilter(filter);
    int returnVal = chooser.showOpenDialog(null);
    if(returnVal == JFileChooser.APPROVE_OPTION) {
        System.out.println("Has elegido abrir el archivo: " +
            chooser.getSelectedFile().getAbsolutePath());

        this.setTitle(chooser.getSelectedFile().getAbsolutePath());

        jMenuItem2.setEnabled(true);
        jMenuItem3.setEnabled(true);
        jMenuItem9.setEnabled(true);
        jMenuItem10.setEnabled(true);
        jTextArea1.setText("");
        DefaultTableModel modelo1 = (DefaultTableModel) jTable1.getModel();
        DefaultTableModel modelo2 = (DefaultTableModel) jTable2.getModel();

        File f = chooser.getSelectedFile();
        Properties prop = new Properties();
        InputStream is = null;
        FontMetrics fm =jTable1.getFontMetrics(jTable1.getFont());
        int imax=0;


        try{
            //limpiar tablas
            int fila1=jTable1.getRowCount();
            int fila2=jTable2.getRowCount();
            jComboBox1.setModel(new
                DefaultComboBoxModel(datosGlobales.listaIngredientes));
            jComboBox3.setModel(new DefaultComboBoxModel(datosGlobales.listaNutrientes));
            jComboBox3.removeItemAt(0);
            int desp1 = jComboBox2.getItemCount();
            int desp2 = jComboBox4.getItemCount();

            for (int i = 0;fila1>i; i++) {
                modelo1.removeRow(0);
            }
            for (int i = 0;fila2>i; i++) {
                modelo2.removeRow(0);
            }
```

```
            for(int i=0;i<desp1;i++){
               jComboBox2.removeItemAt(0);
            }
            for(int i=0;i<desp2;i++){
               jComboBox4.removeItemAt(0);
            }

            is = new FileInputStream(f);
            prop.load(is);

            int a = Integer.parseInt(prop.getProperty("NumIngredientes"));
            int b = Integer.parseInt(prop.getProperty("NumNutrientes"));

            System.out.println("Numero ingredientes del fichero: "+a);
            System.out.println("Numero nutrientes del fichero: "+b);

            imax=0;
            for(int i = 0; i < a ; i++){

               String linea = prop.getProperty("Ingrediente"+(i+1));
               System.out.println(linea);
               String [] filas = linea.split("//");

               modelo1.addRow(filas);
               String dato = filas[0];
               this.jComboBox1.removeItem(dato);
               jComboBox2.addItem(dato);
               if( fm.stringWidth(dato)>imax)
               imax=fm.stringWidth(dato);

            }
            jTable1.getColumnModel().getColumn(0).setPreferredWidth(imax);

            imax=0;
            fm=jTable2.getFontMetrics(jTable2.getFont());

            for(int i = 0; i < b ; i++){

               String linea = prop.getProperty("Nutriente"+(i+1));
               System.out.println(linea);
               String [] filas = linea.split("//");

               modelo2.addRow(filas);
               String dato = filas[0];
               this.jComboBox3.removeItem(dato);
               jComboBox4.addItem(dato);
               if( fm.stringWidth(dato)>imax)
               imax=fm.stringWidth(dato);
            }
            jTable2.getColumnModel().getColumn(0).setPreferredWidth(imax);
            datosGlobales.nombreFichero = chooser.getSelectedFile().getAbsolutePath();
            System.out.println(datosGlobales.nombreFichero);
            this.setTitle("CompFeed v1.0"+" ("+nombreFichero+")");
            datosGlobales.cargarguardar=true;
         }
         catch(Exception e){
            System.out.println(e.getMessage());
         }
```

```
   }

}

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {

   String ingr="";
   String nutr="";

   Properties prop = new Properties();
   prop.setProperty("NumIngredientes", jTable1.getRowCount()+"");
   prop.setProperty("NumNutrientes", jTable2.getRowCount()+"");

   for (int i=0; i< jTable1.getRowCount(); i++){
      String datos = jTable1.getValueAt(i,0)+"";
      String min = jTable1.getValueAt(i,1)+"";
      String max = jTable1.getValueAt(i,2)+"";
      String precio = jTable1.getValueAt(i,3)+"";

      Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
      String ingred = ingrediente.Nombre();
      String minim = ingrediente.Min();
      String maxim = ingrediente.Max();
      String prec = ingrediente.Precio();

      ingr=ingred+"//"+minim+"//"+maxim+"//"+prec;
      prop.setProperty("Ingrediente"+(i+1),ingr);
   }

   for (int i=0; i< jTable2.getRowCount(); i++){

      String datos = jTable2.getValueAt(i,0)+"";
      String min = jTable2.getValueAt(i,1)+"";
      String max = jTable2.getValueAt(i,2)+"";

      Nutriente nutriente = new Nutriente("",datos,min,max);
      String nutri = nutriente.Nombre();
      String minim = nutriente.Min();
      String maxim = nutriente.Max();

      nutr=nutri+"//"+minim+"//"+maxim;
      prop.setProperty("Nutriente"+(i+1),nutr);
   }


   JFileChooser chooser = new JFileChooser(new File(System.getProperty("user.home")));
   FileNameExtensionFilter filter = new FileNameExtensionFilter("Ficheros de dieta",
       "die");
   chooser.setFileFilter(filter);
   int setVal = chooser.showSaveDialog(null);
   if(setVal == JFileChooser.APPROVE_OPTION) {
      try{
         System.out.println("Has elegido guardar este archivo: " +
             chooser.getSelectedFile().getAbsolutePath()+".die");

         prop.setProperty("NombreDeDieta",chooser.getSelectedFile().getAbsolutePath()+".die");
         this.setTitle("Nombre
             "+"("+chooser.getSelectedFile().getAbsolutePath()+".die"+")");
```

```
        try {
           File f = new File(chooser.getSelectedFile().getAbsolutePath()+".die");
           OutputStream out;
           out = new FileOutputStream( f );
           prop.store(out, ingr);
        } catch (IOException ex) {
        Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
      }

      datosGlobales.nombreFichero =
          chooser.getSelectedFile().getAbsolutePath()+".die";
      System.out.println(datosGlobales.nombreFichero);
   }   catch(Exception e){
   System.out.println(e.getMessage());
}
}
}

private void jMenuItem9ActionPerformed(java.awt.event.ActionEvent evt) {
   int jj=0;
   if(jTable1.isEditing())
   jTable1.getCellEditor().stopCellEditing();
   if(jTable2.isEditing())
   jTable2.getCellEditor().stopCellEditing();

   jTabbedPane1.setSelectedIndex(1);
   jMenuItem4.setEnabled(true);
   JTextArea texto = jTextArea1;

   int ingr[] = new int[jTable1.getRowCount()];
   int nutr[] = new int[jTable2.getRowCount()];
   datosGlobales.c = new double[jTable1.getRowCount()];
   datosGlobales.l = new double[jTable1.getRowCount()];
   datosGlobales.u = new double[jTable1.getRowCount()];
   datosGlobales.bL = new double[jTable2.getRowCount()];
   datosGlobales.bU = new double[jTable2.getRowCount()];
   datosGlobales.A = new double[jTable2.getRowCount()][jTable1.getRowCount()];

   for (int i=0; i< jTable1.getRowCount(); i++){
      String temp= (String)(jTable1.getValueAt(i,0));
      ingr[i] = Integer.parseInt(temp.split(":")[0]);
   }
   for (int j=0; j< jTable2.getRowCount(); j++){
      String temp= (String)(jTable2.getValueAt(j,0));
      nutr[j] = Integer.parseInt(temp.split(":")[0]);
   }

   for (int i=0; i< jTable1.getRowCount(); i++){
      try{
         c[i] = Double.parseDouble(jTable1.getValueAt(i,3).toString());
      }catch(NumberFormatException nfe){
      jTabbedPane1.setSelectedIndex(0);
      JOptionPane.showMessageDialog(this, "Formato numérico incorrecto
          ("+jTable1.getValueAt(i,3).toString()+")");
      jTable1.changeSelection(i, 3, false, false);
      jTable1.requestFocus();
      jTable1.editCellAt(i, 3);
```

```
      return;
   }
   //System.out.println(c[i]);
}

for (int i=0; i< jTable1.getRowCount(); i++){

   try{
      jj=1;
      double a=(Double.parseDouble(jTable1.getValueAt(i,jj).toString()))/100;
      jj=2;
      double b=(Double.parseDouble(jTable1.getValueAt(i,jj).toString()))/100;
      l[i] = a;
      u[i] = b;


   }catch(NumberFormatException nfe){
   jTabbedPane1.setSelectedIndex(0);
   JOptionPane.showMessageDialog(this, "Formato numérico incorrecto
       ("+jTable1.getValueAt(i,jj).toString()+")");
   jTable1.changeSelection(i, jj, false, false);
   jTable1.requestFocus();
   jTable1.editCellAt(i, jj);
   return;
}

