INSTITUTO SUPERIOR MINERO METALURGICO

“Dr. Antonio Núñez Jiménez”.

Facultad de Metalurgia - Eletromecânica

Moa, Holguín

TRABAJO DE DIPLOMA

“Sistema informático para la solución de problemas de programación lineal”.

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor: Yoel Orlando Perdomo Fernández.

Tutor: Ing. Isidro Manuel Corría Ramírez.

Consultantes: Ing. Yadira Romero Rodríguez.

Ing. Roiki Rodríguez Noa.

Moa, Cuba

Junio, 2008

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro INSTITUTO SUPERIOR MINERO METALURGICO “Dr. Antonio Núñez Jiménez” para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los _____ días del mes de _____ del 2008.

Yoel Orlando Perdomo Fernández.
Nombre completo del primer autor

Isidro Manuel Corría Ramírez
Nombre completo del primer tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado “Sistema informático para la solución de problemas de programación lineal”, fue realizado en nuestra entidad Instituto Superior Minero Metalúrgico de Moa. Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

_____

_____

_____

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reporta un efecto económico que asciende a _____ MN y _____ CUC.

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año 2008

_____ _____

Nombre del representante de la entidad Cargo

_____

Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA.

Título: Sistema informático para la solución de problemas de programación lineal.

Autor: Yoel Orlando Perdomo Fernández.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____

Nombre completo del primer tutor

Fecha: _____

Agradecimientos

A mi familia, y muy especialmente a mi madre, y padre, por tanto amor y confianza que han depositado en mi.

A mis amigos y compañeros de aula: en especial a Ariel, Paucides, Osmani, Tony, Yodexy, Maurice, Alexei, por darme su apoyo incondicional.

A todos aquellos que de una forma u otra han contribuido con la realización de este trabajo y en mi formación como futuro profesional.

A mi tutor, Ing. Isidro Manuel Corría Ramírez, a las profesoras Virgen Cuza Noa y Yadira Romero Rodríguez por su ayuda y ser guía en todo momento, a mis compañeros de grupo y profesores del departamento.

A todos, gracias…

Yoel Orlando Perdomo Fernández.

Dedicatoria

Dedico este trabajo a todas aquellas personas portadoras de los más nobles sentimientos: el amor, la amistad, la confianza, que me mostraron siempre el camino correcto por el cual he de transitar.

En especial:
A mis padres, símbolos de inspiración de todos mis proyectos.

A mis tíos, faro de mis pasos, y oportuno consejo.

A mis hermanos, por inspirarme a ser un ejemplo y guía para ellos.

A mi novia, por alentarme y estar a mi lado todo el tiempo.

A mis suegros, por acogerme como un hijo más en la familia.

En general a todas aquellas personas que no menciono, no por ser menos importantes sino porque no alcanzarían estas pocas líneas para mencionarlos.

Resumen

El advenimiento y desarrollo de las Nuevas Tecnologías de la Información y las Comunicaciones, han contribuido de forma sustancial al incremento del caudal de información. Por lo que cada vez se impone la necesidad de manipular grandes volúmenes de información para facilitar la toma de decisiones.

Con el desarrollo de esta investigación se propone implementar un subsistema informático que solucione problemas de programación lineal, con los métodos más referenciados en las bibliografías, que facilite su integración a un sistema general que responda a las necesidades de la investigación de operaciones.

Para la realización de la investigación se realizó una revisión bibliográfica sobre aplicaciones y herramientas para su construcción. En el presente documento se recoge un resumen del estudio bibliográfico realizado, se presenta la metodología de Ingeniería de Software que se siguió para la construcción del software la cuál se propone como solución de la problemática encontrada. En el trabajo se realiza además un estudio de factibilidad y una valoración de sostenibilidad del producto informático que se obtiene como resultado de la investigación.

Índice

INTRODUCCIÓN **1**

CAPÍTULO 1 FUNDAMENTACIÓN. **9**

1.1 INTRODUCCIÓN 9
1.2 ESTADO DEL ARTE. 9
1.2.1 Conceptos Fundamentales *9*
1.3 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN 21
1.4 TENDENCIAS Y TECNOLOGÍAS ACTUALES 21
1.4.1 Lenguajes de programación para el desarrollo de aplicaciones. *21*
1.4.2 Fundamentación de la selección de lenguaje a utilizar. *27*
1.4.3 Metodologías para el desarrollo de Sistemas Informáticos. *28*
1.5 CONCLUSIONES 34

CAPÍTULO 2 MODELO DEL DOMINIO **35**

2.1 INTRODUCCIÓN 35
2.2 DEFINICIÓN DE LAS ENTIDADES Y LOS CONCEPTOS PRINCIPALES 35
2.3 REPRESENTACIÓN DEL MODELO DEL DOMINIO 37
2.4 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES DEL SISTEMA. 38
2.4.1 Requisitos Funcionales *38*
2.4.2 Requisitos no Funcionales *38*
2.5 CONCLUSIONES. 39

CAPÍTULO 3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA **40**

3.1 INTRODUCCIÓN 40
3.2 ACTORES DEL SISTEMA A AUTOMATIZAR 40
3.3 DIAGRAMA DE CASOS DE USO DEL SISTEMA A AUTOMATIZAR 40
3.4 DESCRIPCIÓN DE LOS CASOS DE USO 41
3.5 DIAGRAMA DE CLASES DEL DISEÑO 47
3.5.1 Diagrama de Clases General *47*
3.5.2 Descripción General de las Clases. *48*
3.6 PRINCIPIOS DE DISEÑO 56
3.6.1 Interfaz de usuário *56*
3.6.2 Formato de salida de los reportes *61*
3.7 DISEÑO DE LA BASE DE DATOS 63
3.7.1 Modelo lógico de datos *63*
3.8 DIAGRAMAS DE SECUENCIA 64
3.9 DIAGRAMA DE DESPLIEGUE 69
3.10 DIAGRAMA DE COMPONENTES 69
3.11 CONCLUSIONES 70

CAPÍTULO 4 ESTUDIO DE FACTIBILIDAD **71**

4.1 INTRODUCCIÓN. 71
4.2 EFECTOS ECONÓMICOS 71
4.3 BENEFICIOS Y COSTOS INTANGIBLES EN EL PROYECTO 74
4.4 FICHA DE COSTO 74
4.5 CONCLUSIONES 77

CONCLUSIONES **78**

RECOMENDACIONES ..... **I**

REFERENCIAS BIBLIOGRÁFICAS ..... **II**

BIBLIOGRAFIA ..... **III**

GLOSARIO DE TÉRMINOS IV

Indice de tablas

TABLA 1 ENTIDADES Y CONCEPTOS DEL DOMINIO. 35
TABLA 2. DEFINICIÓN DE ACTORES DEL SISTEMA A AUTOMATIZAR 40
TABLA 3.DESCRIPCIÓN DEL CASO DE USO "INSERTAR MODELO" 41
TABLA 4. DESCRIPCIÓN DEL CASO DE USO "GUARDAR MODELO" 41
TABLA 5. DESCRIPCIÓN DEL CASO DE USO "EDITAR MODELO" 42
TABLA 6. DESCRIPCIÓN DEL CASO DE USO "CARGAR MODELO" 42
TABLA 7. DESCRIPCIÓN DEL CASO DE USO "SOLUCIONAR MODELO PRIMAL" 43
TABLA 8. DESCRIPCIÓN DEL CASO DE USO "CALCULAR MODELO DUAL" 43
TABLA 9. DESCRIPCIÓN DEL CASO DE USO "SOLUCIONAR MODELO DUAL" 44
TABLA 10. DESCRIPCIÓN DEL CASO DE USO "REPORTE POSTÓPTIMO" 44
TABLA 11. DESCRIPCIÓN DEL CASO DE USO "SENSIBILIDAD SI SE AGREGA UNA NUEVA VARIABLE" ... 45
TABLA 12. DESCRIPCIÓN DEL CASO DE USO "SENSIBILIDAD SI SE AGREGA UNA NUEVA RESTRICCIÓN"45
TABLA 13. DESCRIPCIÓN DEL CASO DE USO "INTERPRETACIÓN ECONÓMICA" 46
TABLA 14 DESCRIPCIÓN DE LA CLASE "PROG_LINEAL". 48
TABLA 15. DESCRIPCIÓN DE LA CLASE "FICHERO". 50
TABLA 16. DESCRIPCIÓN DE LA CLASE "MODELO". 50
TABLA 17. DESCRIPCIÓN DE LA CLASE "MODELOORIG". 51
TABLA 18. DESCRIPCIÓN DE LA CLASE "MODELOREV". 51
TABLA 19. DESCRIPCIÓN DE LA CLASE "DSIP_RECURSOS". 52
TABLA 20. DESCRIPCIÓN DE LA CLASE "VARIABLES". 52
TABLA 21. DESCRIPCIÓN DE LA CLASE "SIMPLEX". 52
TABLA 22. DESCRIPCIÓN DE LA CLASE "SIMPLEX_ORIGINAL". 53
TABLA 23. DESCRIPCIÓN DE LA CLASE "SIMPLEX_REVISADO". 55

Índice de Figuras

FIGURA 1 DIAGRAMA DEL MODELO DE DOMINIO. 37
FIGURA 2 DIAGRAMA DE CASO DE USO DEL SISTEMA. 40
FIGURA 3 DIAGRAMA DE CLASE DEL SISTEMA. 47
FIGURA 4 INTERFAZ DE USUARIO "INSERTAR MODELO 1". 56
FIGURA 5 INTERFAZ DE USUARIO "INSERTAR MODELO 2". 57
FIGURA 6 INTERFAZ DE USUARIO "INSERTAR MODELO 3". 57
FIGURA 7 INTERFAZ DE USUARIO "INSERTAR MODELO 4". 58
FIGURA 8 INTERFAZ DE USUARIO "SOLUCIONAR MODELO". 58
FIGURA 9 INTERFAZ DE USUARIO "AGREGAR NUEVA VARIABLE". 59
FIGURA 10 INTERFAZ DE USUARIO "AGREGAR NUEVA RESTRICCIÓN". 59
FIGURA 11 INTERFAZ DE USUARIO "CARGAR MODELO". 60
FIGURA 12 INTERFAZ DE USUARIO "GUARDAR MODELO". 60
FIGURA 13 INTERFAZ DE USUARIO "SOLUCIÓN DE MODELO". 61
FIGURA 14 INTERFAZ DE USUARIO "SOLUCIÓN DE MODELO 1". 61
FIGURA 15 INTERFAZ DE USUARIO "REPORTE POSTÓPTIMO". 62
FIGURA 16 INTERFAZ DE USUARIO "INTERPRETACIÓN ECONÓMICA". 62
FIGURA 17 DIAGRAMA DE CLASES PERSISTENTE 63
FIGURA 18 DIAGRAMA DE SECUENCIA "CALCULAR DUAL". 64
FIGURA 19 DIAGRAMA DE SECUENCIA "CARGAR MODELO / GUARDAR MODELO". 64
FIGURA 20 DIAGRAMA DE SECUENCIA "INSERTAR MODELO" 65
FIGURA 21 DIAGRAMA DE SECUENCIA "ANÁLISIS DE SENSIBILIDAD SI SE AGREGA UNA NUEVA VARIABLE" 65
FIGURA 22 DIAGRAMA DE SECUENCIA "REPORTE POSTÓPTIMO". 66
FIGURA 23 DIAGRAMA DE SECUENCIA "INTERPRETACIÓN ECONÓMICA". 66
FIGURA 24 DIAGRAMA DE SECUENCIA "SOLUCIONAR MODELO". 67
FIGURA 25 DIAGRAMA DE SECUENCIA "EDITAR MODELO". 68
FIGURA 26 DIAGRAMA DE SECUENCIA "ANÁLISIS DE SENSIBILIDAD SI SE AGREGA UNA NUEVA RESTRICCIÓN". 68
FIGURA 27 DIAGRAMA DE DESPLIEGUE 69
FIGURA 28 DIAGRAMA DE COMPONENTE. 69
FIGURA 29 PUNTO DE EQUILIBRIO. 77

Introducción

Desde los orígenes de la humanidad el hombre en su afán por la supervivencia se ha visto inmerso en la necesidad de recurrir a la toma de decisiones con el fin de adaptarse a las diferentes condiciones en que se ha desarrollado.

La situación económica y social que caracteriza a la sociedad moderna genera profundos cambios en las organizaciones, las cuales se preparan para ser más flexibles y establecen estrategias con el objetivo de adaptarse al entorno altamente turbulento en el que desarrollan sus acciones. Ante ambientes tan poco estables y la imposibilidad de actuar a ciegas, los miembros de las organizaciones y, en particular, sus altas gerencias necesitan manipular grandes volúmenes de información para cumplir con sus funciones esenciales. Deben implementarse entonces, prácticas administrativas dirigidas a garantizar el éxito organizacional y entre ellas, la toma de decisiones, soportada en el análisis de información. Para lo que ha venido desarrollando mecanismos que avalen o justifiquen dichas decisiones.

Con respecto al concepto "toma de decisiones", Schein, plantea: "la toma de decisiones es el proceso de identificación de un problema u oportunidad y la selección de una alternativa de acción entre varias existentes, es una actividad diligente clave en todo tipo de organización."[1].

La teoría de decisiones plantea que cuando se está en la tarea de seleccionar la mejor alternativa se caerá en una, de cuatro categorías generales, dependiendo de las habilidades para predecir la consecuencia de cada alternativa; derivándose de estas diferentes formas de toma de decisión las siguientes:

Bajo certidumbre: si se puede predecir con certeza las consecuencias de cada alternativa.

Bajo riesgo: esta categoría incluye aquellas decisiones para las que las consecuencias de una acción dada dependen de algún evento probabilista.

Bajo incertidumbre: es parecido a la toma de decisiones bajo riesgos con la diferencia de que no se tiene conocimientos de las probabilidades de los eventos futuros, es decir, de cuán posible sean las diferentes consecuencias.

Bajo conflicto: aquí se tiene aquellos casos de toma de decisiones bajo incertidumbre en los que hay un oponente, las probabilidades de los eventos no solo se conocen; están influenciadas por un oponente cuya meta es vencer.

La segunda mitad del siglo XX fue testigo de algunos hechos concretos que propiciaron grandes cambios, entre ellos: la aparición de las computadoras, el crecimiento acelerado de su velocidad de procesamiento y capacidad para el almacenamiento de datos, además de la facilidad de interconexión, sin dudas, ha posibilitado disponer de servicios de acceso en línea a bases de datos y ha provocado una gran explosión de información que rebasa la capacidad de procesamiento de las organizaciones y la búsqueda de herramientas para el manejo de estos grandes volúmenes de información.

Todo esto, unido al incremento de la competencia a nivel mundial, impulsado por el dominio de las trasnacionales, ha originado desde los años ochenta, un contexto de creciente necesidad de dotarse de técnicas de captación y análisis de información sobre el entorno competitivo y tecnológico y, en particular, de formas organizativas y herramientas que faciliten dicho objetivo.

La adopción de posiciones de carácter preactivo y su práctica implica actualmente el análisis de las características, tanto del entorno como de las condiciones internas de la organización con el propósito de reducir la

incertidumbre en la toma de decisiones y realizar una planeación con un mayor grado de certeza.

Uno de los factores más importantes en el desarrollo del mundo contemporáneo son los conocimientos que posea una organización, cualquiera que esta sea y claro está, como sea capaz de aplicarlos.

En este empeño, comprobada está, la marcada importancia del aprovechamiento del imponente caudal de recursos de información existentes para la toma de decisiones en las instituciones. El sólido crecimiento y desarrollo de cualquier economía, desde la de una pequeña organización hasta la de un país entero, dependerá indudablemente de decisiones basadas en el conocimiento.

El advenimiento de las Nuevas Tecnologías de la Información y las Comunicaciones (NTIC), así como el surgimiento y auge de la llamada industria de la información ha contribuido de forma sustancial al incremento del caudal de información. Su principal exponente, Internet, ha favorecido a que la generación de conocimientos, se acelere cada vez más y lleve a las organizaciones por caminos insospechados.

Apoyándonos en lo antes mencionado y en el siguiente pensamiento: "El acceso rápido y eficiente a una información confiable y precisa permite adoptar una posición adecuada a la hora de tomar una decisión para solucionar un problema con un menor costo. Pero esto sólo es posible, si se ha realizado previamente un proceso de análisis de la información, en el que se adicione un conjunto de valores pertinentes a partir del trabajo intenso que realizan especialistas entrenados en el uso de las técnicas de información". [2]

Podemos afirmar que la utilización de sistemas informáticos de análisis de datos ha supuesto una revolución en la toma de decisiones por varias razones:

1. Se simplifica enormemente los procedimientos matemáticos y como consecuencia se emplean técnicas modernas que antes eran muy difícil aplicarlas.

2. Se pasa del cálculo manual tradicional al cálculo por ordenador lo que supone una mayor rapidez en la obtención de los resultados, mayor fiabilidad, mejor manejo de estos procedimientos y una gran facilidad de aplicación.

3. Propicia una nueva forma de enseñanza del análisis de datos sin necesidad de una gran formación matemática para efectuar operaciones que en ocasiones son muy complejas.