}

for (int j=0; j< jTable2.getRowCount(); j++){
   try{
      jj=1;
      bL[j] = Double.parseDouble(jTable2.getValueAt(j,jj).toString());
      jj=2;
      bU[j] = Double.parseDouble(jTable2.getValueAt(j,jj).toString());
   }catch(NumberFormatException nfe){
   jTabbedPane1.setSelectedIndex(0);
   JOptionPane.showMessageDialog(this, "Formato numérico incorrecto
       ("+jTable2.getValueAt(j,jj).toString()+")");
   jTable2.changeSelection(j, jj, false, false);
   jTable2.requestFocus();
   jTable2.editCellAt(j, jj);
   return;
}
//System.out.println(bL[j]);
for (int i=0; i< jTable1.getRowCount(); i++){
   A[j][i] = datosGlobales.tablaAlimentos[ingr[i]-1][nutr[j]];
   //System.out.println(A[j][i]);
}

//System.out.println(bU[j]);
}



int confirmar=JOptionPane.showConfirmDialog(null,
"¿Quiere mostrar el problema en la pestaña de fórmulas? ");

texto.setText("PPL generado: \n\n");
```

```
if(JOptionPane.OK_OPTION==confirmar) {
   // Función objetivo
   texto.append("Min Z= ");
   for (int i=0; i< jTable1.getRowCount()-1; i++){
      texto.append(""+jTable1.getValueAt(i,3)+"X"+ingr[i]+"+");
   }
   texto.append(jTable1.getValueAt(jTable1.getRowCount()-1,3)+"X"+
           ingr[jTable1.getRowCount()-1]);
   texto.append("\n");


   // Restricción de suma igual a 1
   for (int i=0; i< jTable1.getRowCount()-1; i++){
      texto.append("X"+ingr[i]+"+");
   }
   texto.append("X"+ingr[jTable1.getRowCount()-1]);
   texto.append("= 1");
   texto.append("\n");


   // Restricciones de ingredientes
   for (int i=0; i< jTable1.getRowCount(); i++){
      double a=(Double.valueOf(jTable1.getValueAt(i,1).toString()).doubleValue())/100;
      double b=(Double.valueOf(jTable1.getValueAt(i,2).toString()).doubleValue())/100;
      //texto.append(a+"<= X"+ingr[i]+"<="+b+"   "+"Ingediente "+ingr[i]+"\n");
      texto.append( String.format("%1$-16s",a+"<= X"+ingr[i]+"<="+b)+"
          "+jTable1.getValueAt(i, 0)+"\n");
   }

   texto.append("\n");

   // Restricciones de nutrientes
   for (int j=0; j< jTable2.getRowCount(); j++){
      texto.append(jTable2.getValueAt(j,1)+"<=");

      for (int i=0; i< jTable1.getRowCount()-1; i++){
         texto.append(datosGlobales.tablaAlimentos[ingr[i]-1]
               [nutr[j]]+"X"+ingr[i]+"+");
      }
      texto.append(datosGlobales.tablaAlimentos[ingr[jTable1.getRowCount()-1]-1]
               [nutr[j]]+"X"+ingr[jTable1.getRowCount()-1]);
      // texto.append("<="+jTable2.getValueAt(j,2)+"  "+"Nutriente "+nutr[j]+"\n");
      texto.append("<="+jTable2.getValueAt(j,2)+"  "+jTable2.getValueAt(j, 0)+"\n");
   }
}
}

private void MenuPrincipalComponentResized(java.awt.event.ComponentEvent evt) {
   // TODO add your handling code here:
}

private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {

   jMenuItem5.setEnabled(true);

   JTable tabla1 = this.jTable1;
   JTable tabla2 = this.jTable2;
   JTextArea texto = this.jTextArea1;
```

```
   PPLs opt = new PPLs(tabla1,tabla2,texto);
   this.jTextArea1.append("\n");
   opt.resuelveProblema();
   opt.escribeSolucion();


}

private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
   //conecta a la base de datos
   baseDeDatos tabla = new baseDeDatos();
   String ruta = null;
   tabla.conectar(ruta);
   String datos=null;
   //almacena los nuevos precios
   String precio=null;
   for (int i=0; i< jTable1.getRowCount(); i++){
      datos = jTable1.getValueAt(i,0)+"";
      precio = jTable1.getValueAt(i,3)+"";
      tabla.guardaPrecio(datos.split(":")[0], precio);
   }

   //Actualiza la pestaña de base de datos
   datosGlobales.listaNutrientes=tabla.obtenListaNutrientes();
   datosGlobales.listaIngredientes = tabla.obtenListaIngredientes();
   datosGlobales.listaUnidades=tabla.obtenListaUnidadesNutrientes();
   datosGlobales.tablaAlimentosSS= tabla.obtenMatrizAString();
   datosGlobales.tablaAlimentos=tabla.obtenMatrizA();
   jTable3.setModel(new DefaultTableModel(datosGlobales.tablaAlimentosSS,
   datosGlobales.listaNutrientes));
   tabla.cierraConexion();
}

private void jTable3MousePressed(java.awt.event.MouseEvent evt) {
   JTable table =(JTable) evt.getSource();
   Point point = evt.getPoint();
   int row = table.rowAtPoint(point);
   int col =table.columnAtPoint(point);

   if (evt.getClickCount() == 2 && col==0) {
      System.out.println("--"+table.getValueAt(table.getSelectedRow(), 0));
      String dato =""+table.getValueAt(table.getSelectedRow(), 0);
      for(int i=0;i<jComboBox2.getItemCount();i++) {
         if(jComboBox2.getItemAt(i).equals(dato))return;
      }
      DefaultTableModel modelo = (DefaultTableModel) jTable1.getModel();
      String [] fila = new String[4];
      fila[0]= dato;
      fila[1]="0";
      fila[2]="100";
      fila[3]=datosGlobales.tablaAlimentosSS[-1+Integer.parseInt(dato.split(":")[0])]
              [datosGlobales.tablaAlimentos[0].length-1];
      modelo.addRow(fila);
      this.jComboBox1.removeItem(dato);
      jComboBox2.addItem(dato);
      jTable1.setModel(modelo);
      System.out.println("Número de ingredientes "+jTable1.getRowCount());
      for (int i=0; i< jTable1.getRowCount(); i++){
```

```
            String datos = jTable1.getValueAt(i,0)+"";
            String min = jTable1.getValueAt(i,1)+"";
            String max = jTable1.getValueAt(i,2)+"";
            String precio = jTable1.getValueAt(i,3)+"";

            Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
            String ingred = ingrediente.Nombre();
            String minim = ingrediente.Min();
            String maxim = ingrediente.Max();
            String prec = ingrediente.Precio();
            System.out.println(ingred+"::"+minim+"::"+maxim+"::"+prec);
        }
        System.out.println("\n");
    }

    jMenuItem2.setEnabled(true);
    jMenuItem3.setEnabled(true);
    jMenuItem9.setEnabled(true);
    jMenuItem10.setEnabled(true);
}

private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {
    JTable tabla1 = this.jTable1;
    JTable tabla2 = this.jTable2;
    JTextArea texto = this.jTextArea1;
    PPLs opt = new PPLs(tabla1,tabla2,texto);
    GeneraExcel ge = new GeneraExcel(tabla1,tabla2,opt);
    //texto.append("\n\nSe ha generado un informe en Excel.");
}

private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void jMenuItem10ActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel modelo1 = (DefaultTableModel) jTable1.getModel();
    DefaultTableModel modelo2 = (DefaultTableModel) jTable2.getModel();
    jTextArea1.setText("");

    // Aparece ventana con opciones para poder guardar el diseño actual antes de
        vaciar las tablas
    if (jTable1.getRowCount()!=0 || jTable2.getRowCount()!=0){

        int resp = JOptionPane.showOptionDialog(null,
        "¿Quiere guardar el diseño actual?", "Seleccionar una Opción",
            JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, null,
            null);

        // Si la respuesta es si,
        if (resp == JOptionPane.YES_OPTION){
            String ingr="";
            String nutr="";

            Properties prop = new Properties();
            prop.setProperty("NumIngredientes", jTable1.getRowCount()+"");
            prop.setProperty("NumNutrientes", jTable2.getRowCount()+"");

            for (int i=0; i< jTable1.getRowCount(); i++){
```

```
            String datos = jTable1.getValueAt(i,0)+"";
            String min = jTable1.getValueAt(i,1)+"";
            String max = jTable1.getValueAt(i,2)+"";
            String precio = jTable1.getValueAt(i,3)+"";

            Ingrediente ingrediente = new Ingrediente("",datos,min,max,precio);
            String ingred = ingrediente.Nombre();
            String minim = ingrediente.Min();
            String maxim = ingrediente.Max();
            String prec = ingrediente.Precio();

            ingr=ingred+"//"+minim+"//"+maxim+"//"+prec;
            prop.setProperty("Ingrediente"+(i+1),ingr);
        }

        for (int i=0; i< jTable2.getRowCount(); i++){

            String datos = jTable2.getValueAt(i,0)+"";
            String min = jTable2.getValueAt(i,1)+"";
            String max = jTable2.getValueAt(i,2)+"";