A pesar de las favorables condiciones que nos brindan las nuevas tecnologías existen elementos que dificultan un mejor desempeño del profesional a la hora de la toma de decisiones, como es la integración de las herramientas informáticas.

En la actualidad existen software que facilitan el proceso de la toma de decisiones como el Matemática, Matlab, Excel, entre otros, pero a medida que estos se desarrollan necesitan ordenadores con condiciones más exigentes, encontrándonos en ocasiones en la necesidad de utilizar más de uno de estos para la interpretación de un modelo, no contando con la integridad de todos los conocimientos para el eficiente análisis e interpretación de los resultados, y como otro inconveniente en ocasiones prescindimos de las supercomputadoras que demandan estos programas debido a su altos costos en el mercado mundial, por lo que esto nos conlleva a plantearnos el **problema** siguiente:

La insuficiente integración de las herramientas informáticas relacionadas con la investigación de operaciones disponibles dificulta la formación de los modos de actuación profesional en la toma de decisiones.

Por lo antes referido tomamos como **Objeto de Investigación:** La metodología de investigación de operaciones para la toma de decisiones.

Investigación Operativa o Investigación de Operaciones es una disciplina moderna que mediante el uso de modelos matemáticos, estadística y algoritmos modela y resuelve problemas complejos determinando la solución óptima y permitiendo, de esta forma, la toma de decisiones. Actualmente incluye gran cantidad de ramas como la Programación Lineal, Programación No Lineal, Programación Dinámica, Simulación, Teoría de Colas, Teoría de Inventarios, Teoría de Grafos, etc.

Aunque su nacimiento como ciencia se establece durante la Segunda Guerra Mundial y debe su nombre a las operaciones militares, los verdaderos orígenes de la Investigación Operativa se remontan mucho más atrás en el tiempo, hasta el siglo XVII. Sin embargo su auge es debido, en su mayor parte, al gran desarrollo de la informática, gracias a la cual es posible resolver problemas en la práctica y obtener soluciones que de otra forma conllevarían un enorme tiempo de cálculo. Debido al gran éxito de la Investigación Operativa en el campo militar, esta se extendió a otros campos tales como la industria, física, informática, economía, estadística y probabilidad, ecología, educación, servicio social, etc. Siendo hoy en día utilizada prácticamente en todas las áreas.

Desde el punto de vista matemático, en los siglos XVII y XVIII, Newton, Leibnitz, Bernoulli y Lagrange, trabajaron en obtener máximos y mínimos condicionados de ciertas funciones. El matemático francés Jean Baptiste-Joseph Fourier esbozó métodos de la actual programación lineal. Y en los últimos años del siglo XVIII, Gaspar Monge asentó los precedentes del Método Gráfico gracias a su desarrollo de la Geometría Descriptiva.

Por lo que nuestro **campo de acción** en esta investigación se enmarca en: La Programación Lineal.

Para la solución del problema antes mencionado nos trazamos como **Objetivo**: Implementar un sistema informático con los métodos más referenciados por las bibliografías en la solución de problemas de programación lineal permitiendo empotrarse en un sistema integrador.

Para dar cumplimiento de este último nos hemos propuesto cumplir con los siguientes **Objetivos específicos:**

1 Realizar un estudio de los principales conceptos y teorías referidas al campo de acción.

2 Profundizar en el análisis del estado del arte de la investigación de operaciones.

3 Diseñar e implementar un sistema informático que solucione problemas de programación lineal en la investigación de operaciones.

Como **Hipótesis** partimos de la idea de que: Si se implementa un sistema informático con los métodos más referenciados por las bibliografías de programación lineal en la investigación de operaciones facilitará la integración de las herramientas informáticas para la toma de decisiones.

Para cumplir con nuestros objetivos y resolver la situación problémica planteada, se proponen las siguientes **tareas:**

1. Análisis de la literatura existente sobre la investigación de Operaciones.
2. Selección del lenguaje de programación para llevar a cabo el proyecto y fundamento de la elección.
3. Selección de la metodología de Análisis y Diseño de sistemas informáticos, que facilite la creación y garantice la calidad del sistema.
4. Realizar el análisis, diseño e implementación de la aplicación que facilite la solución de problemas de programación lineal.

Para cumplimentar estas tareas se han empleado **métodos teóricos** y **empíricos** de la investigación científica.

Entre los métodos empíricos usados podemos citar **la entrevista** y **el análisis de documentos** para la recopilación de la información. La entrevista posibilitó conocer más a fondo las necesidades de los profesionales y estudiantes en la solución de problemas de programación lineal además de permitirnos determinar los principales requerimientos del sistema. Mediante el análisis de documentos se supo como funcionan actualmente los procesos de solución de problemas de programación lineal.

Los métodos teóricos proporcionaran calidad en la investigación. En el desarrollo del proceso de investigación se usaron **el análisis y síntesis** para la recopilación y el procesamiento de la información obtenida en los métodos empíricos y arribar a las conclusiones de la investigación. El **hipotético deductivo** utilizó en la elaboración de la hipótesis y para su verificación. La **modelación** permitió realizar una representación de la realidad, se logró detectar problemas en la forma actual de procesar la información y encontrar las funcionalidades que debe de tener el sistema que se propone, que lo harán más completo y le brindarán satisfacción al usuario con un producto de mayor calidad.

El presente trabajo consta de introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía, anexos y glosario de términos:

En el Capítulo 1, Fundamentación, se ofrece una breve descripción del objeto de estudio, el flujo de los procesos en los que interviene el mismo y un análisis crítico al respecto. Se brinda además una panorámica de los sistemas automatizados existentes vinculados al campo de acción y las tendencias y tecnologías actuales así como un análisis crítico de las fuentes utilizadas.

En el Capítulo 2, Modelo del Dominio, se explica el bajo nivel de estructuración del negocio y la necesidad de utilizar un modelo de domino para mostrar la dinámica del sistema, se definen las entidades y conceptos

principales y las reglas de negocio a considerar así como la representación del modelo de dominio.

En el Capítulo 3, Diseño e Implementación del Sistema, se describen en detalles los flujos de trabajos relacionados a estas etapas de diseño e implementación de la metodología utilizada, RUP.

En el Capítulo 4, Estudio de Factibilidad, se presenta el estudio de factibilidad de este proyecto, se utilizara la Metodología Costo Efectividad (Beneficio), la cual plantea la conveniencia de la ejecución del proyecto.

Para concluir se muestran las Conclusiones a las que se arribaron, las Recomendaciones que se proponen, la Bibliografía utilizada, Anexos con información necesaria sobre el trabajo y el Glosario de Términos.

Capítulo 1 Fundamentación.

1.1 Introducción

En el presente capítulo se brinda una visión general de los aspectos relacionados con la solución de problemas de programación lineal. Se abordan las pautas específicas que constituyen los fundamentos teóricos sobre los que se apoya nuestra propuesta, además se realiza un análisis de las herramientas más comunes para el desarrollo de aplicaciones y se fundamenta la elección del lenguaje seleccionado, así como la metodología a utilizar.

Por otra parte se hace una valoración de las características de los software que han trabajado la temática, analizando las desventajas de los mismos, así como los beneficios de nuestro sistema informático.

1.2 Estado del Arte.

1.2.1 Conceptos Fundamentales

La investigación de operaciones como metodología de análisis para la toma de decisiones, es un valioso instrumento aplicable a cualquier problema de investigación económica, puesto que brinda al investigador o grupo de investigadores una determinada estructura, mediante la cual puede enmarcar su trabajo.

Para la solución de problemas de programación lineal se tiene en cuenta una metodología, que cuenta con las siguientes fases:

1. Formulación del problema.

   Esto es frecuentemente un problema secuencial. Se completa una reformulación inicial y continúa la investigación pero, al continuar la misma, el problema está sujeto a reformulación y refinamiento casi continuo y progresivo. Esto continúa hasta llegar a una solución.
2. Construcción del modelo matemático.

Un modelo matemático expresa la efectividad del sistema que se estudia como una función de un conjunto de variables, estando al menos una de ellas sujeta a control.

La fórmula general de un modelo de investigación operacional puede expresarse como E = F (x_i, y_i) donde E representa la efectividad del sistema, x_i aquellas variables de sistema que están sujetas a control, y y_i aquellas variables que no están sujetas a control.

En la construcción del modelo se deben seguir los siguientes pasos:

a) Definición de las variables del modelo. Debemos tener en cuenta que las mismas deben estar asociadas a actividades del problema de tal forma que a través de ellas se logren los objetivos planteados.

b) Construcción del sistema de restricciones. Está constituido por el conjunto de limitaciones que restringen el objetivo a lograr y está formado por un sistema de ecuaciones o inecuaciones lineales. Cada una de esta expresará una determinada limitación en recursos.

c) Planteamiento de la función objetivo. Es una función lineal que debe representar el objetivo que se quiere lograr en el problema. Viene dada de la forma:

Maximizar (Minimizar) $Z = \sum_{i=j}^{n} c_i x_i$

3. Derivar una solución del modelo.

Existen dos tipos de procedimientos para obtener una solución óptima de un modelo: el analítico y el numérico. Los procedimientos analíticos consisten en el uso de la deducción matemática, por medio de la aplicación de varias ramas de la matemática tales como el cálculo, el álgebra de matrices, etcétera.

Los procedimientos numéricos consisten en probar varios valores de las variables controlables en el modelo, comprobando los resultados obtenidos y seleccionando aquel conjunto de valores de las variables controladas que produzcan la mejor solución.

4. Probar el modelo y la solución que emane de él.

   Un modelo nunca es más que una representación parcial del sistema bajo estudio. Es un buen modelo si a pesar de ser incompleto, puede predecir el efecto de cambios en el sistema sobre la efectividad global del mismo.

   La solución puede ser evaluada comprobando los resultados obtenidos si aplicar la solución, con los resultados obtenidos cuando se utiliza. Esta evaluación puede ser ejecutada retrospectivamente por el uso de información pasada o por un ensayo de su aplicación o prueba preliminar.

5. Establecer controles sobre la solución.

   Una solución obtenida de un modelo sigue una solución sólo mientras las variables no controladas retienen sus valores y la relación entre las variables del modelo permanecen constantes.

   La solución se va fuera de control cuando el valor de una o más variables no controladas y/o una o más de las relaciones entre las variables han cambiado significativamente.

6. Poner en ejecución la solución.

   La solución probada debe trasladarse a un conjunto de procedimientos operativos capaces de ser comprendidos y aplicados por el personal que será responsable de su uso. Los cambios requeridos en los procedimientos existentes y las excepciones deben ser especificados y llevados a cabo.

Apoyándonos en lo antes referido, a continuación se realizará un estudio más detallado de los procesos que serán objeto de automatizar en los cuales encontramos: Obtención del modelo dual a partir del primal, la solución de

modelos de programación lineal (Primal, Dual), análisis de sensibilidad o postóptimo.

1. Obtención del modelo dual a partir del primal.

Modelo Primal: es el modelo obtenido del problema original. Para la obtención del modelo dual se siguen los siguientes pasos.

Dado el siguiente problema de programación lineal:

	Actividades		
Tipo de recurso	A	B	Cantidad de recursos disponibles
R_1	1	1	3
R_2	4	2	8
Contribución	3.5	2.5	

Función objetivo Primal (Maximizar, Minimizar):

Maximizar $Z = 3x + 8y$

Restricciones Primal

$$x + 2y \geq 2.5$$

$$x + 4y \geq 3.5$$

$$x \geq 0 , y \geq 0$$

1. Todas las inecuaciones deben corresponder en sentido con el criterio de la función objetivo. En el problema se multiplica la inecuación por (-1) si:

Objetivo	**Signos**
Maximizar	≥
Minimizar	≤

En nuestro caso multiplicaremos la ecuación 1 y 2 por (-1) por tanto.

$$- x -2 y \leq -2.5$$

$$- x - 4y \leq -3.5$$

2. Definimos las variables duales de acuerdo con el número de ecuaciones del problema primitivo. En nuestro caso.

$$x_1, y_1$$

3. Escribimos la función objetivo del dual multiplicando las variables duales por los términos independientes y con el criterio opuesto a la función objetivo del primitivo.

$$\text{Minimizar } Z = -2.5x_1 - 3.5y_1$$

4. Trasponemos la matriz del primitivo, asociándoles las correspondientes variables duales, colocando los términos independientes del dual tomándolos de los coeficientes de la función objetivo del primitivo y escribiendo los signos de las restricciones del dual teniendo en cuenta lo siguiente:

x_j no esta restringido $\sum_j a_{ij}y_i = c_j$ $j \ \{1,\ldots,n\}$

$x_j \geq 0$ $\sum_j a_{ij}y_i \leq c_j$ $j \ \{1,\ldots,n\}$

$$- x_1 – y_1 \leq 3$$

$$-2x_1 - 4y \leq 8$$

5. Restringimos las variables duales, teniendo en cuenta el sentido de las restricciones del primitivo. Es $\sum_j a_{ij}x_i = b_j$ decir y_j no esta restringido $j \{1: n\}$

$$\sum_j a_{ij}x_i \geq b_j \quad y_j \geq 0 \quad j \{1: n\}$$

De manera que

$$x_1, y_1 \geq 0$$

Así nuestro problema dual quedaría

Función objetivo Dual:

Minimizar $Z = -2.5x_1 - 3.5y_1$

Restricciones Dual.

$$- x_1 – y_1 \leq 3$$

$$-2x_1 - 4y \leq 8$$

$$x_1, y_1 \geq 0$$

2. Solución de los modelos ya sea primal o dual.

Una vez obtenido el modelo que consta de:

a) Variables de decisión.

X1: Nombre de la variable, descripción.

X2: Nombre de la variable, descripción

a) Función objetivo (Maximizar, Minimizar):

Maximizar $Z = 3 x_1 + 2 x_2$

b) Restricciones

$$2x_1 + x_2 \leq 18$$

$$2\,x_1 + 3\,x_2 \leq 42$$

$$3\,x_1 + x_2 \leq 24$$

c) Condición de no negatividad :

$$x_1 \geq 0\ ,\ x_2 \geq 0.$$

Para la solución del modelo se realizará la automatización de dos métodos el Simplex Original, y el Simplex Revisado. Se consideran las siguientes fases para el método simplex original:

1. Convertir las desigualdades en igualdades (estandarización) y lograr en la matriz asociada al sistema de restricciones una matriz (B) denominada matriz básica la cual debe tener la forma de la matriz idéntica (I), lo cual se logra introduciendo variables de holguras y artificiales.

Para convertir las desigualdades en igualdades podemos guiarnos por la siguiente tabla

Tipo de desigualdad	Tipo de variable que aparece
≥	- exceso + artificial
=	+ artificial
≤	+ holgura

En este caso se introdujo una variable de holgura por cada una de las restricciones del tipo ≤, para convertirlas en igualdades, resultando el sistema de ecuaciones lineales:

$$2x_1 + x_2 + x_3 = 18$$

$$2x_1 + 3x_2 + x_4 = 42$$

$$3x_1 + x_2 + x_5 = 24$$

2. Igualar la función objetivo a cero

$$-3x - 2y + Z = 0$$

3. Escribir la tabla inicial simplex

En las columnas aparecerán todas las variables de decisión del problema incluyendo las variables de holgura/exceso y artificiales. En las filas se observan, para cada restricción las variables que conforman la base con sus coeficientes de costo asociado, y la formula Zj-Cj:

Donde $Zj = \sum Cbj/Pij$, $Pij > 0$

Tabla I. Iteración nº 1

		Cj >>	3	2	0	0	0
Base	Cb	P0	P1	P2	P3	P4	P5
P3	0	18	2	1	1	0	0
P4	0	42	2	3	0	1	0
P5	0	24	3	1	0	0	1
Zj – Cj>>		0	-3	-2	0	0	0

4. Condición de parada

Para el caso de máximo la condición de parada es: $(Zj - Cj) > 0$

5. Condición de entrada y salida de la base

A. Primero debemos saber la variable que entra en la base. Para ello debemos tener en cuenta si el problema es de Máximo o Mínimo, en este caso (Máximo) entra:

Min:{Zj-Cj}, (Zj-Cj)<0 Min{-3,-2}

. En este caso sería la variable x (P1) de coeficiente - 3.

Si existiesen dos o más coeficientes iguales que cumplan la condición anterior (caso de empate), entonces se optará indistintamente por cualquiera.

La columna de la variable que entra en la base se llama columna pivote (En color verde).

5. Condición de entrada y salida de la base

B. Primero debemos saber la variable que entra en la base. Para ello escogemos la columna de aquel valor que en la fila Z sea el menor de los negativos. En este caso sería la variable x (P1) de coeficiente - 3.

Si existiesen dos o más coeficientes iguales que cumplan la condición anterior (caso de empate), entonces se optará por aquella variable que sea básica.

La columna de la variable que entra en la base se llama columna pivote (En color verde).

C. Una vez obtenida la variable que entra en la base, estamos en condiciones de deducir cual será la variable que sale. Para ello se divide cada término independiente (P0) entre el elemento correspondiente de la columna pivote, siempre que el resultado sea mayor que cero, y se escoge el mínimo de ellos.

En nuestro caso: 18/2 [=9], 42/2 [=21] y 24/3 [=8]

Si hubiera algún elemento menor o igual a cero no se realiza dicho cociente, y en caso de que todos los elementos de la columna pivote fueran de ésta condición tendríamos una solución no acotada y terminaríamos el problema.

El término de la columna pivote que en la división anterior de lugar al menor cociente positivo, el 3, ya que 8 es el menor cociente, indica la fila de la variable de holgura que sale de la base, t (P5). Esta fila se llama fila pivote (En color verde).

Si al calcular los cocientes, dos o más son iguales (caso de empate), se escoge aquella que no sea variable básica (si es posible).

D. En la intersección de la fila pivote y columna pivote tenemos el elemento pivote, 3.

6. Encontrar los coeficientes de la nueva tabla.

Los nuevos coeficientes de la fila pivote, t (P5), se obtienen dividiendo todos los coeficientes de dicha fila entre el elemento pivote, 3, que es el que hay que convertir en 1.

A continuación mediante la reducción gaussiana hacemos ceros los restantes términos de su columna, con lo que obtenemos los nuevos coeficientes de las otras filas incluyendo los de la función objetivo Z.

También se puede hacer de la siguiente manera:

Fila del pivote:

Nueva fila del pivote = (Vieja fila del pivote) / (Pivote)

Resto de las filas:

Nueva fila = (Vieja fila) -(Coeficiente de la vieja fila en la columna de la variable entrante) x (Nueva fila del pivote)

Veámoslo con un ejemplo una vez calculada la fila del pivote (fila de x (P1) en la Tabla II): 26

Vieja fila de P4	42	2	3	0	1	0
	-	-	-	-	-	-
Coeficiente	2	2	2	2	2	2
	x	x	x	x	x	x
Nueva fila pivote	8	1	1/3	0	0	1/3
	=	=	=	=	=	=
Nueva fila de P4	26	0	2.333	0	1	- 0.667

Tabla III. Iteración nº 3

			3	2	0	0	0
Base		P0	P1	P2	P3	P4	P5
P2		6	0	1	3	0	-2
P4		12	0	0	-7	1	4
P1		6	1	0	-1	0	1
Zj – Cj>>		30	0	0	3	0	-1

Como en los elementos de la fila Z hay uno negativo, -1, significa que no hemos llegado todavía a la solución óptima. Hay que repetir el proceso:

A. La variable que entra en la base es t (P5), por ser la variable que corresponde al coeficiente -1.

B. Para calcular la variable que sale, dividimos los términos de la última columna entre los términos correspondientes de la nueva columna pivote: 6/(-2) [=-3], 12/4 [=3], y 6/1 [=6] y como el menor cociente positivo es 3, tenemos que la variable que sale es s (P4).

C. El elemento pivote, que ahora hay que hacer 1, es 4.

Obtenemos la tabla:

Tabla IV. Iteración nº 4

			3	2	0	0	0
Base	Cb	P0	P1	P2	P3	P4	P5
P2	2	12	0	1	-0.5	0.5	0
P5	0	3	0	0	-1.75	0.25	1
P1	3	3	1	0	0.75	-0.25	0
Z		33	0	0	1.25	0.25	0

$x_1 = 3$

$x_2 = 12$

$Z = 3x + 2y$

$Z = 3 * 3 + 2 * 12$

La solución óptima es Z = 33

3. Análisis de sensibilidad.

Este no trata de resolver un problema de programación lineal. Entra en acción después que se ha encontrado una solución óptima.

Dentro del análisis de sensibilidad se automatizará:

- La sensibilidad para cuando se agrega una nueva variable al problema.
- La sensibilidad para cuando se agrega una nueva restricción al problema.

1.3 Sistemas automatizados existentes vinculados al campo de acción

Con el desarrollo de las nuevas tecnologías el universo de información ha crecido exponencialmente, provocando que los especialistas se vean en la necesidad de hacer uso cada vez más frecuentes de las ventajas que nos brinda los sistemas informáticos en la investigación de operaciones en la solución de problemas de programación lineal para la toma de decisiones, apoyando la idea anterior, podemos citar el siguiente planteamiento:

"El trabajo profesional vinculado a la información en cualquier organización, exige el dominio de un conjunto de variables que están presentes en su tratamiento. Asimismo, el dominio de las funciones de la gestión y de algunas de las principales herramientas que desarrolladas en las últimas décadas, no deviene en mero elemento cultural del profesional que maneja información, sino en una necesidad imperiosa". [3]

En la actualidad existe una gran cantidad de software y núcleos matemáticos dedicados a la solución de problemas de programación lineal como son Lingo, Lindo, Excel, Matemática, Phpsimplex, PLemath, Matlab, Maple. Los cuales independientemente que soportan métodos eficientes y bien validados que permiten una elevada eficacia en la soluciones de problemas medianamente grandes, no permiten una fácil integración a aplicaciones que soporten otras ramas de la Investigación de Operaciones como la programación en enteros, programación no lineal, planeación de proyectos, problemas de redes etc. Además que estos casi nunca son gratis y no son en su mayoría multiplataforma.

1.4 Tendencias y tecnologías actuales

1.4.1 Lenguajes de programación para el desarrollo de aplicaciones.

Teniendo en cuenta el desarrollo que han venido adquiriendo los deferentes lenguajes de programación para la creación de aplicaciones de escritorio y valorando los conocimientos alcanzados durante el estudio de la carrera. Hemos

considerado como posibles propuestas los lenguajes C++, Delphi, Java, siendo estos los más conocidos. Para los cuales se ha realizado una breve descripción de las características generales:

C++.

Es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

El nombre C++ fue propuesto por Rick Masciatti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases".

Se puede decir que abarca tres paradigmas de la programación:

La programación estructurada.

La programación genérica.

La programación orientada a objetos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución

Está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

Delphi.

Es un entorno de desarrollo de software (IDE) diseñado para la programación de propósito general con énfasis en la programación visual. El Delphi utiliza como lenguaje de programación una versión moderna de Pascal llamada

Object Pascal. Es producido comercialmente por la empresa estadounidense CodeGear. En sus diferentes variantes, permite producir archivos ejecutables para Windows, Linux y la plataforma .NET.

Está basado en una versión moderna de Pascal, denominada Object Pascal. Borland en los últimos años defendía que el nombre correcto del lenguaje es también Delphi, posiblemente debido a pretensiones de marca, aunque en sus mismos manuales el nombre del lenguaje aparecía como Object Pascal, por lo que la comunidad de programadores no ha adoptado mayoritariamente este cambio (supuesta aclaración, según Borland). Object Pascal expande las funcionalidades del Pascal estándar:

- Soporte para la programación orientada a objetos (habitualmente llamada POO) también existente desde Turbo Pascal 5.5, pero más evolucionada en cuanto a:
  - Encapsulación: declarando partes privadas, protegidas, públicas y publicadas de las clases
  - Propiedades: concepto nuevo que luego han adaptado muchos otros lenguajes. Las propiedades permiten usar la sintaxis de asignación para setters y getters.
  - Simplificación de la sintaxis de referencias a clases y punteros.
- Soporte para manejo estructurado de excepciones, mejorando sensiblemente el control de errores de usuario y del sistema.
- Programación activada por eventos (event-driven), posible gracias a la técnica de delegación de eventos. Esta técnica permite asignar el método de un objeto para responder a un evento lanzado sobre otro objeto. Fue adoptada por Niklaus Wirth, autor del Pascal Original, e incorporada a otros de sus lenguajes como Component Pascal.

Java.

Es un lenguaje de programación desarrollado por Sun Microsystems a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

Las características principales que nos ofrece Java son:

Simple

Ofrece toda la funcionalidad de un lenguaje potente. Debido a que C y C++ son los lenguajes más difundidos, Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Elimina muchas de las características de otros lenguajes para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello

y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Orientado a objetos

Con el objetivo de mantener la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, necesitan ser construidas y destruidas en espacios de memoria.

Incorpora funcionalidades como por ejemplo, la resolución dinámica de métodos mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

Distribuido

Se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

Proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Robusto

Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

También implementa los arrays auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobreescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.

Arquitectura neutral

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado. Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando en el porting a otras plataformas.

Seguro

La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el casting se eliminan para prevenir el acceso ilegal a la memoria. Cuando se usa Java para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.

El código Java pasa muchos tests antes de ejecutarse en una máquina. El código se pasa a través de un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal -código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto-.

El Cargador de Clases también ayuda a Java a mantener su seguridad, separando el espacio de nombres del sistema de ficheros local, del de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo Caballo de Troya, ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior.

Las clases importadas de la red se almacenan en un espacio de nombres privado, asociado con el origen. Cuando una clase del espacio de nombres privado accede a otra clase, primero se busca en las clases predefinidas (del sistema local) y luego en el espacio de nombres de la clase que hace la referencia. Esto imposibilita que una clase suplante a una predefinida.

Multihebra

Soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

1.4.2 Fundamentación de la selección de lenguaje a utilizar.

Teniendo en cuenta las características de los lenguajes antes mencionados hemos seleccionado Java para el desarrollo de nuestra aplicación por contar con las siguientes características:

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros.
- No existen referencias.
- Registros (struct).
- Definición de tipos (typedef).
- Macros (#define).
- Necesidad de liberar memoria (free).
- Aunque, en realidad, lo que hace es eliminar las palabras reservadas (struct, typedef), ya que las clases son algo parecido.
- Además, el intérprete completo de Java que hay en este momento es muy pequeño, solamente ocupa 215 Kb de RAM.

Con respecto a delphi tiene ventajas en cuanto a seguridad ya que al tener la característica de ser interpretado el código Java pasa a través de un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal -código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto. El Cargador de Clases, separando el espacio de nombres del sistema de ficheros local, del de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo Caballo de Troya, ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior.

1.4.3 *Metodologías para el desarrollo de Sistemas Informáticos*.

Unified Modeling Language (UML) o Lenguaje Unificado de Modelado, un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.
UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

Es un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes.

Objetivos del UML

- UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. UML incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
- Ser tan simple como sea posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la

concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

- Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
- Imponer un estándar mundial.

Fundamentación de la metodología a utilizar para el modelado.

Existen varias metodologías que usan el lenguaje UML para indicar el camino a seguir para el desarrollo de sistemas de informáticos las cuales han ido evolucionando. Algunas de ellas son: OMT, XP, Microsoft Solution Framework, OBJECTORY, BOOCH, RUP y AUP.

Para el desarrollo de este proyecto hemos tomado en consideración utilizar la metodología el Proceso Unificado del Racional (RUP) para el cual se hace una caracterización a continuación.

Los aspectos que definen el Proceso Unificado se resumen en tres frases claves:

- dirigido por casos de uso
- centrado en la arquitectura
- iterativo e incremental

Un ***caso de uso*** es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el ***modelo de casos de uso*** que describe la funcionalidad total del sistema.

Los casos de uso guían el proceso entero de desarrollo.

La **arquitectura del software** es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado.

Cada producto tiene una función y una forma. Ninguna es suficiente por sí misma, debiendo interactuar entre ellas y evolucionar en paralelo. La función corresponde a los casos de uso y la forma a la arquitectura.

Para encontrar la forma, la arquitectura, los arquitectos deben trabajar sobre la comprensión de los casos de uso claves que vienen a suponer entre el 5 y el 10 por ciento del total de casos de uso.

El Proceso Unificado es **iterativo e incremental** lo que supone dividir el proyecto en pequeñas partes denominadas mini-proyectos. Cada mini-proyecto es una ***iteración*** que genera un ***incremento***. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos al crecimiento del producto.

Las iteraciones deben ser controladas, deben ser seleccionadas y ejecutarse de forma planificada. Por estas razones son consideradas mini-proyectos.

Una iteración trata un conjunto de casos de uso que amplían la utilidad del producto desarrollado hasta ese momento. Cada iteración gestiona **los riesgos** identificados más importantes.

Este proceso iterativo trae como beneficios:

- Reduce el riesgo a los costes de un solo incremento.
- Reduce el riesgo de no sacar el producto en plazo.
- Los trabajadores son más eficientes al trabajar para obtener resultados a corto plazo.
- Buena adaptación a los requisitos cambiantes.

La **arquitectura** proporciona la estructura sobre la que guiar las **iteraciones**, mientras que los **casos de uso** definen los objetivos y dirigen el trabajo de cada **iteración**.

El Proceso Unificado se repite a lo largo de una serie de **ciclos** que constituyen la vida de un sistema. Cada **ciclo** concluye con una **nueva versión del producto**. El **producto terminado** incluye *los requisitos, casos de uso, especificaciones no funcionales, casos de prueba, código fuente incluido en componentes ejecutables, manuales, el modelo de arquitectura y el modelo visual (modelos UML*).

Cada **ciclo** consta de **cuatro fases**: **Inicio, Elaboración, Construcción y Transición.**

Cada **fase**, a su vez, se subdivide en **iteraciones**. Cada **fase** debe terminar en un **hito** que contemple la disponibilidad de ciertos modelos o documentos. El **hito** implica la toma de decisiones.

Una iteración típica dentro de una fase pasa por los siguientes flujos de trabajo:

- **Requisitos**
- **Análisis**
- **Diseño**
- **Implementación**
- **Prueba**

Descripción de las Fases del Ciclo:

Fase de INICIO

- Se desarrolla una descripción del producto final y se presenta el análisis de negocio para el producto.
- Se construye un modelo de casos de uso simplificado.
- Se esbozan los subsistemas más importantes lo que origina una arquitectura provisional.

- Se identifican los riesgos más importantes.
- Se planifica la fase de ELABORACIÓN.
- Se estima el proyecto de manera aproximada.

Fase de ELABORACIÓN

- Se especifican en detalle la mayoría de los casos de uso del producto.
- Se diseña la arquitectura del sistema a través de vistas de todos los modelos del sistema (análisis, diseño, implementación y despliegue), obteniéndose una **línea base de la arquitectura**.
- Se realizan los casos de uso más críticos identificados en la

Fase de INICIO.

- El jefe del proyecto ya está en disposición de planificar y estimar los recursos necesarios para terminar el proyecto.

Fase de CONSTRUCCION

- Se crea el producto.
- La **línea base de la arquitectura** crece hasta convertirse en el sistema completo.
- La descripción evoluciona hasta convertirse en un **producto** preparado para entregarse al usuario.
- Al final de esta fase, el producto contiene todos los casos de uso acordados con el usuario, aunque puede que no estén libres de defectos.
- Muchos de los defectos se descubrirán y solucionarán

Fase de TRANSICIÓN.

- Cubre el período en que el **producto** se convierte en versión beta.
- Los desarrolladores corrigen los defectos detectados e incorporan algunas mejoras.

- Esta fase conlleva a actividades de fabricación, formación a usuarios, corrección de defectos tras la entrega.
- El equipo de mantenimiento valora los defectos tras la entrega en dos categorías: los que tienen suficiente impacto para justificar una versión incrementada y los que pueden corregirse en la siguiente versión normal.

1.5 Conclusiones

En este capítulo se realiza un análisis completo de las tecnologías que serán utilizadas a lo largo del desarrollo del sistema propuesto, y se fundamentan las elecciones del lenguaje, el sistema gestor de bases de datos, y la metodología a utilizar. Una vez conocidas las herramientas óptimas, y los conceptos a utilizar podemos empezar a desarrollar la propuesta de sistema.

Capítulo 2 Modelo del dominio

2.1 Introducción

Analizando la descripción de los procesos realizada en el capitulo I, llegamos a la conclusión de que el negocio que se esta estudiando tiene muy bajo nivel de estructuración, RUP propone para estos casos realizar un modelo del dominio; que no es más que una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés, por lo que permite mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema. Tal modelo no incluye las responsabilidades que llevan a cabo las personas

De modo que en el presente capítulo se especificará lo siguiente:

1. Definición de entidades y los conceptos principales. Las cuales se definirán a partir de su símbolo, su intención, y su extensión.
   Símbolo: Palabra que representa el concepto
   Intensión: La definición del concepto
   Extensión: El conjunto de ejemplos a que se aplica el concepto
2. Diagrama del Modelo del Dominio.
3. Representación de los requerimientos funcionales y no funcionales del sistema.

2.2 Definición de las entidades y los conceptos principales

Tabla 1 Entidades y conceptos del dominio.