            Nutriente nutriente = new Nutriente("",datos,min,max);
            String nutri = nutriente.Nombre();
            String minim = nutriente.Min();
            String maxim = nutriente.Max();

            nutr=nutri+"//"+minim+"//"+maxim;
            prop.setProperty("Nutriente"+(i+1),nutr);
        }

        JFileChooser chooser = new JFileChooser(new
            File(System.getProperty("user.home")));
        FileNameExtensionFilter filter = new FileNameExtensionFilter("Ficheros de
            dieta", "die");
        chooser.setFileFilter(filter);
        int setVal = chooser.showSaveDialog(null);
        if(setVal == JFileChooser.APPROVE_OPTION) {
            System.out.println("Has elegido guardar este archivo: " +
                chooser.getSelectedFile().getAbsolutePath());

            prop.setProperty("NombreDeDieta",chooser.getSelectedFile().getAbsolutePath());
            this.setTitle("CompFeed v1.0");
            datosGlobales.cargarguardar=false;

            try {
                File f = chooser.getSelectedFile();
                OutputStream out;
                out = new FileOutputStream( f );
                prop.store(out, ingr);
            } catch (IOException ex) {
            Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    //limpiar tablas
    int fila1=jTable1.getRowCount();
    int fila2=jTable2.getRowCount();
    jComboBox1.setModel(new DefaultComboBoxModel(datosGlobales.listaIngredientes));
```

```
      jComboBox3.setModel(new DefaultComboBoxModel(datosGlobales.listaNutrientes));
      jComboBox3.removeItemAt(0);
      int desp1 = jComboBox2.getItemCount();
      int desp2 = jComboBox4.getItemCount();

      for (int i = 0;fila1>i; i++) {
         modelo1.removeRow(0);
      }
      for (int i = 0;fila2>i; i++) {
         modelo2.removeRow(0);
      }
      for(int i=0;i<desp1;i++){
         jComboBox2.removeItemAt(0);
      }
      for(int i=0;i<desp2;i++){
         jComboBox4.removeItemAt(0);
      }
   }

   if (resp == JOptionPane.NO_OPTION){
      //limpiar tablas
      int fila1=jTable1.getRowCount();
      int fila2=jTable2.getRowCount();
      jComboBox1.setModel(new DefaultComboBoxModel(datosGlobales.listaIngredientes));
      jComboBox3.setModel(new DefaultComboBoxModel(datosGlobales.listaNutrientes));
      jComboBox3.removeItemAt(0);
      int desp1 = jComboBox2.getItemCount();
      int desp2 = jComboBox4.getItemCount();

      for (int i = 0;fila1>i; i++) {
         modelo1.removeRow(0);
      }
      for (int i = 0;fila2>i; i++) {
         modelo2.removeRow(0);
      }
      for(int i=0;i<desp1;i++){
         jComboBox2.removeItemAt(0);
      }
      for(int i=0;i<desp2;i++){
         jComboBox4.removeItemAt(0);
      }

      this.setTitle("CompFeed v1.0");
      datosGlobales.cargarguardar=false;
   }

   if (resp == JOptionPane.CANCEL_OPTION){
      this.setDefaultCloseOperation(WindowConstants.HIDE_ON_CLOSE);
   }
 }

}

private void jTable1FocusLost(java.awt.event.FocusEvent evt) {
   if(jTable1.isEditing())
   jTable1.getCellEditor().stopCellEditing();
}
```

```
private void jTable2FocusLost(java.awt.event.FocusEvent evt) {
   if(jTable2.isEditing())
   jTable2.getCellEditor().stopCellEditing();
}

private void jTextArea1MousePressed(java.awt.event.MouseEvent evt) {
   // TODO add your handling code here:
   if(evt.getButton()==MouseEvent.BUTTON3){
      JFileChooser fc = new JFileChooser();
      int Val = fc.showSaveDialog(null);

      if (Val == JFileChooser.APPROVE_OPTION) {
         File file = fc.getSelectedFile();
         String ss=jTextArea1.getText();
         FileWriter fileWriter = null;
         try{
            if (!file.exists()) {
               file.createNewFile();
            }

            fileWriter = new FileWriter(file);
            fileWriter.write(ss);
            fileWriter.close();
         }
         catch(IOException fnofe){}
      }
   }
}

private void jMenu6MouseClicked(java.awt.event.MouseEvent evt) {
   new AcercaDe(null,true).setVisible(true);
}


/**
* @param args the command line arguments
*/
public static void main(String args[]) {
   /* Set the Nimbus look and feel */
   //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
       (optional) ">
   /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
       look and feel.
   * For details see
       http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
   */
   try {
      for (javax.swing.UIManager.LookAndFeelInfo info :
          javax.swing.UIManager.getInstalledLookAndFeels()) {
         if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
         }
      }
   } catch (ClassNotFoundException ex) {
   java.util.logging.Logger.getLogger(Ventana.class.getName()).log(java.util.logging.Level.
          SEVERE, null, ex);
} catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(Ventana.class.getName()).log(java.util.logging.Level.
          SEVERE, null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Ventana.class.getName()).log(java.util.logging.Level.
          SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Ventana.class.getName()).log(java.util.logging.Level.
          SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
      new Ventana().setVisible(true);
   }
});
}

// Variables declaration - do not modify
private javax.swing.JMenuBar MenuPrincipal;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
private javax.swing.JComboBox jComboBox3;
private javax.swing.JComboBox jComboBox4;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenu jMenu3;
private javax.swing.JMenu jMenu5;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem10;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JMenuItem jMenuItem4;
private javax.swing.JMenuItem jMenuItem5;
private javax.swing.JMenuItem jMenuItem6;
private javax.swing.JMenuItem jMenuItem7;
private javax.swing.JMenuItem jMenuItem8;
private javax.swing.JMenuItem jMenuItem9;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane4;
private javax.swing.JPopupMenu.Separator jSeparator1;
private javax.swing.JPopupMenu.Separator jSeparator2;
private javax.swing.JPopupMenu.Separator jSeparator3;
private javax.swing.JPopupMenu.Separator jSeparator4;
private javax.swing.JPopupMenu.Separator jSeparator5;
private javax.swing.JPopupMenu.Separator jSeparator6;
```

```
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTable jTable2;
private javax.swing.JTable jTable3;
private javax.swing.JTextArea jTextArea1;
// End of variables declaration
private javax.swing.JMenu jMenu6;
}
```

C.2. Clase baseDeDatos

```
package programadietas;

/**
*
* @author pmateo
*/


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.NumberFormat;
import java.util.Locale;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
* Clase simple que se encarga de pasar la conexión a la base de datos.
* @author Gustavo Hernández Salinas
*/
public final class baseDeDatos {
   //Declaramos la conexión:
   private Connection conn = null;
   private int numeroIngredientes;
   private int numeroNutrientes;
   private String ruta=null;

   /**
   * Conecta a una base de datos en la ruta pasada como argumento.
   * Si el archivo señalado en la ruta no existe, se crea.
   * @param ruta - La ruta de la DB a conectar.
   * @return La conexión a la ruta.
   */
   protected Connection conectar(String ruta){
      try {
         //Aquí cargamos el driver de SQLITE.
         Class.forName("org.sqlite.JDBC");
      }
      catch (ClassNotFoundException e) {
```

```
            //Esto se ejecuta si hay un error con el driver de la base de datos.
            e.printStackTrace();
        }



        try {
            //Aquí se obtiene la conexión:
            conn = DriverManager.getConnection("jdbc:sqlite:" +
                "C:/Users/Andreita/Desktop/Matemáticas/Trabajo Fin de
                Grado/ProgramaDietas/src/programadietas/FEDNA2015BIS");

            //Un mensaje en la consola para saber si se realizó la conexión y donde está
                el archivo:
            System.out.println("Conexión realizada correctamente - Ruta de base de datos:
                " + "C:/Users/Andreita/Desktop/Matemáticas/Trabajo Fin de
                Grado/ProgramaDietas/src/programadietas/FEDNA2015BIS");
        }
        catch (SQLException e) {
            //Esto se ejecuta si hay un error en la base de datos:
            e.printStackTrace();
        }

        //Devolvemos la conexión:
        return conn;
    }

    void cierraConexion(){
        try {
            conn.close();
        } catch (SQLException ex) {
        Logger.getLogger(baseDeDatos.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
* Determina el número de ingredientes, es decir el número de registros
* en la base de alimentos/ingredientes
*/
private void obtenNumeroIngredientes(){
    ResultSet rs=null;
    numeroIngredientes=0;
    try {
        Statement statement = conn.createStatement();
        rs = statement.executeQuery("SELECT Count(*) FROM Ingredientes");
        numeroIngredientes=Integer.parseInt(rs.getString(1));
    }
    catch (SQLException e) {
        //Esto se ejecuta si hay algún problema al realizar la conexión.
        e.printStackTrace();
    }