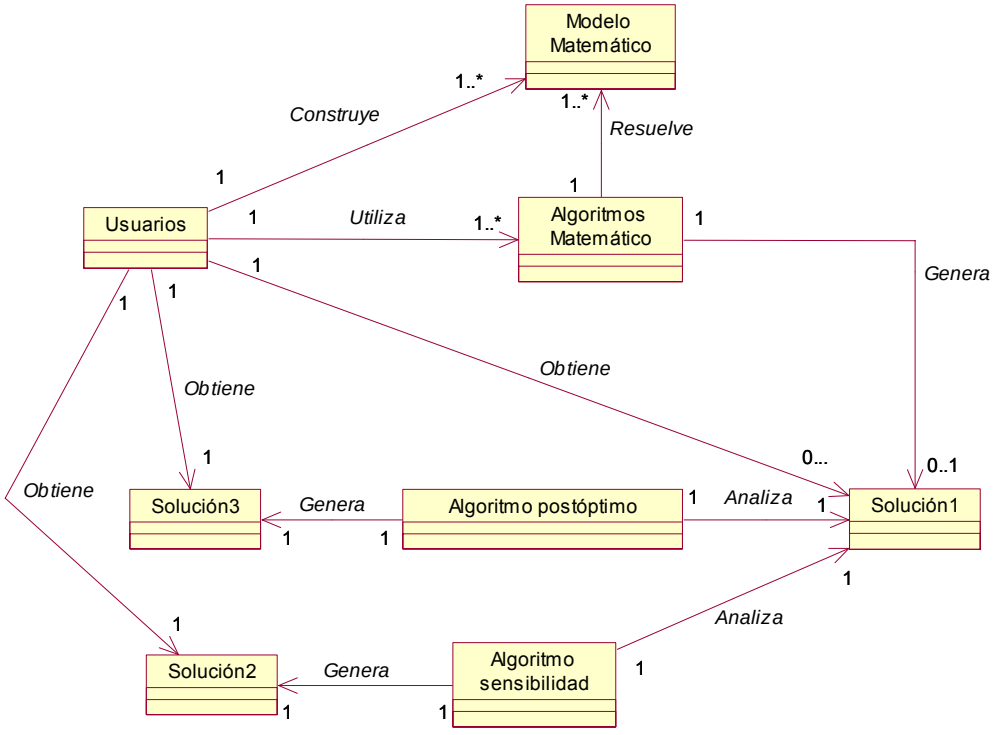
Símbolo	Intensión	Extensión
Usuario	Se le denomina usuarios a todas aquellas personas que interactúan con el sistema.	Personas
Modelo Matemático	Se refiere al conjunto de ecuaciones que constituyen las	Conjunto de ecuaciones

	restricciones del problema de programación lineal, la función objetivo, y descripción de las variables.	
Algoritmos Matemáticos	Se refieren a los métodos para la solución de problemas de programación lineal	Método Simplex Original y Simplex Revisado
Solución1	Es la solución obtenida una vez resuelto los modelos matemáticos ya sea por el método simplex o simplex revisado	Solución óptima
Algoritmo postóptimo	Es el conjunto de pasos que permite realizar el análisis de precio sombra y la interpretación económica de la solución óptima, obtenida como resultado del método simplex o simplex revisado.	Algoritmos para análisis de precio sombra y la interpretación económica
Análisis de sensibilidad	Es el conjunto de pasos que permite realizar el análisis, de la solución óptima obtenida como resultado del método simplex.	Algoritmos para análisis de sensibilidad coeficiente de la función objetivo para variables dentro o fuera de la solución, análisis de sensibilidad si se agrega una nueva variable o una nueva restricción
Solución2	Es la solución obtenida a partir del análisis de sensibilidad realizado a la solución óptima obtenida una vez resuelto los	-

	modelos matemáticos ya sea por el método simples.	
Solución3	Es la solución obtenida a partir del análisis postóptimo realizado a la solución óptima obtenida una vez resuelto los modelos matemáticos ya sea por el método simplex o simplex revisado.	-

2.3 Representación del modelo del dominio

Figura 1 Diagrama del Modelo de Dominio.



2.4 Requerimientos Funcionales y no Funcionales del Sistema.

2.4.1 Requisitos Funcionales

Los requisitos funcionales indican el comportamiento del sistema. Posteriormente estos requisitos son modelados a través del diagrama de casos de uso del sistema.

Requerimientos funcionales "Gestión de modelos".

1. Insertar modelos.
   1.2 Guardar modelos.
   1.3 Editar modelos.
2. Cargar modelos.

Requerimientos funcionales "Solucionar modelos".

3. Solucionar Modelo Primal.
4. Calcular modelo dual.
   4.1 Solucionar Modelo Dual.

Requerimientos funcionales "Análisis de sensibilidad".

5. Sensibilidad si se agrega una nueva variable.
6. Sensibilidad si se agrega una nueva restricción.

Requerimientos funcionales "Análisis postóptimo".

7. Interpretación económica.
8. Reporte postóptimo.

2.4.2 Requisitos no Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

- **Apariencia o interfaz externa.**
  La interfaz debe ser sencilla y amigable puesto que los usuarios pueden o no ser personas expertas. La respuesta del sistema ha de ser rápida.

- **Usabilidad.**

  El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora

- **Rendimiento.**

  Esta aplicación debe tener un rendimiento óptimo. Debe ser rápida y el tiempo de respuesta debe ser el mínimo posible.

- **Portabilidad.**

  El sistema es multiplataforma, todo depende de la portabilidad que alcance la maquina virtual de java

- **Ayuda y documentación en línea.**

  El sistema cuenta con una opción de ayuda y un manual de usuário en version digital, la cual guía al usuario en el funcionamiento del sistema.

- **Software**.

  Se necesita tener instalada la Máquina Virtual de Java, versión JRE 1.6.0 o superior.

2.5 Conclusiones.

En este capítulo se realizó una descripción del funcionamento del sistema representado por un modelo de dominio, asi como se definieron los requerimientos funcionales y no funcionales del sistema propuesto, obteniéndose a partir del análisis de los procesos del dominio, la cual permitirá elaborar el Diagrama de Casos de Uso.

Capítulo 3 Diseño e Implementación del Sistema

3.1 Introducción

Se ofrecen aspectos que dentro de la metodología de ingeniería del software empleada, brindan informaciones específicas para el correcto entendimiento de la solución propuesta, Se definirá los actores del sistema, se realizará una reperserntación del diagrama de Casos de Uso con los que interactúa el usuario, asi como los Diagrama de Clases, de Secuencia y de Despliegue.

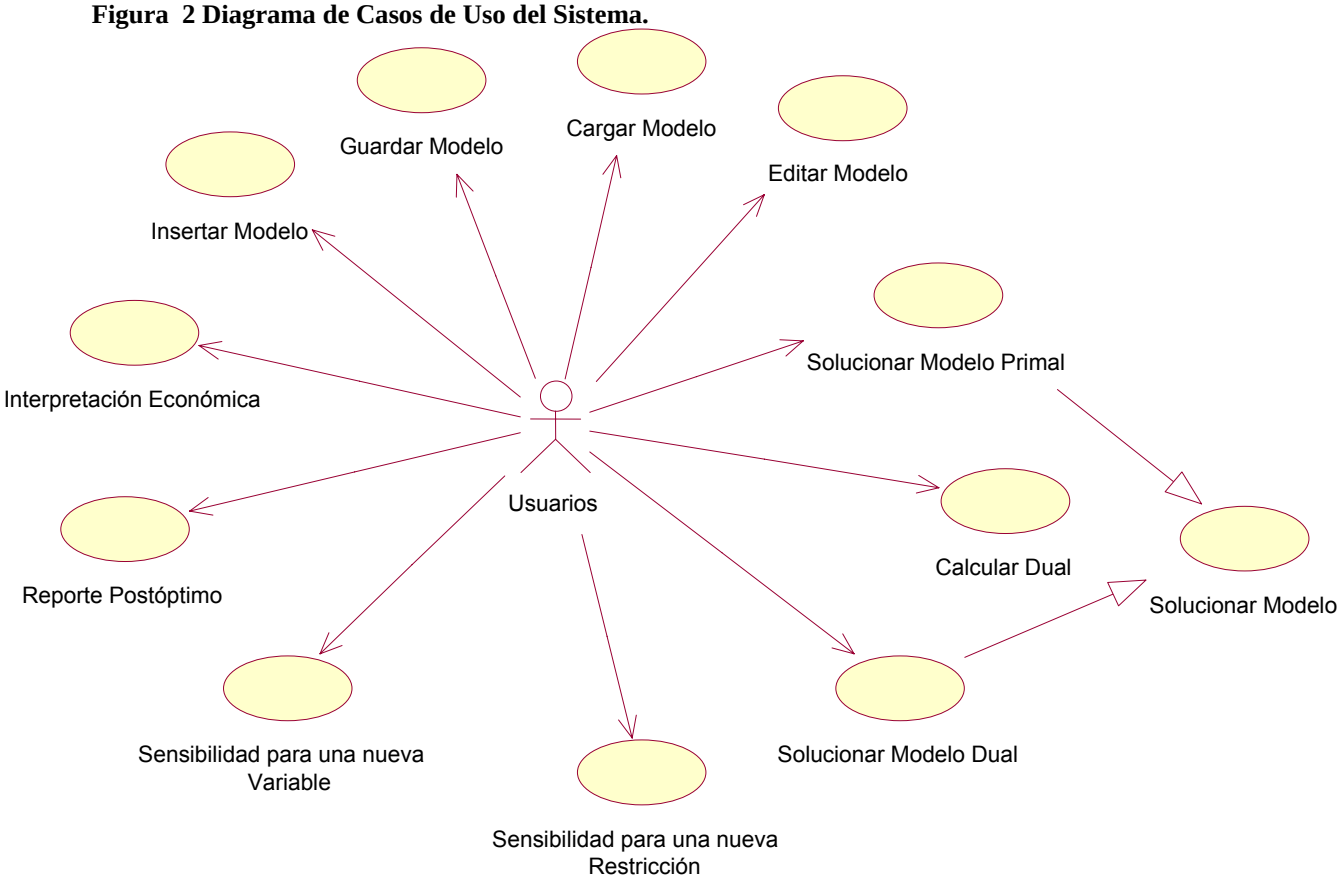
3.2 Actores del sistema a automatizar

Tabla 2. Definición de actores del sistema a automatizar

Nombre del actor	Descripción
Usuarios	Se le denomina usuarios a todas aquellas personas que interactúan con el sistema.

3.3 Diagrama de casos de uso del sistema a automatizar

Figura 2 Diagrama de Casos de Uso del Sistema.



3.4 Descripción de los casos de uso

Tabla 3. Descripción del caso de uso "Insertar Modelo"

Nombre del caso de uso	Insertar Modelo
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide introducir un nuevo modelo matemático al sistema
Referencias	R1
Precondiciones	-
Poscondiciones	Queda insertado el modelo en el sistema listo para ser procesado
Requisitos especiales	Se debe validar que sean introducidos todos los datos

Tabla 4. Descripción del caso de uso "Guardar Modelo"

Nombre del caso de uso	Guardar Modelo
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide guardar el modelo matemático insertado en el sistema
Referencias	R1.2
Precondiciones	Debe haberse insertado un nuevo modelo matemático antes en el sistema
Poscondiciones	Queda guardado el modelo en un fichero
Requisitos especiales	Se debe validarse que hallan introducidos todos los datos anteriormente

Tabla 5. Descripción del caso de uso "Editar Modelo"

Nombre del caso de uso	Editar Modelo
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide editar un modelo matemático en el sistema esta comprende tres acciones fundamentales editar datos, agregar una nueva restricción, agregar una nueva variables.
Referencias	R1.3
Precondiciones	Debe haberse insertado o cargado el modelo matemático antes
Poscondiciones	Queda modificado el modelo en el sistema listo para ser procesado
Requisitos especiales	

Tabla 6. Descripción del caso de uso "Cargar Modelo"

Nombre del caso de uso	Cargar Modelo
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide cargar un modelo matemático en el sistema
Referencias	R2
Precondiciones	Debe haberse guardado el modelo matemático antes
Poscondiciones	Queda cargado el modelo en el sistema listo para ser procesado
Requisitos especiales	Debe validarse que exista el fichero con el modelo correcto

Tabla 7. Descripción del caso de uso "Solucionar Modelo Primal"

Nombre del caso de uso	Solucionar Modelo Primal.
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide obtener la solución de un modelo matemático
Referencias	R3
Precondiciones	Debe haberse insertado o cargado el modelo matemático antes
Poscondiciones	El sistema devuelve como resultado la solución óptima del modelo primal
Requisitos especiales	-

Tabla 8. Descripción del caso de uso "Calcular Modelo Dual"

Nombre del caso de uso	Calcular Modelo Dual.
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide calcular el modelo dual
Referencias	R4
Precondiciones	Debe haberse insertado o cargado el modelo matemático antes
Poscondiciones	El sistema devuelve como resultado el modelo dual
Requisitos especiales	-

Tabla 9. Descripción del caso de uso "Solucionar Modelo Dual"

Nombre del caso de uso	Solucionar Modelo Dual.
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario cuando decide obtener la solución de un modelo matemático
Referencias	R4.1
Precondiciones	Debe haberse calculado el modelo dual antes
Poscondiciones	El sistema devuelve como resultado la solución óptima del modelo dual
Requisitos especiales	

Tabla 10. Descripción del caso de uso "Reporte Postóptimo"

Nombre del caso de uso	Reporte Postóptimo
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario luego de obtener la solución del modelo matemático, en este caso permite realizar un análisis de variaciones singulares o combinadas en los requerimientos originales, así como de otras posibles redistribuciones de la disponibilidad o exigencia de partida.
Referencias	R8
Precondiciones	Debe haberse solucionado el modelo matemático
Poscondiciones	El sistema devuelve como resultado la el análisis obtenido de la solución del modelo.
Requisitos especiales	-

Tabla 11. Descripción del caso de uso "Sensibilidad si se agrega una nueva variable"

Nombre del caso de uso	Sensibilidad si se agrega una nueva variable
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario luego de obtener la solución del modelo matemático en este caso el sistema debe realizar un análisis del resultado obtenido en la solución del modelo.
Referencias	R5
Precondiciones	Debe haberse solucionado el modelo matemático
Poscondiciones	El sistema devuelve como resultado la el análisis obtenido
Requisitos especiales	

Tabla 12. Descripción del caso de uso "Sensibilidad si se agrega una nueva restricción"

Nombre del caso de uso	Sensibilidad si se agrega una nueva restricción
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario luego de obtener la solución del modelo matemático en este caso el sistema debe realizar un análisis del resultado obtenido en la solución del modelo.
Referencias	R6
Precondiciones	Debe haberse solucionado el modelo matemático
Poscondiciones	El sistema devuelve como resultado la el análisis obtenido
Requisitos especiales	

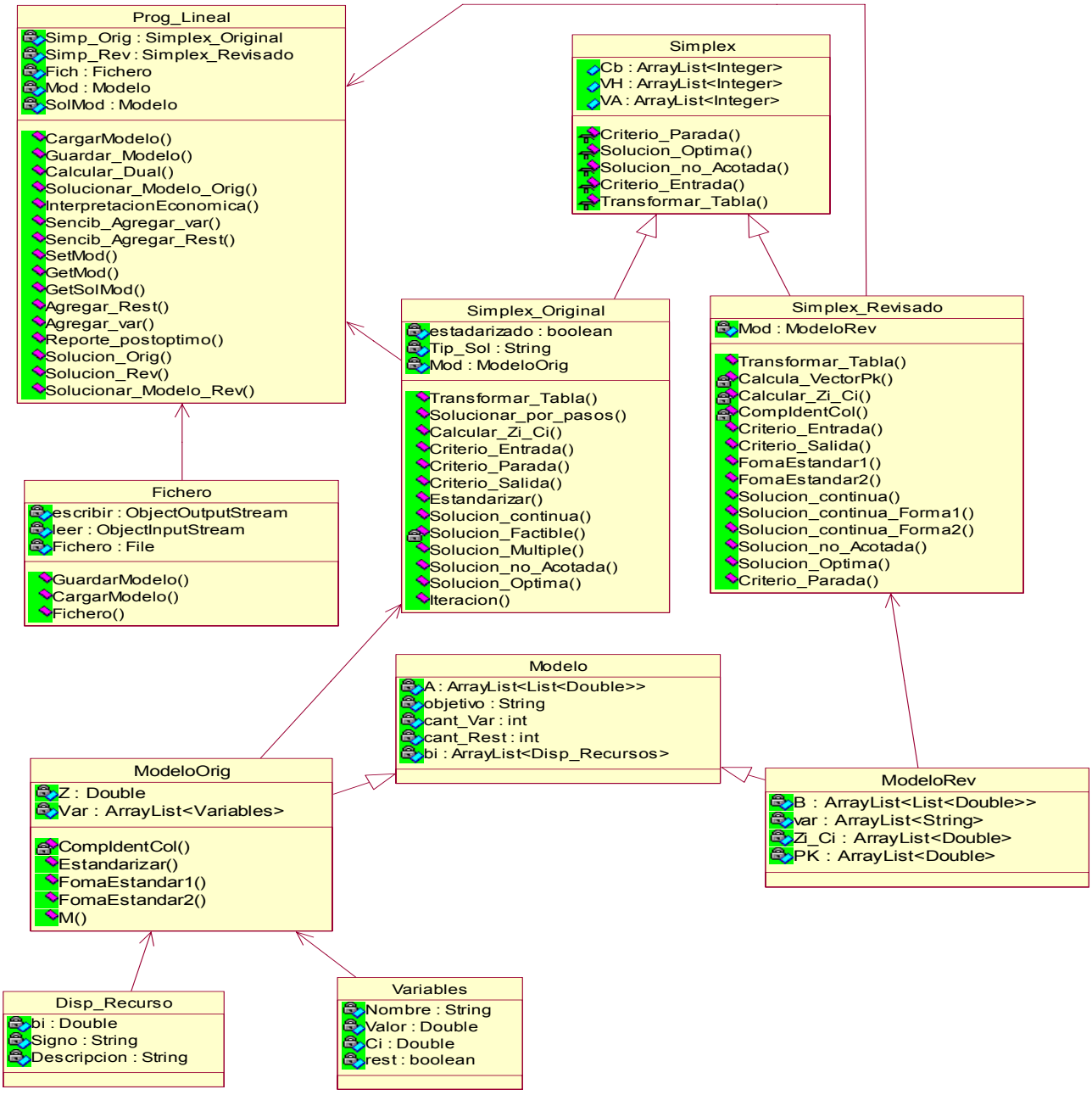
Tabla 13. Descripción del caso de uso "Interpretación económica"

Nombre del caso de uso	Interpretación económica.
Actores	Usuario(inicia)
Resumen	Este caso de uso lo inicia el usuario luego de obtener la solución del modelo matemático en este caso el sistema debe realizar un análisis del resultado obtenido en la solución del modelo.
Referencias	R7
Precondiciones	Debe haberse solucionado el modelo matemático
Poscondiciones	El sistema devuelve como resultado la el análisis obtenido
Requisitos especiales	

3.5 Diagrama de clases del diseño

3.5.1 Diagrama de Clases General

Figura 3 Diagrama de clase del sistema.