}
/**
* Determina el número de nutrientes para cada ingrediente. Incluye
* el codigo y el precio del ingrediente.
*/
private void obtenNumeroNutrientes(){
```

```
   ResultSet rs=null;
   numeroNutrientes=0;
   try {
      Statement statement = conn.createStatement();
      rs = statement.executeQuery("SELECT Count(*) FROM Nutrientes");
      numeroNutrientes=Integer.parseInt(rs.getString(1));
   }
   catch (SQLException e) {
      //Esto se ejecuta si hay algún problema al realizar la conexión.
      e.printStackTrace();
   }
   System.out.println("Numero nutrientes="+numeroNutrientes);
}


String [] obtenListaIngredientes(){
   ResultSet rs=null;
   String lista[]=null;
   obtenNumeroIngredientes();
   lista = new String[numeroIngredientes];
   try {
      Statement statement = conn.createStatement();
      rs = statement.executeQuery("select CODE_INGREDIENTE,INGREDIENT from
          Ingredientes");
      int i=0;
      while(rs.next())
      {
         lista[i]=rs.getString("CODE_INGREDIENTE")+":"+rs.getString("INGREDIENT");
         i++;
      }
   }
   catch (SQLException e) {
      //Esto se ejecuta si hay algún problema al realizar la conexión.
      e.printStackTrace();
   }

   return lista;

}




String [] obtenListaNutrientes(){
   ResultSet rs=null;
   String lista[]=null;
   obtenNumeroNutrientes();
   lista = new String[numeroNutrientes+1];
   lista[0]="Nombre";
   try {
      Statement statement = conn.createStatement();
      rs = statement.executeQuery("select CODE_NUTRIENTE,ABREVIATUR,UNIDADES_S from
          Nutrientes");
      int i=0;
      while(rs.next())
      {
         lista[i+1]=""+rs.getString("CODE_NUTRIENTE")+":"+rs.getString("ABREVIATUR")+"
             "+rs.getString("UNIDADES_S");//+"--->"+rs.getString("DESCRIPCIO")+"--->"+rs.getString("U
```

```
        i++;

      }
   }
   catch (SQLException e) {
      //Esto se ejecuta si hay algún problema al realizar la conexión.
      e.printStackTrace();
   }

   return lista;
}

String [] obtenListaUnidadesNutrientes(){
   ResultSet rs=null;
   String lista[]=null;
   obtenNumeroNutrientes();
   lista = new String[numeroNutrientes];
   try {
      Statement statement = conn.createStatement();
      rs = statement.executeQuery("select UNIDADES_S from Nutrientes");
      int i=0;
      while(rs.next())
      {
         lista[i]=rs.getString("UNIDADES_S");//+"--->"+
                 rs.getString("DESCRIPCIO")+"--->"+rs.getString("UNIDADES_S");
         i++;
      }
   }
   catch (SQLException e) {
      //Esto se ejecuta si hay algún problema al realizar la conexión.
      e.printStackTrace();
   }

   return lista;
}

double[][] obtenMatrizA(){
   this.obtenNumeroIngredientes();
   this.obtenNumeroNutrientes();
   double A[][]= new double[this.numeroIngredientes][this.numeroNutrientes+1];//La
       primera posicion para el codigo
   NumberFormat format = NumberFormat.getInstance(Locale.FRANCE);
   Number number = null;

   ResultSet rs=null;
   try {
      Statement statement = conn.createStatement();
      statement.setMaxRows(500);
      rs = statement.executeQuery("select * from Ingredientes");
      int i=0;

      while(rs.next())
      {
         try{
            A[i][0]=Double.parseDouble(rs.getString(1));
         }catch(NumberFormatException nfe){
         System.out.println("****"+rs.getString(1)+"i="+i+"j="+0);
      }
```

```
      for(int j=0;j<this.numeroNutrientes;j++){
         try{
            A[i][j+1]=Double.parseDouble(rs.getString(j+3));
         }catch(NumberFormatException nfe){
         System.out.println("*BB**"+rs.getString(j+3)+"i="+i+"j="+j);
      }
   }
   i++;

}
}
catch (SQLException e) {
   //Esto se ejecuta si hay algún problema al realizar la conexión.
   e.printStackTrace();
}

return A;

}

String[][] obtenMatrizAString(){
   this.obtenNumeroIngredientes();
   this.obtenNumeroNutrientes();
   String A[][]= new String[this.numeroIngredientes][1+this.numeroNutrientes];
   NumberFormat format = NumberFormat.getInstance(Locale.FRANCE);
   Number number = null;

   ResultSet rs=null;
   try {
      Statement statement = conn.createStatement();
      statement.setMaxRows(500);
      rs = statement.executeQuery("select * from Ingredientes");
      int i=0;

      while(rs.next())
      {
         try{
            A[i][0]=rs.getString(1)+":"+rs.getString(2);
         }catch(NumberFormatException nfe){
         System.out.println("*ff*"+rs.getString(1)+":"+rs.getString(2)+"i="+i+"j="+0);
      }
      for(int j=1;j<this.numeroNutrientes+1;j++){
         //try{
            // number=format.parse(rs.getString(j+2));
            //A[i][j]=number.doubleValue();
            //}catch(ParseException pae){System.out.println(rs.getString(j+2)+"i="+i);}
         try{
            A[i][j]=rs.getString(j+2);
         }catch(NumberFormatException nfe){
         System.out.println("*lll*"+rs.getString(j+2)+"i="+i+"j="+j);
      }
   }
   i++;

}
}
catch (SQLException e) {
   //Esto se ejecuta si hay algún problema al realizar la conexión.
```

```
   e.printStackTrace();
}

return A;

}


void guardaPrecio(String codeIngrediente, String valor){
   try {
      System.out.println("update Ingredientes set Precio="+valor+
      " where CODE_INGREDIENTE="+codeIngrediente);
      Statement statement = conn.createStatement();
      statement.execute("update Ingredientes set Precio="+valor+
      " where CODE_INGREDIENTE="+codeIngrediente);
   }
   catch (SQLException e) {
      //Esto se ejecuta si hay algún problema al realizar la conexión.
      e.printStackTrace();
   }

 }

}
```

C.3. Clase datosGlobales

```
package programadietas;

import java.util.ArrayList;

/**
*
* @author aaguilar, pmateo
*/
public class datosGlobales {
   static baseDeDatos bdd=null;
   static String[] listaIngredientes=null;
   static String[] listaNutrientes=null;
   static String[] listaUnidades=null;
   static String[][] tablaAlimentosSS=null;
   static double[][] tablaAlimentos=null;
   boolean baseDedatosModificada=false;

   static ArrayList<Ingrediente> Ingrediente = new ArrayList<Ingrediente>();
   static ArrayList<Nutriente> Nutriente = new ArrayList<Nutriente>();

   public static double [][] A = null;
   public static double [] bL = null;
   public static double [] bU = null;
   public static double [] c = null;
   public static double [] l = null;
```

```
   public static double [] u = null;

   static boolean cargarguardar=false;
   static String nombreFichero=null;


   /**
   * Se encarga de dibujar la matriz completa de ingredientes con sus nutrientes
   *
   */
   static void pintaMatrizA(){
      for(int i=0;i<tablaAlimentos.length;i++) {
         for(int j=0;j<tablaAlimentos[i].length;j++)
         System.out.print(tablaAlimentos[i][j]+" ");
         System.out.println();
      }

   }

}
```

C.4. Clase Ingrediente

```
package programadietas;

import javax.swing.JOptionPane;
import javax.swing.JTable;

/**
*
* @author aaguilar, pmateo
*/
public class Ingrediente {

   private String codigo;
   private String nombre;
   private String min;
   private String max;
   private String precio;


   public Ingrediente(String codigo, String nombre, String min, String max, String
       precio){
      this.codigo = codigo;
      this.nombre = nombre;
      this.min = min;
      this.max = max;
      this.precio = precio;
   }


   @Override
```

```
   public String toString() {
      return this.nombre;
   }

   public String Codigo() {
      return this.codigo;
   }

   public String Nombre() {
      return this.nombre;
   }

   public String Min() {
      return this.min;
   }

   public String Max() {
      return this.max;
   }

   public String Precio() {
      return this.precio;
   }

}
```

C.5. Clase Nutriente

```
package programadietas;

import javax.swing.JTable;

/**
*
* @author aaguilar, pmateo
*/
public class Nutriente {

   private String codigo, min, max;
   private String nombre;



   public Nutriente(String codigo, String nombre, String min, String max){
      this.codigo = codigo;
      this.nombre = nombre;
      this.min = min;
      this.max = max;

   }

   @Override
```

```
   public String toString() {
      return this.nombre;
   }

   public String Codigo() {
      return this.codigo;
   }

   public String Nombre() {
      return this.nombre;
   }

   public String Min() {
      return this.min;
   }

   public String Max() {
      return this.max;
   }
}
```

C.6. Clase PPLs

```
package programadietas;

import java.awt.TextArea;
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import org.gnu.glpk.GLPK;
import org.gnu.glpk.GLPKConstants;
import static org.gnu.glpk.GLPKConstants.GLP_OPT;
import org.gnu.glpk.GlpkException;
import org.gnu.glpk.SWIGTYPE_p_double;
import org.gnu.glpk.SWIGTYPE_p_int;
import org.gnu.glpk.glp_prob;
import org.gnu.glpk.glp_smcp;
import static programadietas.datosGlobales.A;
import static programadietas.datosGlobales.bL;
import static programadietas.datosGlobales.bU;
import static programadietas.datosGlobales.c;
import static programadietas.datosGlobales.l;
import static programadietas.datosGlobales.u;

/**
*
* @author aaguilar, pmateo
* @version 0.0
*/

public class PPLs {
```

```
protected glp_prob lp=null;
protected glp_smcp parm=null;
protected JTable ingredientes;
protected JTable nutrientes;
protected JTextArea texto;

/**
*
* @param ingredientesArg, primer jTable de la pestaña Ingredientes y nutrientes
* @param nutrientesArg, segundo jTable de la pestaña Ingredientes y nutrientes
* @param textoArg, TextArea de la pestaña formulas en la que se escriben resultados
*/
PPLs(JTable ingredientesArg,JTable nutrientesArg,JTextArea textoArg){
   ingredientes=ingredientesArg;
   nutrientes=nutrientesArg;
   texto=textoArg;

   lp = GLPK.glp_create_prob();
   //En la linea siguiente deberás mirar si la información está guarda en
   //un fichero, en cuyo caso se le puede poner ese nombre al problema
   //En otro caso se puede poner sinNombre (lo que aparece ahora)
   GLPK.glp_set_prob_name(lp,"sinNombre") ;



   //Variables
   GLPK.glp_add_cols(lp,ingredientes.getRowCount()); // el número de columnas es
       el número de ingredientes
   for (int i=0; i< ingredientes.getRowCount(); i++){
      GLPK.glp_set_col_name(lp, i+1, ""+ingredientes.getValueAt(i,0)); //establece
          el nombre de los ingredientes que se usan
      GLPK.glp_set_col_kind(lp, i+1,GLPKConstants.GLP_CV); //establece que las
          variables son continuas
      GLPK.glp_set_col_bnds(lp, i+1,GLPKConstants.GLP_DB,l[i],u[i]); //establece
          que las variables están acotadas por encima y por debajo
   }
   SWIGTYPE_p_int indices;
   SWIGTYPE_p_double valores;
   indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
   valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
   GLPK.glp_add_rows(lp,nutrientes.getRowCount()+1); // el número de filas es el
       número de nutrientes


   //Primera restricción suma igual a 1
   GLPK.glp_set_row_name(lp,1,"Total");
   GLPK.glp_set_row_bnds(lp,1,GLPKConstants.GLP_FX,1.0,1.0); //cota inferior y
       superior iguales
   for (int i=0; i< ingredientes.getRowCount(); i++){
      GLPK.intArray_setitem(indices, i+1, i+1);
      GLPK.doubleArray_setitem(valores, i+1, 1.0);
   }
   GLPK.glp_set_mat_row(lp,1,ingredientes.getRowCount(),indices,valores);
   GLPK.delete_intArray(indices);
   GLPK.delete_doubleArray(valores);
```

```
    //Restricciones de nutrientes
    for(int j=0;j<nutrientes.getRowCount();j++){
       indices = GLPK.new_intArray(ingredientes.getRowCount()+1);
       valores = GLPK.new_doubleArray(ingredientes.getRowCount()+1);
       GLPK.glp_set_row_name(lp,j+2,""+nutrientes.getValueAt(j, 0));
       GLPK.glp_set_row_bnds(lp,j+2,GLPKConstants.GLP_DB,bL[j],bU[j]);
       for (int i=0; i< ingredientes.getRowCount(); i++){
          GLPK.intArray_setitem(indices, i+1, i+1);
          GLPK.doubleArray_setitem(valores, i+1, A[j][i]);

       }
       GLPK.glp_set_mat_row(lp,j+2,ingredientes.getRowCount(),indices,valores);
       GLPK.delete_intArray(indices);
       GLPK.delete_doubleArray(valores);
    }


    //Función objetivo
    GLPK.glp_set_obj_dir(lp,GLPK.GLP_MIN); // indica que el problema es de minimo
    GLPK.glp_set_obj_name(lp, "Z");
    GLPK.glp_set_obj_coef(lp,0,0.0);
    for (int i=0; i< ingredientes.getRowCount(); i++){
       GLPK.glp_set_obj_coef(lp,i+1,c[i]);
    }


    //Guardamos el problema en formato cplex
    GLPK.glp_write_lp(lp, null, "sinNombre.txt");
}

PPLs() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body
        of generated methods, choose Tools | Templates.
}


int resuelveProblema(){
    //Resolvemos
    int retorno=0;
    try{
       parm = new glp_smcp();
       GLPK.glp_init_smcp(parm);
       retorno = GLPK.glp_simplex(lp,parm);

       if(GLPK.glp_get_status(lp)==GLP_OPT){
          //escribeSolucion();
          análisisParametrico();
          variablesDuales();
       }
       else{
          JOptionPane.showMessageDialog(null, "El problema es no factible. No se
              puede resolver.");
          System.out.println("Algo ha pasado, aun no considerado");
       }
    }
    catch(GlpkException ex){
       ex.printStackTrace();
       GLPK.glp_delete_prob(lp);
```

```
    }


    int numVar=GLPK.glp_get_num_cols(lp);
    int numRes=GLPK.glp_get_num_rows(lp);
    System.out.println("La formulación tiene "+numVar+
    " Ingredientes y "+numRes+" requerimientos de nutrientes.");

    System.out.println("Ingredientes utilizados:");
    for(int i=0;i<numVar;i++){
       if(GLPK.glp_get_col_prim(lp, i+1)>0)
       System.out.println(""+GLPK.glp_get_col_name(lp, i+1)+
       "   Aportación ="+String.format("%6.2f", 100*GLPK.glp_get_col_prim(lp,
           i+1))+"%");
    }

    System.out.println("Ingredientes no utilizados:");
    for(int i=0;i<numVar;i++){
       if(GLPK.glp_get_col_prim(lp, i+1)==0)
       System.out.println(""+GLPK.glp_get_col_name(lp, i+1)+
       "   Aportación ="+String.format("%6.2f", 100*GLPK.glp_get_col_prim(lp,
           i+1))+"%");
    }

    System.out.println("Composición lograda:");

    for(int i=1;i<numRes;i++){
       System.out.println(""+GLPK.glp_get_row_name(lp, i+1)+" valor = "+
       GLPK.glp_get_row_prim(lp, i+1)+" en ("+GLPK.glp_get_row_lb(lp,i+1)
       +","+GLPK.glp_get_row_ub(lp,i+1)+")");
    }

    return retorno;
}


int escribeSolucion(){
    DecimalFormat decimales = new DecimalFormat("0.00");
    double costo = GLPK.glp_get_obj_val(lp);
    texto.append(String.format("\nSolución de costo: %7.4f ? por Kg\n\n", costo));

    texto.append("\nIngredientes utilizados: \n");
    for(int i=1;i<=GLPK.glp_get_num_cols(lp);i++){
       if (GLPK.glp_get_col_prim(lp, i)!=0)

       texto.append(
           String.format("%1$-40s",GLPK.glp_get_col_name(lp,i)+":")+String.format("%10.3f
           %%\n",100*GLPK.glp_get_col_prim(lp, i)));
    }


    texto.append("\nIngredientes no utilizados: \n");
    for(int i=1;i<=GLPK.glp_get_num_cols(lp);i++){
       if (GLPK.glp_get_col_prim(lp, i)==0)
       //texto.append(GLPK.glp_get_col_name(lp,i)+": "+String.format("%10.3f
           ",100*GLPK.glp_get_col_prim(lp, i))+"%\n");
       texto.append(
           String.format("%1$-40s",GLPK.glp_get_col_name(lp,i)+":")+String.format("%10.3f
```

```
            %%\n",100*GLPK.glp_get_col_prim(lp, i)));
      }

      texto.append("\nComposición lograda: \n");
      for (int j=1;j<GLPK.glp_get_num_rows(lp);j++){
         texto.append(String.format("%1$-22s",GLPK.glp_get_row_name(lp,
             j+1)+":")+String.format("%10.3f ",GLPK.glp_get_row_prim(lp, j+1))+
         " en
             ("+String.format("%1$8.2f",GLPK.glp_get_row_lb(lp,j+1))+","+String.format(
                 "%1$8.2f",GLPK.glp_get_row_ub(lp,j+1))+")\n");