3.5.2 Descripción General de las Clases.

Tabla 14 Descripción de la clase "Prog_Lineal".

Nombre: Prog_Lineal	
Tipo de clase: Controladora	
Atributo	**Tipo**
Simp_Orig	Simplex_Original
Simp_Rev	Simplex_Revisadol
Fich	Fichero
Mod	Modelo
SolMod	Modelo
Para cada responsabilidad:	
Nombre:	**Descripción**
CargarModelo (File Archivo)	Se entra por parámetro una dirección, llama a una función cargar modelo y devuelve como resultado un modelo.
Guardar_Modelo (File Archivo)	Se entra por parámetro una dirección, llama a un procedimiento guardar modelo.
Calcular_Dual ()	Calcula el dual de un modelo primal y devuelve como resultado el modelo modificado.
Solucionar_Modelo_Orig (String forma)	Se pasa por parámetro la forma por la que va a dar solución al modelo ya sea "directo o por pasos", mediante el método simplex original Y llama a las funciones según la condición anterior.
Solucionar_Modelo_Rev ()	Realiza un llamado a un procedimiento para la solución del

	modelo por el método simplex revisado.
InterpretacionEconomica ()	Devuelve una interpretación de los resultados obtenidos de la solución de un modelo.
Reporte_posoptimo ()	Devuelve en un arreglo un análisis de variaciones singulares o combinadas en los requerimientos originales, así como de otras posibles redistribuciones de la disponibilidad o exigencia de partida.
Sensib_agregar_Var (Double Cj, Double [] ciA)	Realiza el análisis de sensibilidad para cuando se agrega una nueva variable.
Sensib_agregar_Rest (Double bi, String sig, Double [] ciA)	Realiza el análisis de sensibilidad para cuando se agrega una nueva restricción.
GetMod ()	Devuelve como resultado el modelo inicial.
GetSolMod ()	Devuelve como resultado el modelo resuelto.
SetMod (ModeloOrig aMod)	Se pasa por parámetro un modelo y se modifica el modelo inicial.
Solucion_Rev ()	Devuelve un resumen de la solución del modelo resuelto por el método simplex revisado.
Solucion_Orig ()	Devuelve un resumen de la solución del modelo resuelto por el método simplex original.

Agregar_Rest (Disp_Recursos bi, Double [] ci)	Agrega una nueva restricción al modelo.
Agregar_Var (Variables var, Double [] cj)	Agrega una nueva variable al modelo.
getModificado()	Devuelve verdadero si se modifica el modelo y falso en caso contrario.

Tabla 15. Descripción de la clase "Fichero".

Nombre: Fichero	
Tipo de clase: Controladora	
Atributo	**Tipo**
Fichero	File
Para cada responsabilidad:	
Nombre:	**Descripción**
GuardarModelo(ModeloOrig modelo)	Se entra por parámetro un modelo y lo guardar en un fichero.
CargarModelo ()	Lee el modelo de un fichero lo devuelve como resultado.
Fichero (File aFichero)	Constructor

Tabla 16. Descripción de la clase "Modelo".

Nombre: Modelo	
Tipo de clase: Entidad	
Atributo	**Tipo**
bi	ArrayList<Disp_Recursos>
A	ArrayList<ArrayList<Double>>
objetivo	String
CantRest	int
CantVar	int
Para cada responsabilidad:	
Nombre:	**Descripción**

Tabla 17. Descripción de la clase "ModeloOrig".

Nombre: ModeloOrig	
Tipo de clase: Entidad	
Atributo	**Tipo**
var	ArrayList <Variables>
Z	Double
Para cada responsabilidad:	
Nombre:	**Descripción**
CompIdentCol (int fila)	Devuelve la columna del elemento en la fila cuyo valor es 1 y los valores de la columna correspondiente son 0 y devuelve -1 en caso contrario.
Estandarizar ()	Devuelve verdadero si el modelo es estandarizado y falso en caso contrario.
FormaEstandar1 ()	Estandariza el modelo y devuelve verdadero en caso de que el modelo contenga la matriz idéntica o las restricciones sean ≤ y en caso de tener restricciones de igualdad que tenga una variable con coeficiente 1 y los elementos correspondiente a esa columna sean cero 0.
FormaEstandar2 ()	Estandariza el modelo cuando no se cumple la forma estándar 1.
M ()	Calcula y devuelve el valor del coeficiente de las variables artificiales.

Tabla 18. Descripción de la clase "ModeloRev".

Nombre: ModeloRev	
Tipo de clase: Entidad	
Atributo	**Tipo**
var	ArrayList<String>
B	ArrayList<ArrayList<Double>>

Zi_Ci	ArrayList <Double>
Pk	ArrayList <Double>
Para cada responsabilidad:	
Nombre:	**Descripción**

Tabla 19. Descripción de la clase "Dsip_Recursos".

Nombre: Disp_Recursos	
Tipo de clase: Entidad	
Atributo	Tipo
bi	Double
signo	String
Descrip	String
Para cada responsabilidad:	
Nombre:	**Descripción**

Tabla 20. Descripción de la clase "Variables".

Nombre: Variables	
Tipo de clase: Entidad	
Atributo	Tipo
nombre	String
valor	Double
ci	Double
rest	boolean
Para cada responsabilidad:	
Nombre:	**Descripción**

Tabla 21. Descripción de la clase "Simplex".

Nombre: Simplex
Tipo de clase: Controladora

Atributo	**Tipo**
Cb	ArrayList< Integer>
VH	ArrayList< Integer>
VA	ArrayList< Integer>
Para cada responsabilidad:	
Nombre:	**Descripción**
Criterio_Parada ()	Es un método abstracto que calcula si se puede realizar una iteración de la solución y devuelve el tipo de solución.
Solucion_Optima ()	Es un método abstracto que devuelve verdadero cuando se ha alcanzado una solución óptima.
Solucion_no_Acotada ()	Es un método abstracto que devuelve verdadero cuando se ha alcanzado una solución no acotada.
Criterio_Entrada ()	Es un método abstracto que calcula cual es el elemento que entra a la base y devuelve la columna de dicho elemento
Transformar_Tabla (int filaP, int colP)	Es un método abstracto que calcula los valores del modelo durante una iteración.

Tabla 22. Descripción de la clase "Simplex_Original".

Nombre: Simplex_Original	
Tipo de clase: Controladora	
Atributo	**Tipo**
estandarizado	boolean
Tip_Sol	String
Mod	ModeloOrig

Para cada responsabilidad:	
Nombre:	**Descripción**
Solucionar_por_pasos ()	Soluciona una iteración del modelo y devuelve dicho modelo.
Transformar_Tabla (int filaP, int colP)	Dado la fila y la columna del pivote calcula los valores del modelo durante una iteración.
Calcular_Zi_Ci ()	Calcula los valores Zi – Ci para encontrar el criterio de parada.
Criterio_Entrada ()	Calcula cual es el elemento que entra a la base y devuelve la columna de dicho elemento
Criterio_Parada ()	Calcula si se puede realizar una iteración de la solución y devuelve el tipo de solución.
Criterio_Salida (int Columna)	Se entra por parámetro el elemento que entra a la base y devuelve la fila del elemento que sale de la base.
Estandarizar ()	Convierte las restricciones en igualdades y devuelve verdadero en caso de lograrlo.
Solucion_continua ()	Soluciona el modelo y devuelve el resultado de la solución del modelo.
Solucion_Factible ()	Devuelve verdadero cuando se ha alcanzado una solución factible.
Solucion_Multiple ()	Devuelve verdadero cuando se ha alcanzado una solución múltiple.
Solucion_no_Acotada ()	Devuelve verdadero cuando se ha alcanzado una solución no acotada.
Solucion_Optima ()	Devuelve verdadero cuando se ha alcanzado una solución óptima
Iteracion ()	Realiza una iteración de en la solución del modelo.

Tabla 23. Descripción de la clase "Simplex_Revisado".

Nombre: Simplex_Revisado	
Tipo de clase: Controladora	
Atributo	**Tipo**
Mod	ModeloRev
Para cada responsabilidad:	
Nombre:	**Descripción**
Transformar_Tabla (int filaP, int colP)	Dado la fila y la columna del pivote calcula los valores del modelo durante una iteración.
Calcula_VectorPk (int col)	Calcula los valores Pk para encontrar el criterio de entrada a la base.
Calcular_Zi_Ci (String forma)	Calcula los valores Zi – Ci para encontrar el criterio de parada.
CompIdentCol (int fila)	Devuelve la columna del elemento cuyo valor es 1 y los valores de la columna correspondiente son 0 y devuelve -1 en caso contrario.
Criterio_Entrada ()	Calcula cual es el elemento que entra a la base y devuelve la columna de dicho elemento
Criterio_Salida (String forma, int Columna)	Se entra por parámetro el elemento que entra a la base y devuelve la fila del elemento que sale de la base.
FomaEstandar1 ()	Convierte las restricciones en igualdades y devuelve verdadero en caso de que el modelo contenga la matriz idéntica o las restricciones sean ≤ y en caso de tener restricciones de igualdad que tenga una variable con coeficiente 1 y los elementos correspondiente a esa columna sean cero 0.

FormaEstandar2 ()	Convierte las restricciones en igualdades, en caso contrario a la forma estándar 1.
Solucion_continua ()	Soluciona el modelo y devuelve el resultado de la solución del modelo.
Solucion_continua_Forma1 ()	Soluciona el modelo y devuelve el resultado de la solución del modelo.
Solucion_continua_Forma2 ()	Soluciona el modelo y devuelve el resultado de la solución del modelo.
Solucion_no_Acotada ()	Devuelve verdadero cuando se ha alcanzado una solución no acotada.
Solucion_Optima ()	Devuelve verdadero cuando se ha alcanzado una solución óptima.
Criterio_Parada ()	Calcula si se puede realizar una iteración de la solución y devuelve el tipo de solución.

3.6 Principios de diseño

3.6.1 Interfaz de usuario

Figura 4 Interfaz de usuario "Insertar Modelo 1".

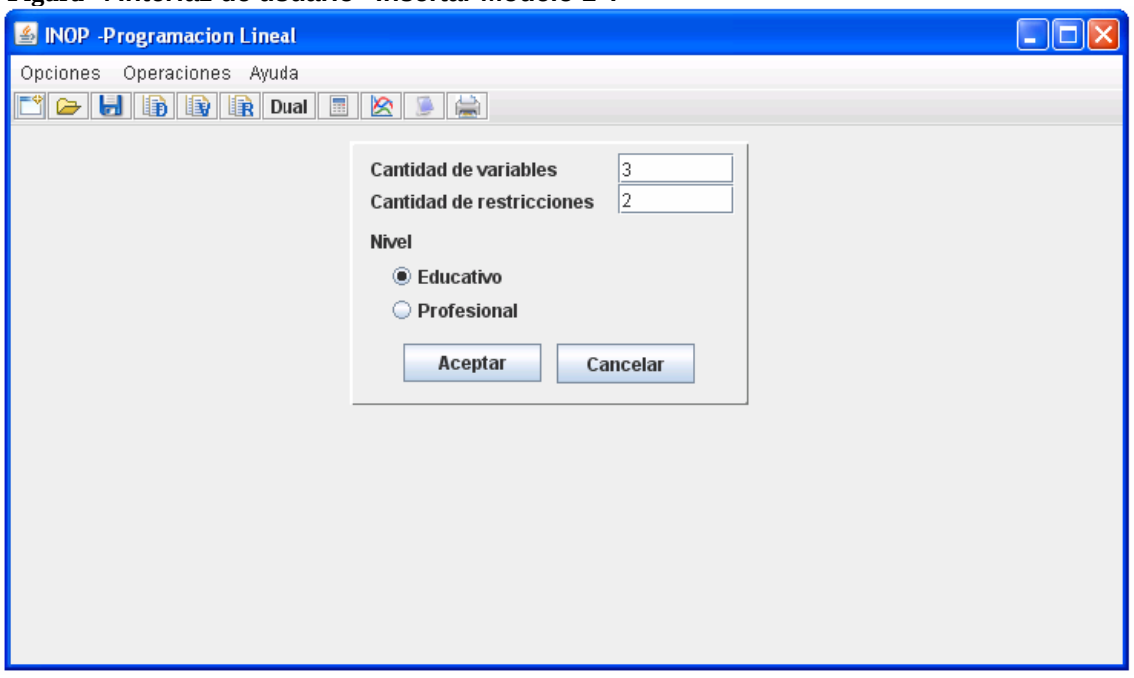


Figura 5 Interfaz de usuario "Insertar Modelo 2".

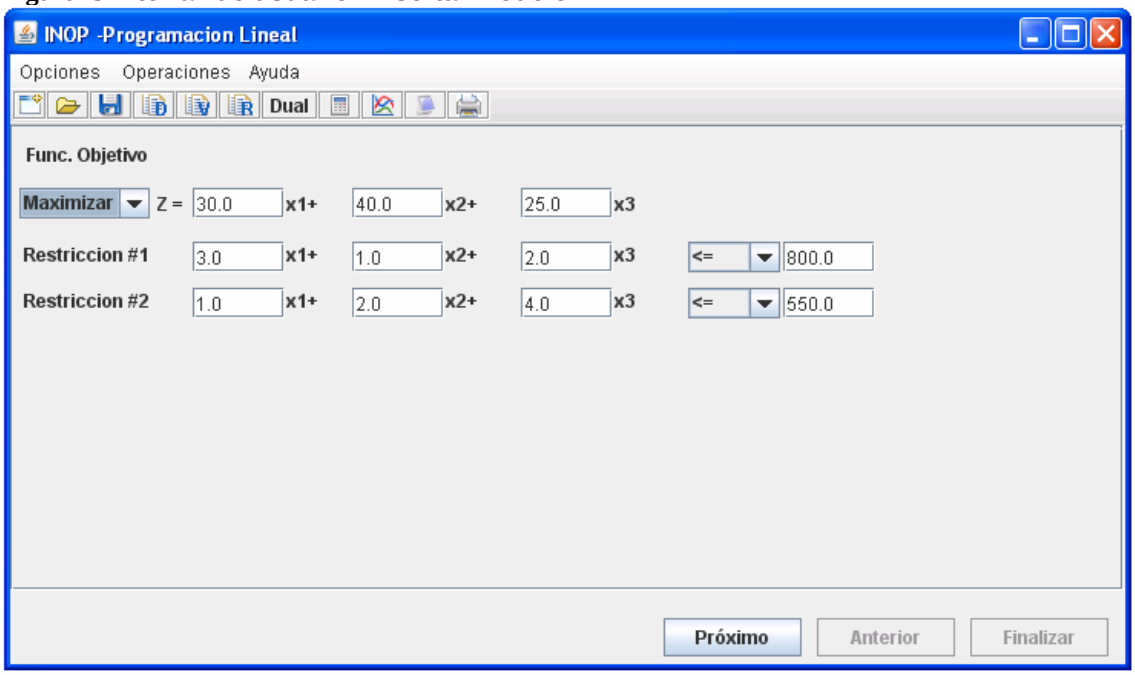


Figura 6 Interfaz de usuario "Insertar Modelo 3".