      }

      return 0;
   }


   int análisisDeSensibilidad(){
      SWIGTYPE_p_double coef1;
      SWIGTYPE_p_double coef2;
      coef1 = GLPK.new_doubleArray(1);
      coef2 = GLPK.new_doubleArray(1);

      int numVar=GLPK.glp_get_num_cols(lp);
      int numRes=GLPK.glp_get_num_rows(lp);


      System.out.println("La formulación se mantiene si los precios de los "+
      "ingredientes varían dentro de los rangos siguientes:");
      texto.append("\nLa formulación se mantiene si los precios de los "+
      "ingredientes varían dentro de los rangos siguientes: \n\n");

      for(int i=1;i<=numVar;i++){
         if(GLPK.glp_get_col_stat(lp,i)==GLPK.GLP_BS){
            GLPK.glp_analyze_coef(lp, i+numRes, coef1, null, null,coef2,null,null);
            System.out.println(GLPK.glp_get_col_name(lp, i)+":
                "+GLPK.glp_get_obj_coef(lp, i)+" Pudiendo variar entre "+
            GLPK.doubleArray_getitem(coef1, 0)+" y "+GLPK.doubleArray_getitem(coef2,
                0)+" *");
            texto.append(String.format("%1$-40s",GLPK.glp_get_col_name(lp,
                i)+":")+String.format("%10.3f",GLPK.glp_get_obj_coef(lp, i))+" en ("+
            String.format("%1$8.2f",GLPK.doubleArray_getitem(coef1,
                0))+","+String.format("%1$8.2f",GLPK.doubleArray_getitem(coef2,
                0))+")\n");
         }
         else{
            if(GLPK.glp_get_col_stat(lp,i)==GLPK.GLP_NL){
               System.out.println(GLPK.glp_get_col_name(lp, i)+":
                   "+GLPK.glp_get_obj_coef(lp, i)+" Pudiendo variar entre "+
               (GLPK.glp_get_obj_coef(lp,i)-GLPK.glp_get_col_dual(lp, i))+" y Inf ");
               texto.append(String.format("%1$-40s",GLPK.glp_get_col_name(lp,
                   i)+":")+String.format("%10.3f",GLPK.glp_get_obj_coef(lp, i))+" en
                   ("+
               String.format("%1$8.2f",(GLPK.glp_get_obj_coef(lp,i))-GLPK.glp_get_col_dual(
                       lp, i))+"," +String.format("%1$8s","Inf") +")\n");
            }
            else { if (GLPK.glp_get_col_stat(lp,i)==GLPK.GLP_NU)
               System.out.println(GLPK.glp_get_col_name(lp, i)+
```

```
            ":  "+GLPK.glp_get_obj_coef(lp, i)+": Pudiendo entre -Inf y
                "+(GLPK.glp_get_obj_coef(lp,i)-GLPK.glp_get_col_dual(lp, i)));
            texto.append(String.format("%1$-40s",GLPK.glp_get_col_name(lp, i)+
            ":")+String.format("%10.3f",GLPK.glp_get_obj_coef(lp, i))+": en
                ("+String.format("%1$7s", "-Inf")
                +","+String.format("%10.3f",GLPK.glp_get_obj_coef(lp,i)-
                       GLPK.glp_get_col_dual(lp, i))+")\n");
         }
      }

   }



   System.out.println("Valoración de nutrientes");
   texto.append("\n\nValoración de nutrientes:\n");
   texto.append("Una unidad adicional de requerimiento mínimo incrementaría el
       costo en la cantidad indicada.\n");
   for(int i=1;i<numRes;i++){
      if(GLPK.glp_get_row_stat(lp, i+1)==GLPK.GLP_NL){

         System.out.println(GLPK.glp_get_row_name(lp, i+1)+" (contenido
             "+String.format("%10.3f",GLPK.glp_get_row_prim(lp,i+1))
         +") "+String.format("%10.4f?",GLPK.glp_get_row_dual(lp, i+1)));
         texto.append(String.format("%1$-22s",GLPK.glp_get_row_name(lp, i+1))+"
             (contenido "+String.format("%10.3f",GLPK.glp_get_row_prim(lp,i+1))+
         ") "+String.format("%10.4f?",GLPK.glp_get_row_dual(lp, i+1))+"\n");
      }
   }
   texto.append("Una unidad menos de requerimiento máximo incrementaría el costo
       en la cantidad indicada.\n");
   for(int i=1;i<numRes;i++){
      if(GLPK.glp_get_row_stat(lp, i+1)==GLPK.GLP_NU){
         System.out.println(GLPK.glp_get_row_name(lp, i+1)+"(contenido
             "+String.format("%10.3f",GLPK.glp_get_row_prim(lp,i+1))+
         ")  "+String.format("%10.4f?",GLPK.glp_get_row_dual(lp, i+1)));
         texto.append(String.format("%1$-22s",GLPK.glp_get_row_name(lp, i+1))+"
             (contenido "+String.format("%10.3f",GLPK.glp_get_row_prim(lp,i+1))+
         ")  "+String.format("%10.4f?",GLPK.glp_get_row_dual(lp, i+1))+"\n");
      }

   }

      return 0;
   }

   int análisisParametrico(){
      return 0;
   }
   int variablesDuales(){
      return 0;
   }
}
```

C.7. Clase GeneraExcel

```
package programadietas;

import java.awt.Desktop;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Date;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.filechooser.FileNameExtensionFilter;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.hssf.util.HSSFColor;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.gnu.glpk.GLPK;
import org.gnu.glpk.GLPKConstants;
import static org.gnu.glpk.GLPKConstants.GLP_OPT;
import org.gnu.glpk.GlpkException;
import org.gnu.glpk.SWIGTYPE_p_double;
import org.gnu.glpk.SWIGTYPE_p_int;
import org.gnu.glpk.glp_prob;
import org.gnu.glpk.glp_smcp;
import static programadietas.datosGlobales.A;
import static programadietas.datosGlobales.bL;
import static programadietas.datosGlobales.bU;
import static programadietas.datosGlobales.c;
import static programadietas.datosGlobales.l;
import static programadietas.datosGlobales.u;


/**
*
* @author aaguilar, pmateo
*/
public class GeneraExcel {

   protected JTable ingredientes;
   protected JTable nutrientes;
   protected PPLs opt;
   protected glp_smcp parm=null;
   protected glp_prob lp=null;

   /**
   *
   * @param ingredientesArg, primer jTable de la pestaña Ingredientes y nutrientes
   * @param nutrientesArg, segundo jTable de la pestaña Ingredientes y nutrientes
   */
```

```
GeneraExcel(JTable ingred, JTable nutri,PPLs ppls) {
   ingredientes=ingred;
   nutrientes=nutri;
   opt=ppls;
   glp_prob lp = opt.lp;
   parm = new glp_smcp();
   GLPK.glp_init_smcp(parm);
   GLPK.glp_simplex(lp,parm);

   File ficheroExcel = null;


   try {

      try{

         JFileChooser chooser = new JFileChooser(new
             File(System.getProperty("user.home")));
         FileNameExtensionFilter filter = new FileNameExtensionFilter("Informes de
             dieta", "xls");
         chooser.setFileFilter(filter);
         int setVal = chooser.showSaveDialog(null);
         if(setVal == JFileChooser.APPROVE_OPTION) {
            System.out.println("Has elegido guardar este archivo: " +
                chooser.getSelectedFile().getAbsolutePath());

            ficheroExcel = new File (chooser.getSelectedFile().getAbsolutePath());



            if (ficheroExcel != null) {
               String nombre = ficheroExcel.getAbsolutePath();
               int i = nombre.lastIndexOf(".");
               if (i != -1) {
                  nombre = nombre.substring(0, i);
               }
               ficheroExcel = new File(nombre + ".xls");
               if (ficheroExcel.exists()) {
                  ficheroExcel.delete();
                  ficheroExcel.createNewFile();
               }
               Workbook libro = new HSSFWorkbook();
               FileOutputStream archivo = new FileOutputStream(ficheroExcel);




               //HOJA1 ---> Solución del problema
               Sheet hoja1 = libro.createSheet("Fórmula");
               CellStyle cs = libro.createCellStyle();
               cs.setDataFormat(libro.createDataFormat().getFormat("#######.##"));
               hoja1.setDefaultColumnStyle(2, cs);
               Row fila = hoja1.createRow(0);
               Cell celda = fila.createCell(0);
               fila = hoja1.createRow(0);
               celda = fila.createCell(0);
               celda.setCellValue("Solución de costo (? / Kg):");
               libro.getSheetAt(0).autoSizeColumn((short) 0);
```

```
celda = fila.createCell(1);
celda.setCellValue(String.format("%1$10.3f
    ",GLPK.glp_get_obj_val(lp)));
fila = hoja1.createRow(1);
celda = fila.createCell(0);
celda.setCellValue("Partida en Kg:");
libro.getSheetAt(0).autoSizeColumn((short) 0);
String partida = JOptionPane.showInputDialog(null,"Introduzca la
    partida en Kg",JOptionPane.QUESTION_MESSAGE);
celda=fila.createCell(1);
celda.setCellValue(partida);
fila = hoja1.createRow(3);
celda = fila.createCell(0);
celda.setCellValue("Ingredientes seleccionados:");
fila = hoja1.createRow(4);
celda = fila.createCell(1);
celda.setCellValue("Nombre");
celda = fila.createCell(2);
celda.setCellValue("Mínimo");
celda = fila.createCell(3);
celda.setCellValue("Máximo");
celda = fila.createCell(4);
celda.setCellValue("Cantidad (%)");
libro.getSheetAt(0).autoSizeColumn((short) 4);
celda = fila.createCell(5);
celda.setCellValue("Cantidad (Kg)");
libro.getSheetAt(0).autoSizeColumn((short) 5);


int num=Integer.parseInt(partida);
fila=hoja1.createRow(5);
for (int k=0;k<GLPK.glp_get_num_cols(lp);k++){
   if(GLPK.glp_get_col_prim(lp, k+1)>0){
      celda = fila.createCell(1);
      celda.setCellValue(GLPK.glp_get_col_name(lp,k+1));
      celda = fila.createCell(2);
      celda.setCellValue(String.format("%1$10.3f
          ",100*GLPK.glp_get_col_lb(lp, k+1)));
      celda = fila.createCell(3);
      celda.setCellValue(String.format("%1$10.3f
          ",100*GLPK.glp_get_col_ub(lp, k+1)));
      celda = fila.createCell(4);
      celda.setCellValue(String.format("%1$10.3f
          ",100*GLPK.glp_get_col_prim(lp, k+1)));
      celda = fila.createCell(5);
      celda.setCellValue(String.format("%1$10.3f
          ",num*GLPK.glp_get_col_prim(lp, k+1)));
      fila=hoja1.createRow(fila.getRowNum()+1);
   }
}
libro.getSheetAt(0).autoSizeColumn((short) 4);

for (int k=0;k<GLPK.glp_get_num_cols(lp);k++){
   if (GLPK.glp_get_col_prim(lp, k+1)==0){
      celda = fila.createCell(1);
      celda.setCellValue(GLPK.glp_get_col_name(lp,k+1));
      celda = fila.createCell(2);
```

```
            celda.setCellValue(String.format("%1$10.3f
                ",100*GLPK.glp_get_col_lb(lp, k+1)));
            celda = fila.createCell(3);
            celda.setCellValue(String.format("%1$10.3f
                ",100*GLPK.glp_get_col_ub(lp, k+1)));
            celda = fila.createCell(4);
            celda.setCellValue(String.format("%1$10.3f
                ",100*GLPK.glp_get_col_prim(lp, k+1)));
            celda = fila.createCell(5);
            celda.setCellValue(String.format("%1$10.3f
                ",num*GLPK.glp_get_col_prim(lp, k+1)));
            fila=hoja1.createRow(fila.getRowNum()+1);
        }
    }
    libro.getSheetAt(0).autoSizeColumn((short) 1);



    fila = hoja1.getRow(3);
    celda = fila.createCell(8);
    celda.setCellValue("Composición lograda: ");
    fila = hoja1.getRow(4);
    celda = fila.createCell(9);
    celda.setCellValue("Nombre");
    celda = fila.createCell(10);
    celda.setCellValue("Mínimo");
    celda = fila.createCell(11);
    celda.setCellValue("Máximo");
    celda = fila.createCell(12);
    celda.setCellValue("Cantidad");
    libro.getSheetAt(0).autoSizeColumn((short)12);

    for (int j=1;j<GLPK.glp_get_num_rows(lp);j++){
        fila = hoja1.getRow(j+4);
        celda = fila.createCell(9);
        celda.setCellValue(GLPK.glp_get_row_name(lp,j+1));
        celda = fila.createCell(10);
        celda.setCellValue(String.format("%1$10.3f
            ",GLPK.glp_get_row_lb(lp, j+1)));
        celda = fila.createCell(11);
        celda.setCellValue(String.format("%1$10.3f
            ",GLPK.glp_get_row_ub(lp, j+1)));
        celda = fila.createCell(12);
        celda.setCellValue(String.format("%1$10.3f
            ",GLPK.glp_get_row_prim(lp, j+1)));
    }
    libro.getSheetAt(0).autoSizeColumn((short)9);
    libro.getSheetAt(0).autoSizeColumn((short)10);
    libro.getSheetAt(0).autoSizeColumn((short)11);
    libro.getSheetAt(0).autoSizeColumn((short)12);



    //HOJA2 ---> Análisis de Sensibilidad
    Sheet hoja2 = libro.createSheet("Análisis de sensibilidad");

    fila = hoja2.createRow(0);
```

```
celda = fila.createCell(0);
celda.setCellValue("La formulación se mantiene si los precios de los
    "+
"ingredientes varían dentro de los rangos siguientes:");

fila = hoja2.createRow(2);
celda = fila.createCell(1);
celda.setCellValue("Nombre");
celda = fila.createCell(2);
celda.setCellValue("Precio");
celda = fila.createCell(3);
celda.setCellValue("Intervalo de variación");



SWIGTYPE_p_double coef1;
SWIGTYPE_p_double coef2;
coef1 = GLPK.new_doubleArray(1);
coef2 = GLPK.new_doubleArray(1);

int numVar=GLPK.glp_get_num_cols(lp);
int numRes=GLPK.glp_get_num_rows(lp);


for(int k=1;k<=numVar;k++){
   System.out.println(GLPK.glp_get_col_stat(lp,k)+"STAT"); //////
       %%%%%% !!!!!!
   if(GLPK.glp_get_col_stat(lp,k)==GLPK.GLP_BS){
      GLPK.glp_analyze_coef(lp, k+numRes, coef1, null,
          null,coef2,null,null);
      fila=hoja2.createRow(k+2);
      celda = fila.createCell(1);
      celda.setCellValue(GLPK.glp_get_col_name(lp,k));
      celda = fila.createCell(2);
      celda.setCellValue(String.format("%1$10.3f
          ",GLPK.glp_get_obj_coef(lp, k)));
      celda = fila.createCell(3);
      celda.setCellValue(String.format("%1$10.3f
          ",GLPK.doubleArray_getitem(coef1, 0)));
      celda = fila.createCell(4);
      celda.setCellValue(String.format("%1$10.3f
          ",GLPK.doubleArray_getitem(coef2, 0)));
   }
   else{
      if(GLPK.glp_get_col_stat(lp,k)==GLPK.GLP_NL){

         fila=hoja2.createRow(k+2);
         celda = fila.createCell(1);
         celda.setCellValue(GLPK.glp_get_col_name(lp,k));
         celda = fila.createCell(2);
         celda.setCellValue(String.format("%1$10.3f
             ",GLPK.glp_get_obj_coef(lp, k)));
         celda = fila.createCell(3);
         celda.setCellValue(String.format("%1$10.3f
             ",GLPK.glp_get_obj_coef(lp,k)-GLPK.glp_get_col_dual(lp,
             k)));
         celda = fila.createCell(4);
         celda.setCellValue(String.format("%1$10s","Inf"));
```

```
        }
        else { if (GLPK.glp_get_col_stat(lp,k)==GLPK.GLP_NU)

           fila=hoja2.createRow(k+2);
           celda = fila.createCell(1);
           celda.setCellValue(GLPK.glp_get_col_name(lp,k));
           celda = fila.createCell(2);
           celda.setCellValue(GLPK.glp_get_obj_coef(lp, k));
           celda = fila.createCell(3);
           celda.setCellValue(" -Inf ");
           celda = fila.createCell(4);
           celda.setCellValue(String.format("%1$8.3f
               ",GLPK.glp_get_obj_coef(lp,k)-GLPK.glp_get_col_dual(lp,
               k)));
        }
     }
  }
  libro.getSheetAt(1).autoSizeColumn((short)1);
  libro.getSheetAt(1).autoSizeColumn((short)2);



  fila = hoja2.getRow(0);
  celda = fila.createCell(8);
  celda.setCellValue("Valoración de nutrientes:");
  fila = hoja2.getRow(2);
  celda = fila.createCell(8);
  celda.setCellValue("Nombre");

  fila = hoja2.getRow(3);
  for(int k=1;k<numRes;k++){
     if(GLPK.glp_get_row_stat(lp, k+1)==GLPK.GLP_NL){
        celda = fila.createCell(8);
        celda.setCellValue(GLPK.glp_get_row_name(lp,k+1));
        celda=fila.createCell(9);
        celda.setCellValue("contenido");
        celda=fila.createCell(10);
        celda.setCellValue(String.format("%10.3f",GLPK.glp_get_row_prim(lp,
           k+1)));
        celda=fila.createCell(11);
        celda.setCellValue("unidad adicional de requerimiento mínimo
            incrementa el costo en");
        celda=fila.createCell(12);
        celda.setCellValue(String.format("%10.4f?",GLPK.glp_get_row_dual(lp,
            k+1)));
        fila=hoja2.getRow(fila.getRowNum()+1);
     }

     else if(GLPK.glp_get_row_stat(lp, k+1)==GLPK.GLP_NU){
        celda = fila.createCell(8);
        celda.setCellValue(GLPK.glp_get_row_name(lp,k+1));
        celda=fila.createCell(9);
        celda.setCellValue("contenido");
        celda=fila.createCell(10);
        celda.setCellValue(String.format("%10.3f",GLPK.glp_get_row_prim(lp,
           k+1)));
        celda=fila.createCell(11);
```