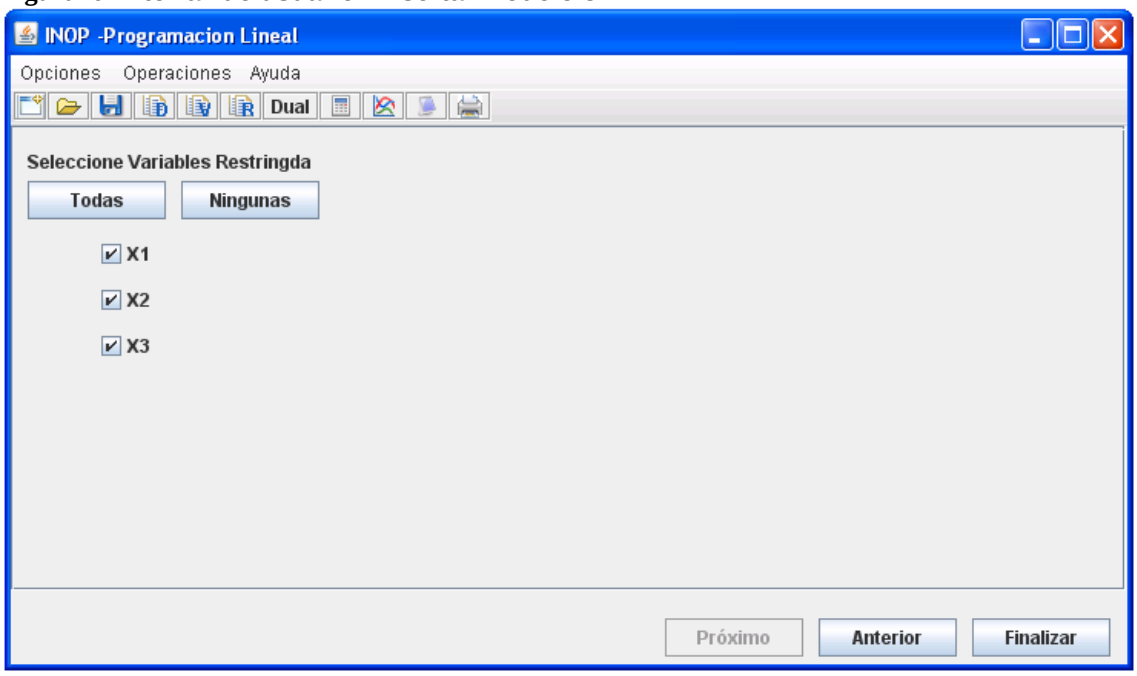


Figura 7 Interfaz de usuario "Insertar Modelo 4".

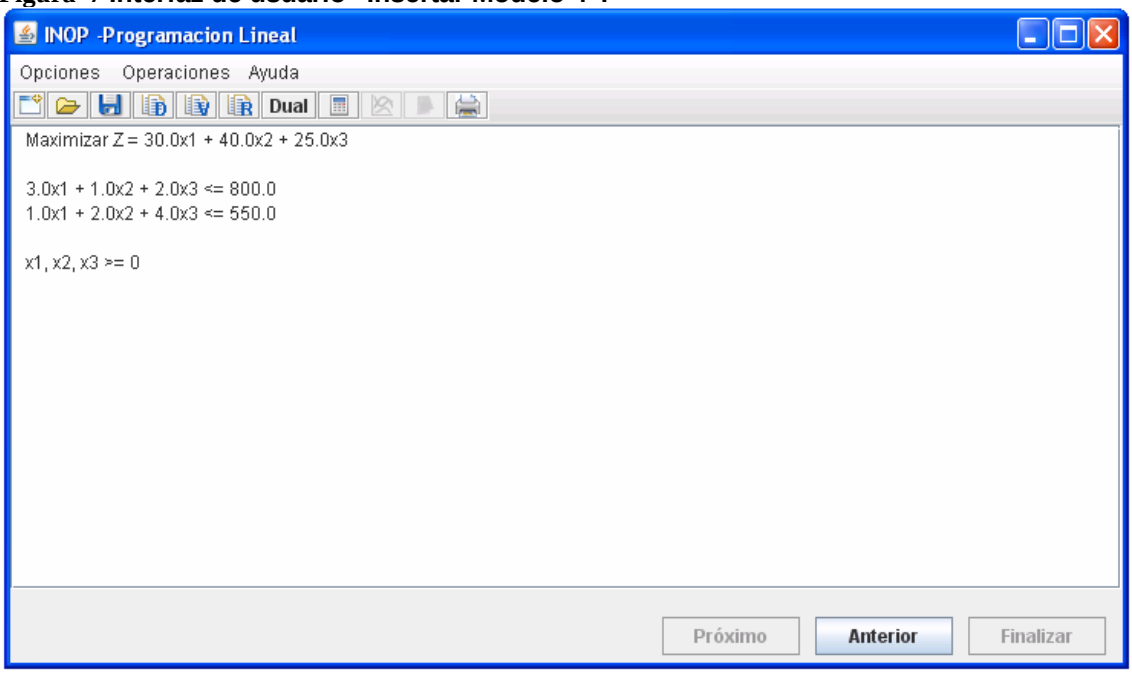


Figura 8 Interfaz de usuario "Solucionar Modelo".

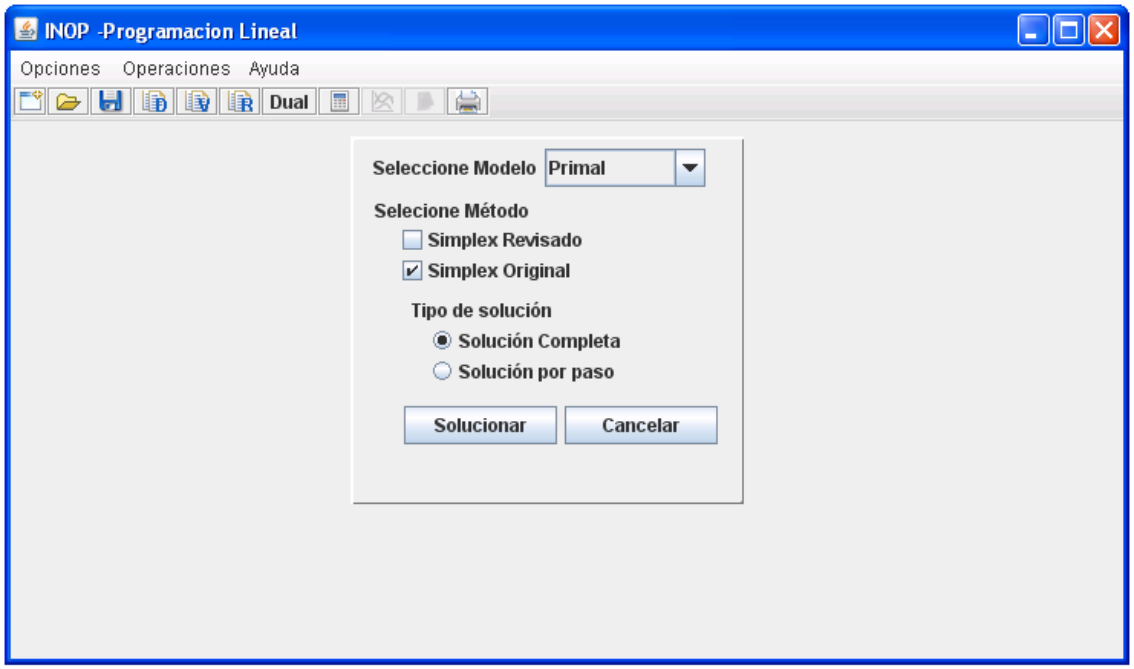


Figura 9 Interfaz de usuario "Agregar nueva variable".

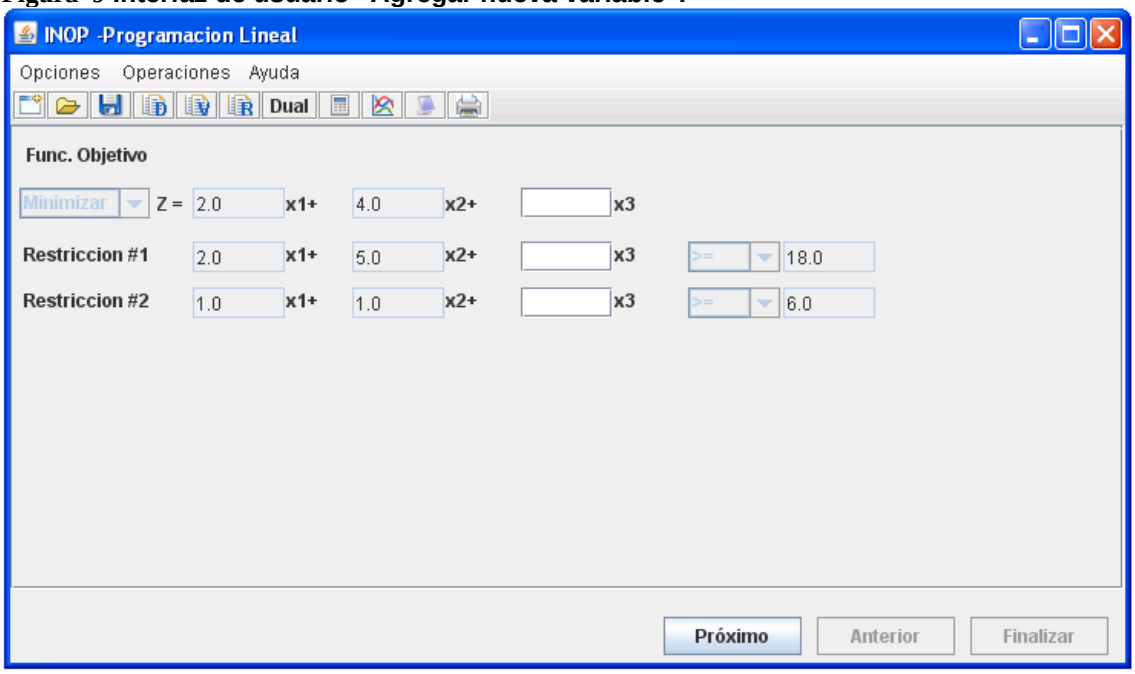


Figura 10 Interfaz de usuario "Agregar nueva restricción".

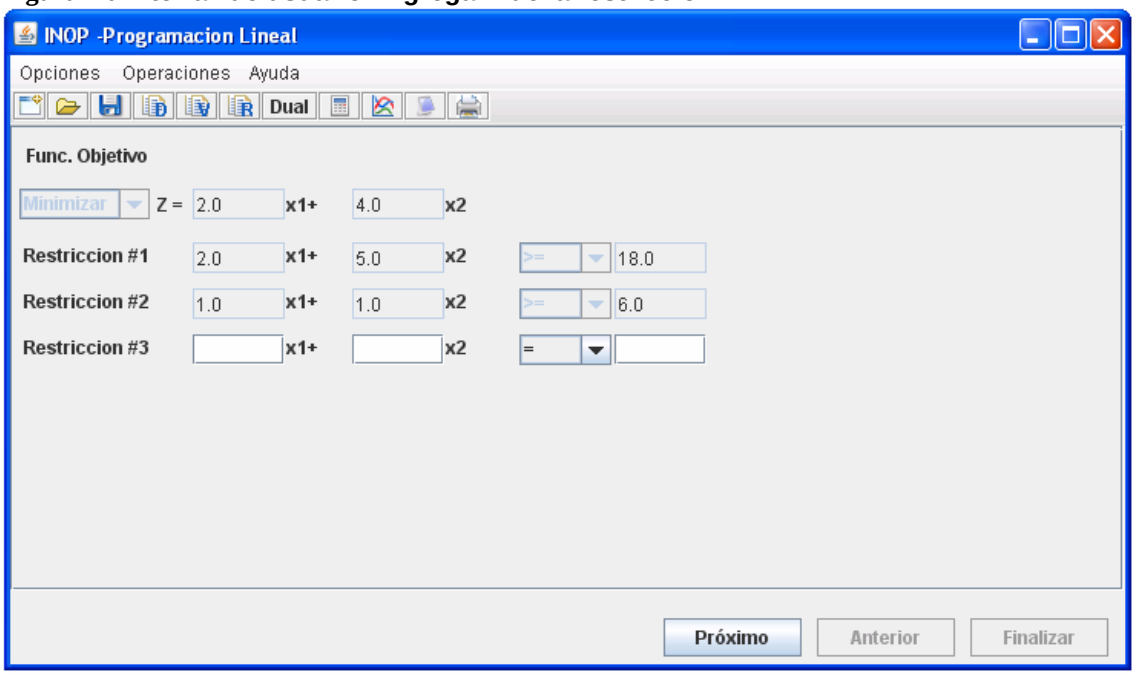


Figura 11 Interfaz de usuario "Cargar Modelo".

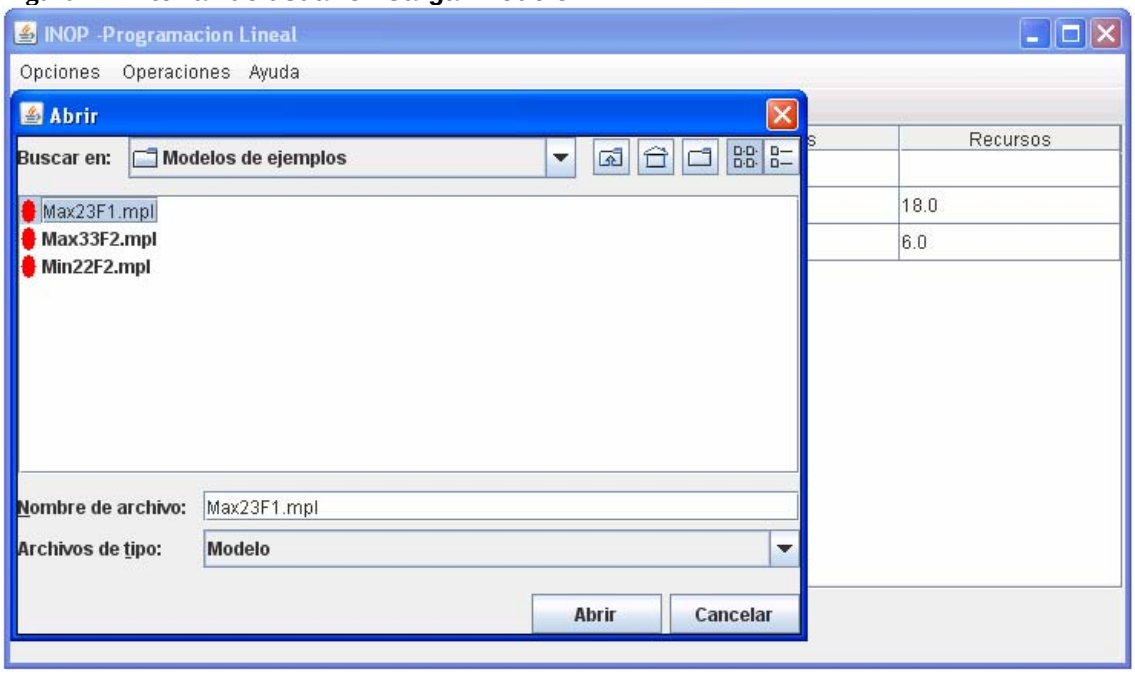
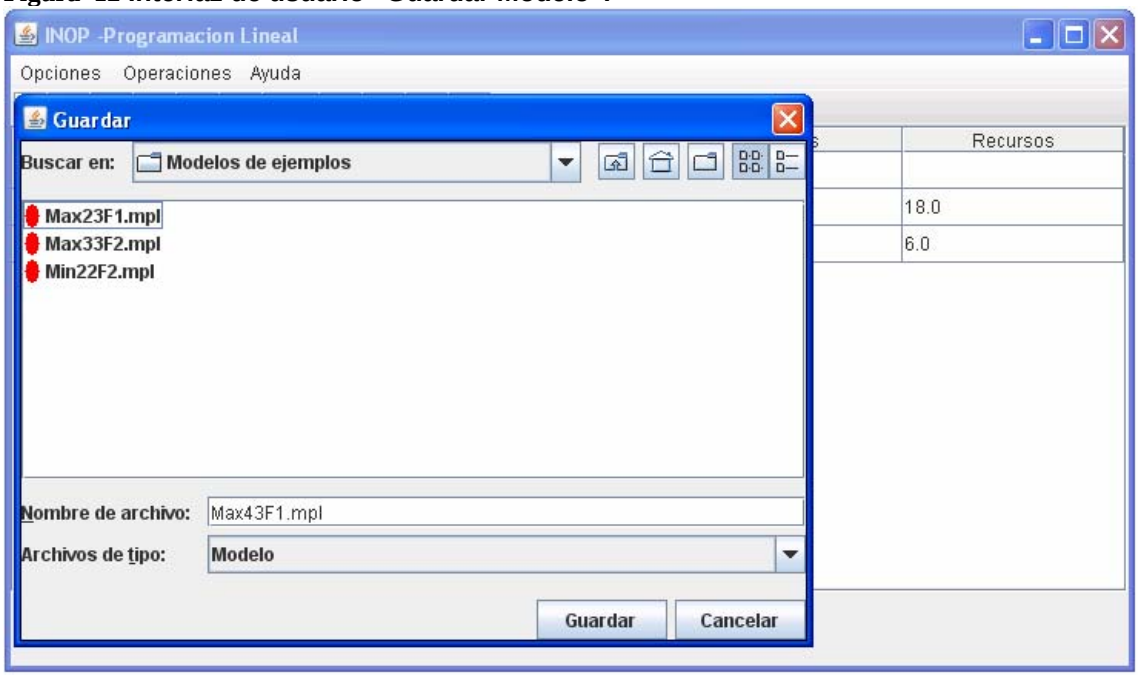


Figura 12 Interfaz de usuario "Guardar Modelo".



3.6.2 Formato de salida de los reportes

Figura 13 Interfaz de usuario "Solución de Modelo".

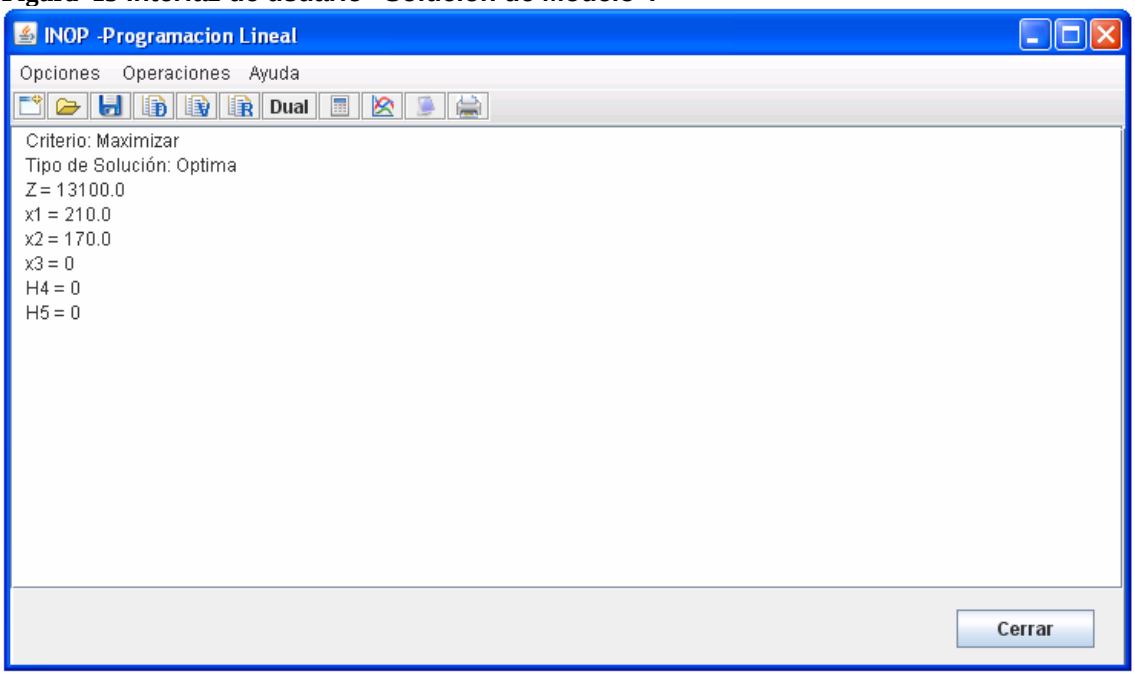
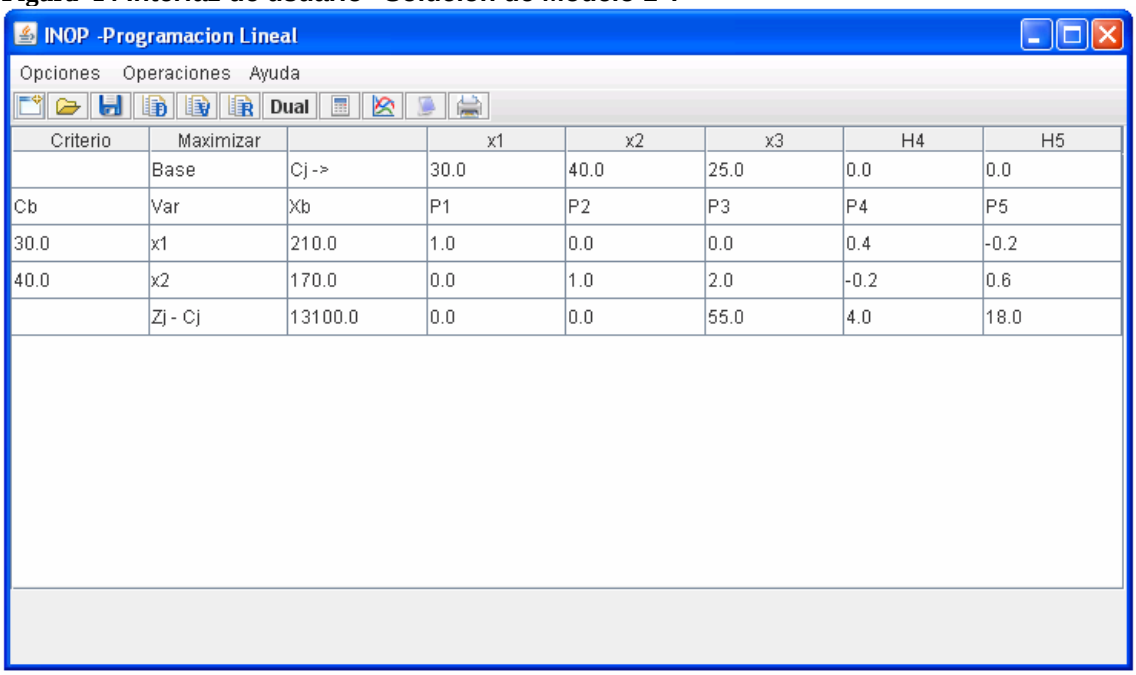


Figura 14 Interfaz de usuario "Solución de Modelo 1".



Criterio	Maximizar		x1	x2	x3	H4	H5
	Base	Cj ->	30.0	40.0	25.0	0.0	0.0
Cb	Var	Xb	P1	P2	P3	P4	P5
30.0	x1	210.0	1.0	0.0	0.0	0.4	-0.2
40.0	x2	170.0	0.0	1.0	2.0	-0.2	0.6
	Zj - Cj	13100.0	0.0	0.0	55.0	4.0	18.0

Figura 15 Interfaz de usuario "Reporte postóptimo".

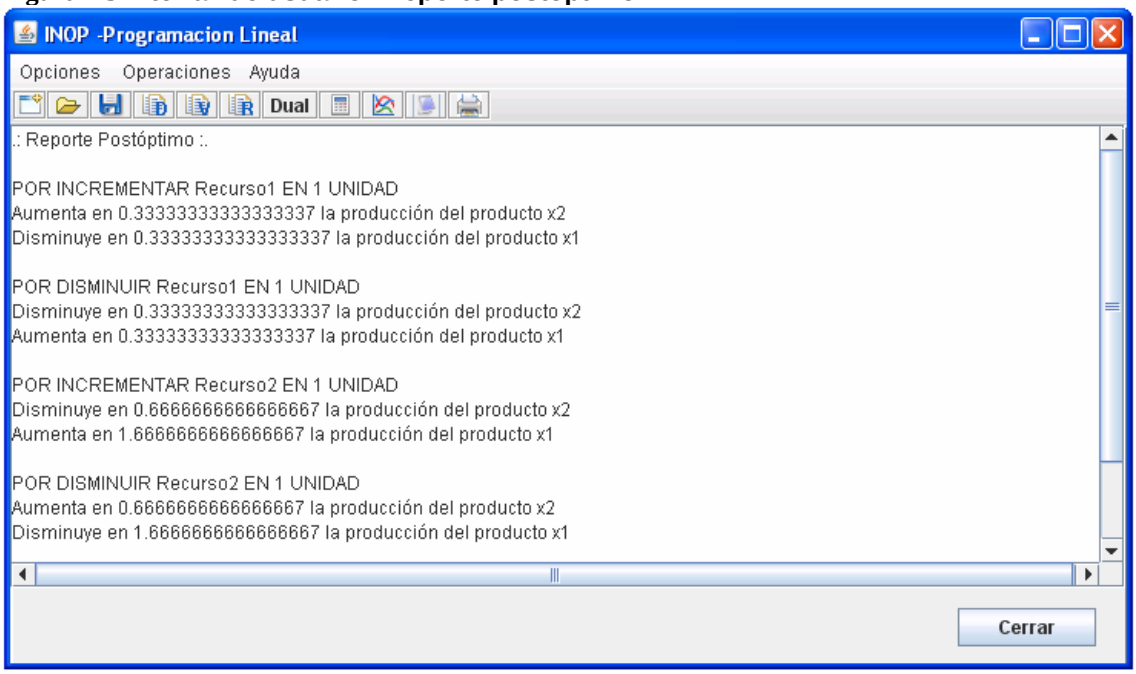
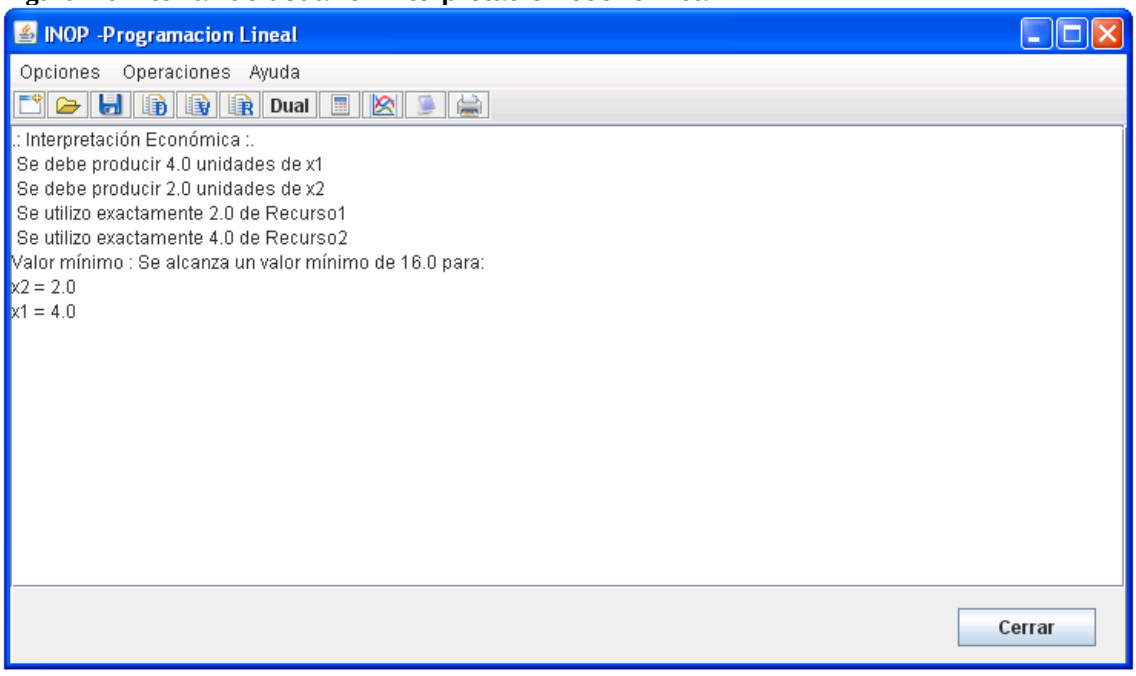


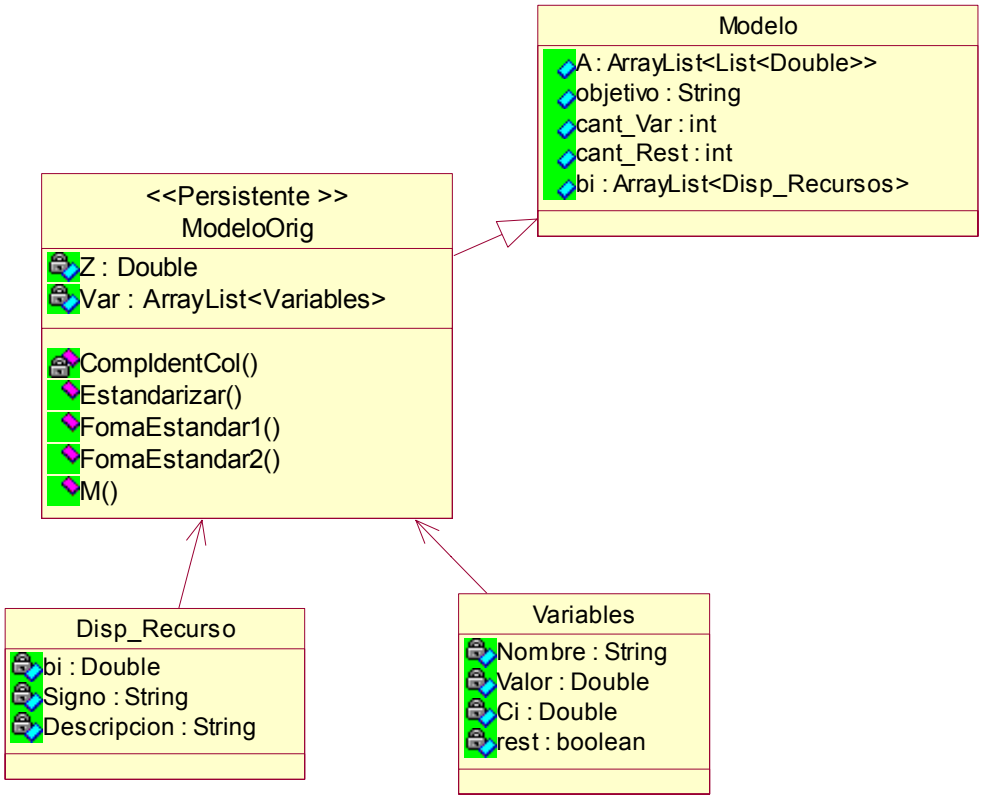
Figura 16 Interfaz de usuario "Interpretación económica".



3.7 Diseño de la base de datos

3.7.1 Modelo lógico de datos

Figura 17 Diagrama de clases persistente



3.8 Diagramas de Secuencia

Figura 18 Diagrama de secuencia "Calcular Dual".

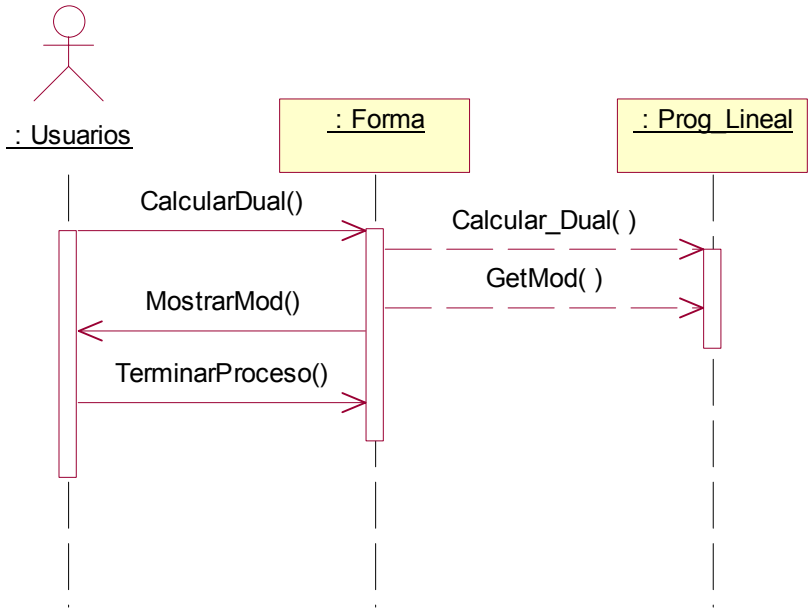


Figura 19 Diagrama de secuencia "Cargar Modelo / Guardar Modelo".

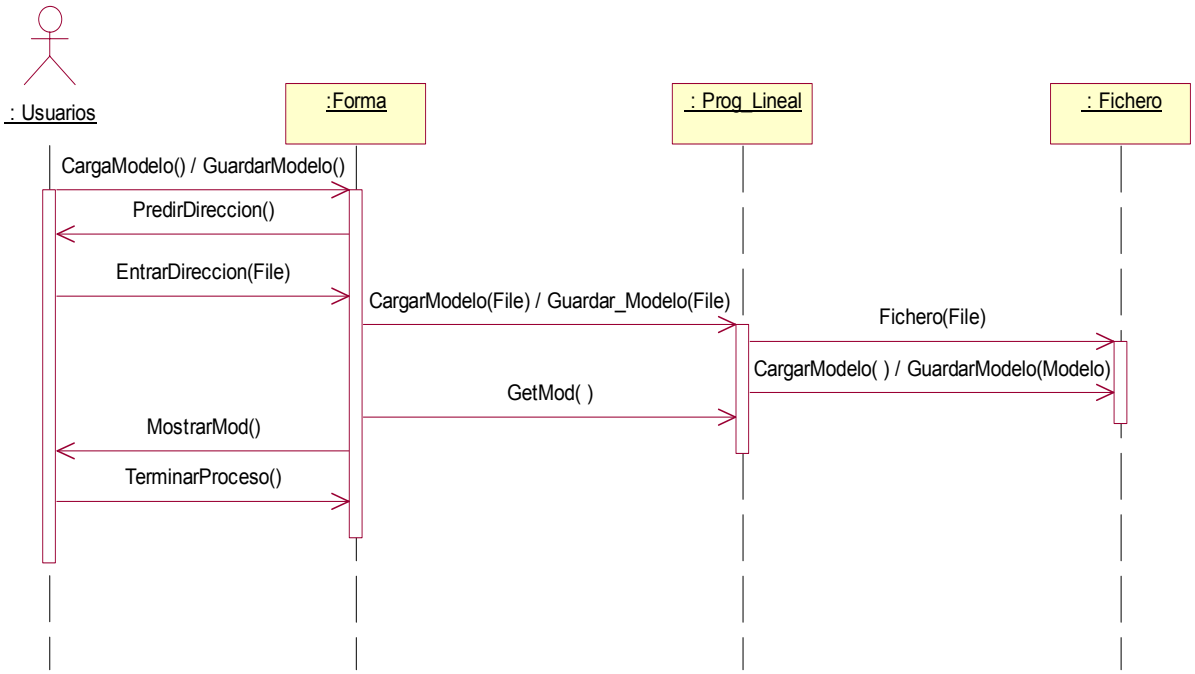


Figura 20 Diagrama de secuencia "Insertar Modelo".