```
            celda.setCellValue("unidad menos de requerimiento máximo
                incrementa el costo en");
            celda=fila.createCell(12);
            celda.setCellValue(String.format("%10.4f?",GLPK.glp_get_row_dual(lp,
                k+1)));
            fila=hoja2.getRow(fila.getRowNum()+1);
        }
    }


    libro.getSheetAt(1).autoSizeColumn((short)8);
    libro.getSheetAt(1).autoSizeColumn((short)11);



    //HOJA 3 ---> Información de ingredientes y nutrientes
    Sheet hoja3 = libro.createSheet("Distribución de nutrientes");
    fila = hoja3.createRow(0);
    celda = fila.createCell(0);
    celda.setCellValue("Ingredientes seleccionados: ");
    fila = hoja3.createRow(1);
    celda = fila.createCell(1);
    celda.setCellValue("Nombre");
    celda = fila.createCell(2);
    celda.setCellValue("Mínimo");
    celda = fila.createCell(3);
    celda.setCellValue("Máximo");
    celda = fila.createCell(4);
    celda.setCellValue("Precio");
    fila = hoja3.getRow(0);
    celda = fila.createCell(8);
    celda.setCellValue("Nutrientes seleccionados: ");
    fila = hoja3.getRow(1);
    celda = fila.createCell(9);
    celda.setCellValue("Nombre");
    celda = fila.createCell(10);
    celda.setCellValue("Mínimo");
    celda = fila.createCell(11);
    celda.setCellValue("Máximo");


    for (int k=0;k<GLPK.glp_get_num_cols(lp);k++){
        fila = hoja3.createRow(k+2);
        celda = fila.createCell(1);
        celda.setCellValue(GLPK.glp_get_col_name(lp,k+1));
        celda = fila.createCell(2);
        celda.setCellValue(String.format("%1$10.3f
            ",100*GLPK.glp_get_col_lb(lp, k+1)));
        celda = fila.createCell(3);
        celda.setCellValue(String.format("%1$10.3f
            ",100*GLPK.glp_get_col_ub(lp, k+1)));
        celda = fila.createCell(4);
        celda.setCellValue(String.format("%1$10.3f
            ",GLPK.glp_get_obj_coef(lp, k+1)));
    }
```

```
for(int j=0; j<GLPK.glp_get_num_rows(lp);j++){
   fila = hoja3.getRow(j+2);
   celda = fila.createCell(9);
   celda.setCellValue(GLPK.glp_get_row_name(lp,j+1));
   celda = fila.createCell(10);
   celda.setCellValue(String.format("%1$10.3f
       ",GLPK.glp_get_row_lb(lp, j+1)));
   celda = fila.createCell(11);
   celda.setCellValue(String.format("%1$10.3f
       ",GLPK.glp_get_row_ub(lp, j+1)));
}
libro.getSheetAt(2).autoSizeColumn((short) 1); //ajusta el ancho de
    la celda para que se vea completo el nombre del ingrediente
libro.getSheetAt(2).autoSizeColumn((short)9); //ajusta el ancho de
    la celda para que se vea completo el nombre del nutriente



// Se añaden en fila los nutrientes utilizados en la composición
fila = hoja3.createRow(GLPK.glp_get_num_cols(lp)+6);
celda = fila.createCell(0);
celda.setCellValue("Contenidos de nutrientes en cada ingrediente: ");

fila = hoja3.createRow(GLPK.glp_get_num_cols(lp)+7);
for(int j=1; j<GLPK.glp_get_num_rows(lp);j++){
   celda=fila.createCell(j+1);
   celda.setCellValue(GLPK.glp_get_row_name(lp, j+1));
   libro.getSheetAt(2).autoSizeColumn((short) j+1); //ajusta el ancho
       de la celda para que se vea completo el nombre del nutriente
}



// Se añaden en columna los ingredientes utilizados en la composición
for(int k=0;k<GLPK.glp_get_num_cols(lp);k++){
   fila = hoja3.createRow(k+GLPK.glp_get_num_cols(lp)+8);
   celda=fila.createCell(1);
   celda.setCellValue(GLPK.glp_get_col_name(lp,k+1));
   fila = hoja3.getRow(k+GLPK.glp_get_num_cols(lp)+8);
   for(int j=1;j<GLPK.glp_get_num_rows(lp);j++){
      celda=fila.createCell(j+1);
      celda.setCellValue(A[j-1][k]);
   }
}


int ingr[] = new int[ingredientes.getRowCount()];
int nutr[] = new int[nutrientes.getRowCount()];
//
for (int k=0; k< ingredientes.getRowCount(); k++){
   String temp= (String)(ingredientes.getValueAt(k,0));
   ingr[k] = Integer.parseInt(temp.split(":")[0]);
}
//
for (int j=0; j< nutrientes.getRowCount(); j++){
   String temp= (String)(nutrientes.getValueAt(j,0));
   nutr[j] = Integer.parseInt(temp.split(":")[0]);
}
```

```
                libro.write(archivo);
                libro.getCreationHelper().createFormulaEvaluator().evaluateAll();
                archivo.close();
             }


             int confirmar=JOptionPane.showConfirmDialog(null, "¿Quiere abrir el
                 fichero Excel creado? ");
             if(JOptionPane.OK_OPTION==confirmar) {
                Desktop.getDesktop().open(ficheroExcel);
             }

          }
       } catch(NullPointerException e) {

    }


   } catch (IOException ioe) {
   ioe.printStackTrace();
}

}
```

C.8. Clase AcercaDe

```
package programadietas;

/**
*
* @author aaguilar, pmateo
*/
public class AcercaDe extends javax.swing.JDialog {

   /**
   * Creates new form NewJDialog
   */
   public AcercaDe(java.awt.Frame parent, boolean modal) {
      super(parent, modal);
      initComponents();
   }

   /**
   * This method is called from within the constructor to initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is always
   * regenerated by the Form Editor.
   */
   @SuppressWarnings("unchecked")
   // <editor-fold defaultstate="collapsed" desc="Generated Code">
   private void initComponents() {
```

```
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Acerca de CompFeed");
        setFont(new java.awt.Font("Arial", 0, 8)); // NOI18N
        setResizable(false);

        jScrollPane1.setHorizontalScrollBarPolicy(javax.swing.ScrollPaneConstants.
                HORIZONTAL_SCROLLBAR_NEVER);

        jTextArea1.setEditable(false);
        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Arial", 0, 10)); // NOI18N
        jTextArea1.setRows(5);
        jTextArea1.setText("\n\nCompFeed es un programa creado para el\n Trabajo Fin de
            Grado en Matemáticas:\n\n \"Desarrollo de un Programa Informático\n para la
            Formulación de Piensos Compuestos\"\n\n Realizado por: \n Andrea Aguilar
            Ibáñez\n\n Dirigido por: \n Pedro Mateo Collazos\n Joaquín Surra Muñoz\n\n
            Curso 2015/2016.\n\n Facultad de Ciencias, \n Universidad de
            Zaragoza.\n\t\n\n");
        jScrollPane1.setViewportView(jTextArea1);

        getContentPane().add(jScrollPane1, java.awt.BorderLayout.CENTER);

        jLabel1.setIcon(new
            javax.swing.ImageIcon(getClass().getResource("/programadietas/cabra.jpg")));
            // NOI18N
        getContentPane().add(jLabel1, java.awt.BorderLayout.PAGE_START);

        pack();
    }// </editor-fold>

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
            (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
            look and feel.
        * For details see
            http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(AcercaDe.class.getName()).log(java.util.logging.
            Level.SEVERE, null, ex);
```

```
   } catch (InstantiationException ex) {
   java.util.logging.Logger.getLogger(AcercaDe.class.getName()).log(java.util.logging.
           Level.SEVERE, null, ex);
} catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(AcercaDe.class.getName()).log(java.util.logging.Level.
           SEVERE, null, ex);
} catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(AcercaDe.class.getName()).log(java.util.logging.Level.
           SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the dialog */
java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
      AcercaDe dialog = new AcercaDe(new javax.swing.JFrame(), true);
      dialog.addWindowListener(new java.awt.event.WindowAdapter() {
         @Override
         public void windowClosing(java.awt.event.WindowEvent e) {
            System.exit(0);
         }
      });
      dialog.setVisible(true);
   }
});
}

// Variables declaration - do not modify
private javax.swing.JLabel jLabel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
// End of variables declaration
}
```