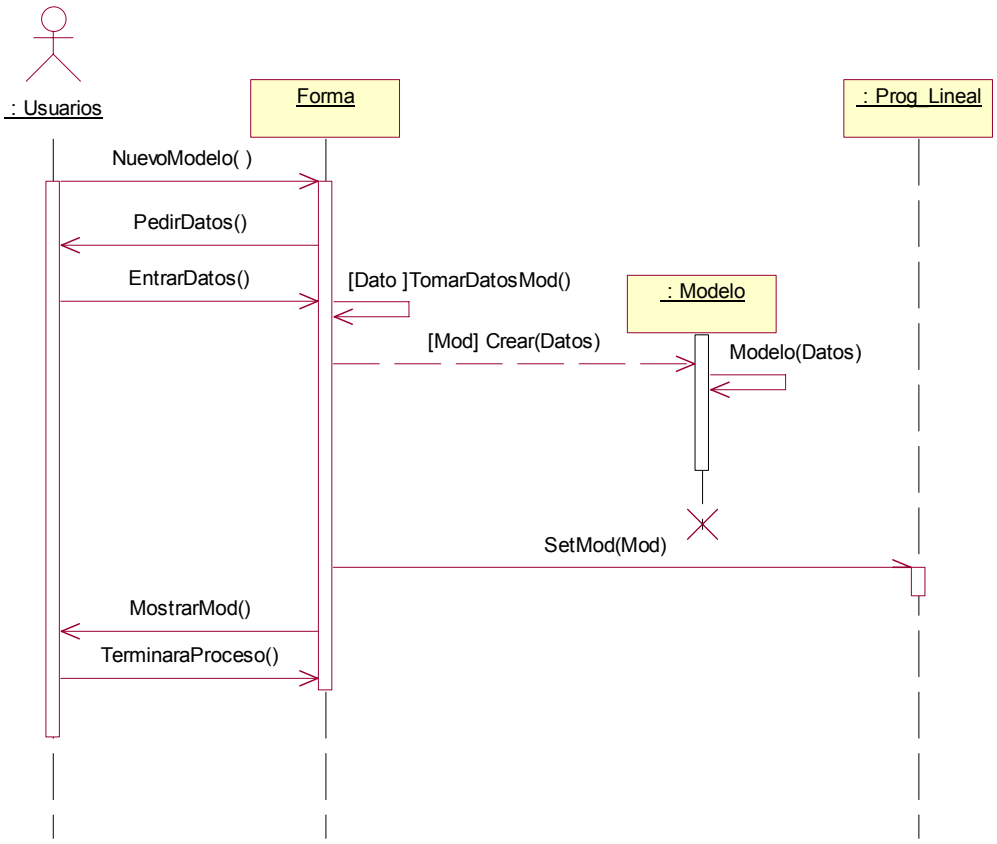


Figura 21 Diagrama de secuencia "Análisis de sensibilidad si se agrega una nueva variable".

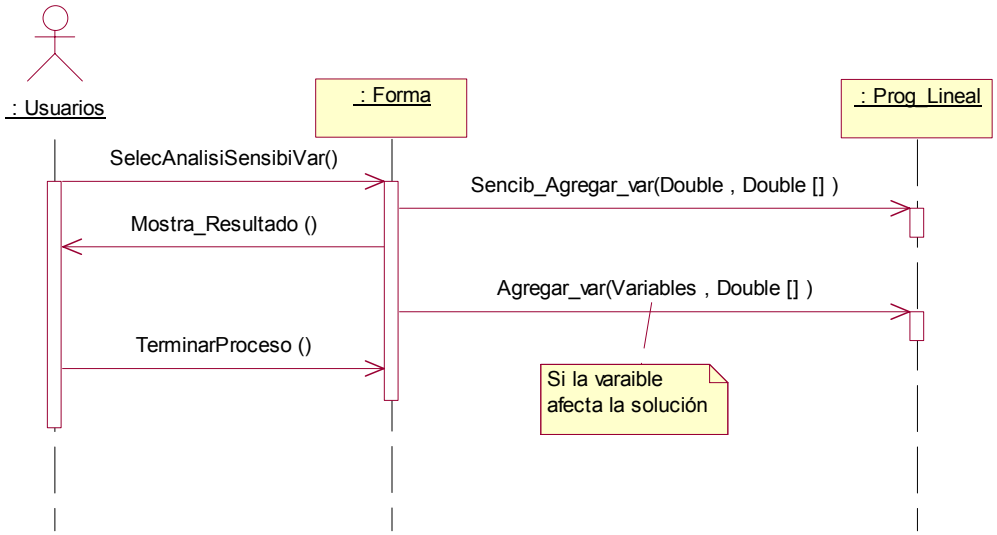


Figura 22 Diagrama de secuencia "Reporte postóptimo".

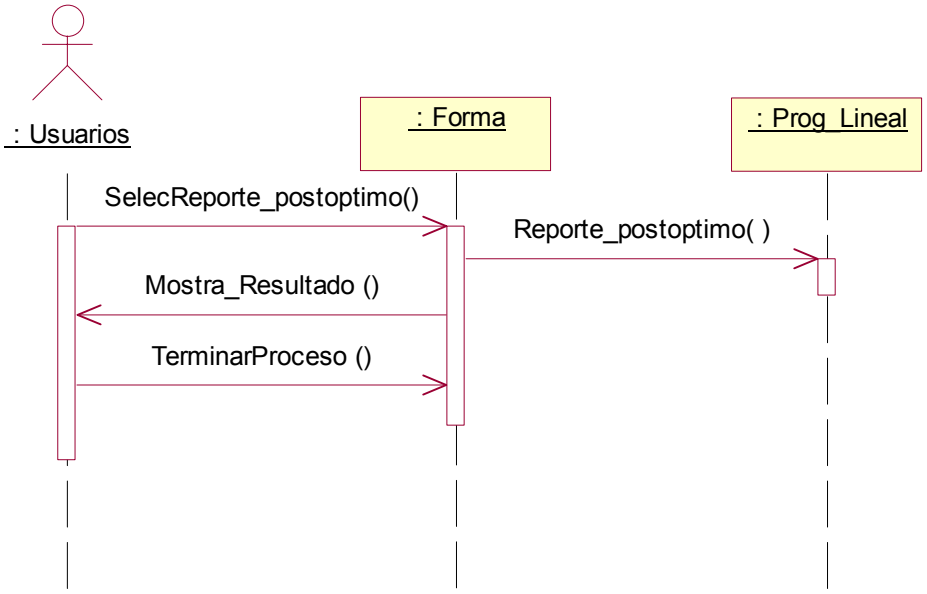


Figura 23 Diagrama de secuencia "Interpretación Económica".

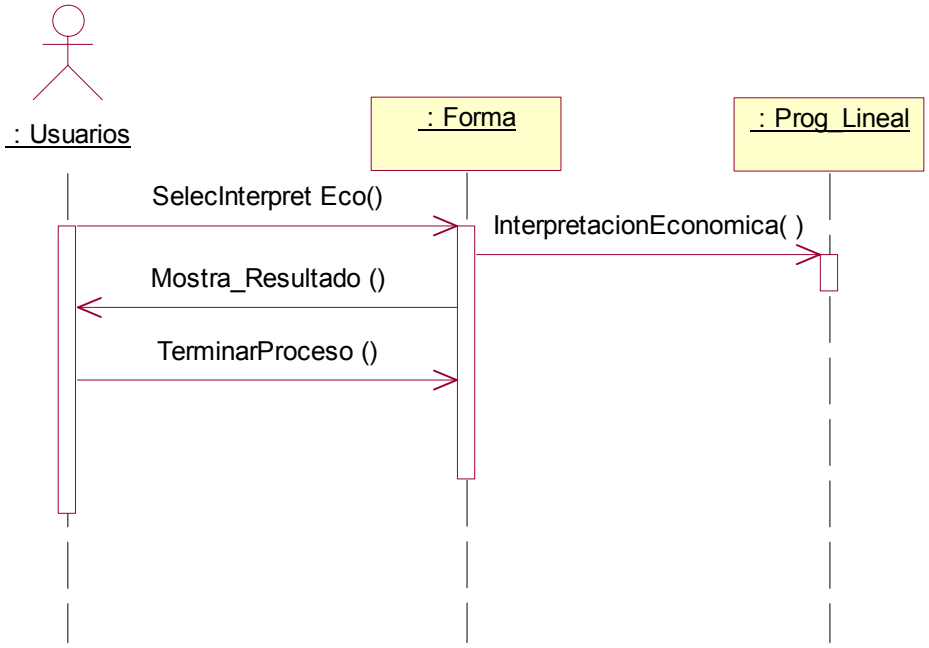


Figura 24 Diagrama de secuencia "Solucionar Modelo".

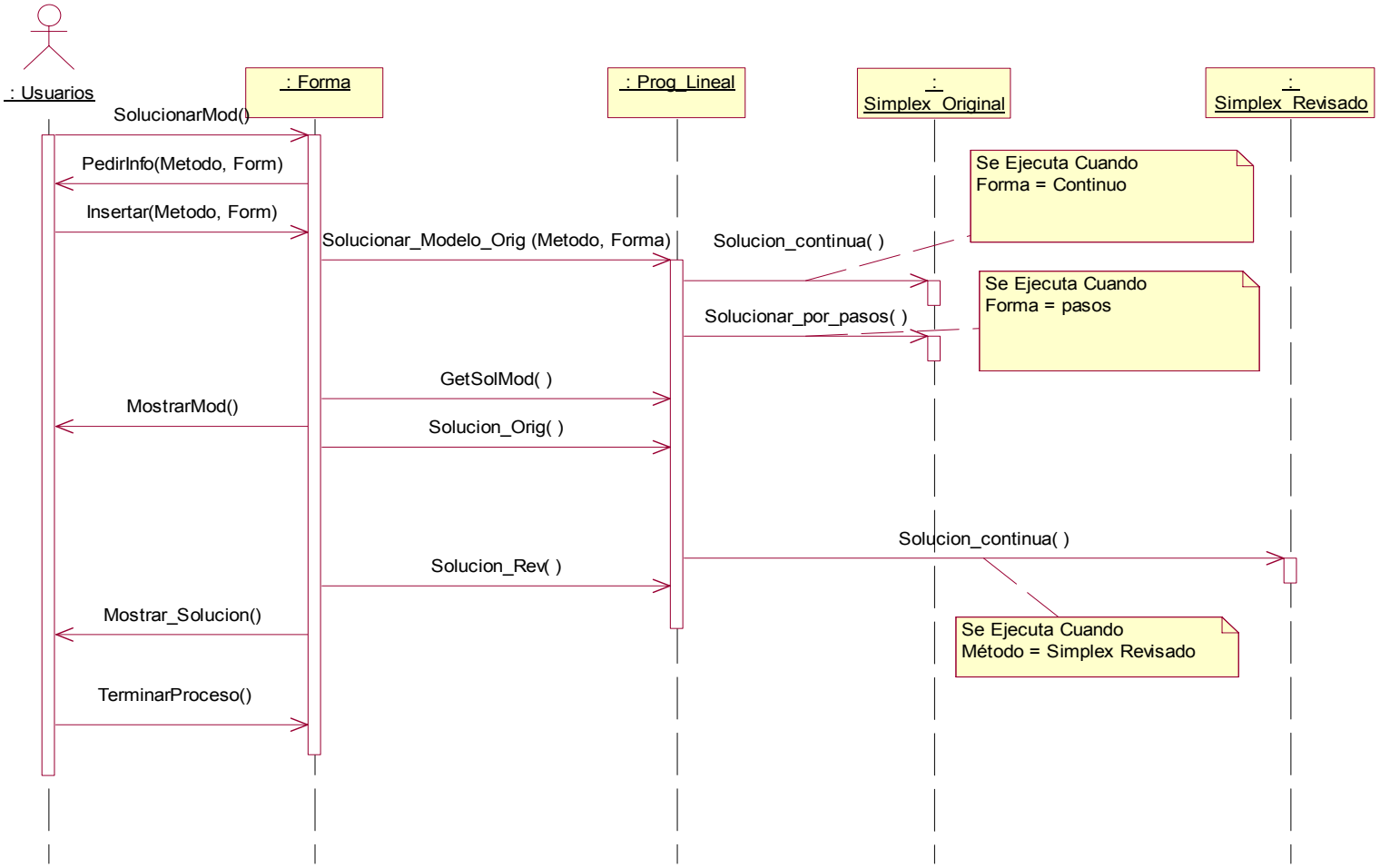


Figura 25 Diagrama de secuencia "Editar Modelo".

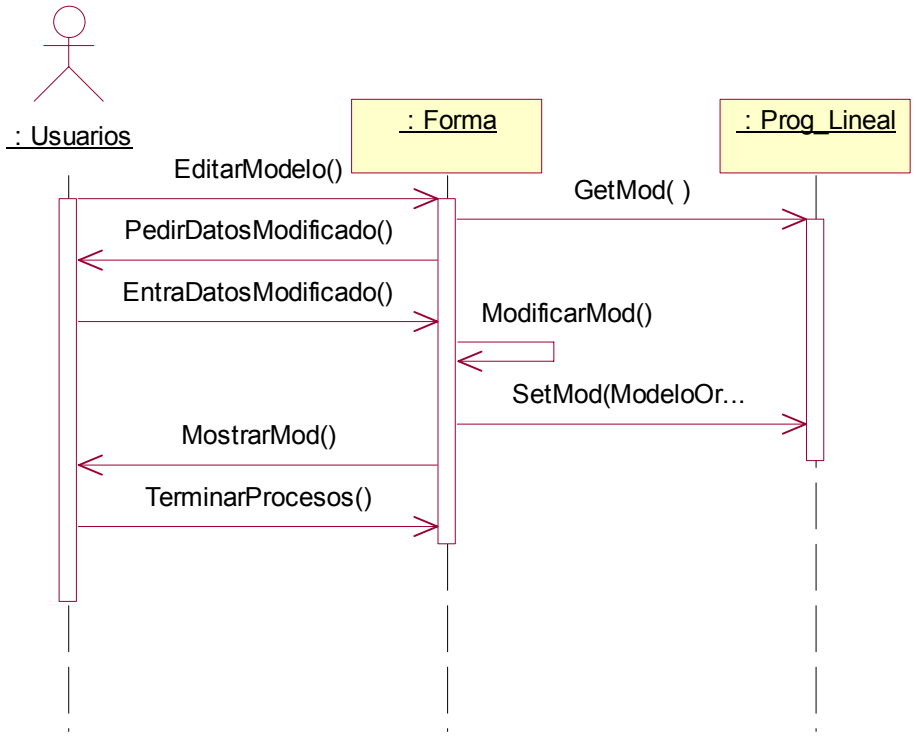
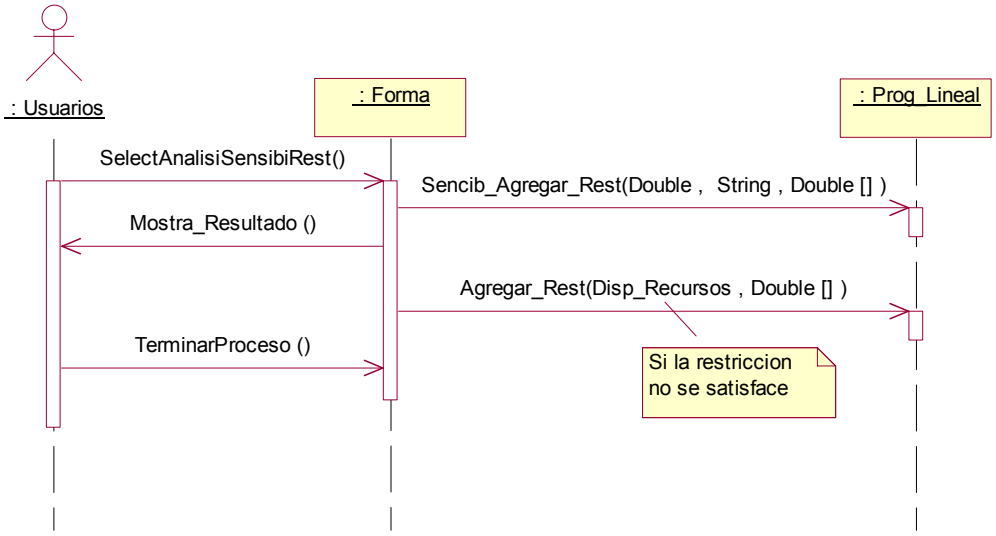
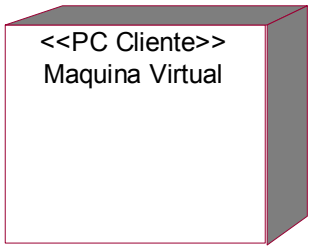


Figura 26 Diagrama de secuencia "Análisis de sensibilidad si se agrega una nueva restricción".



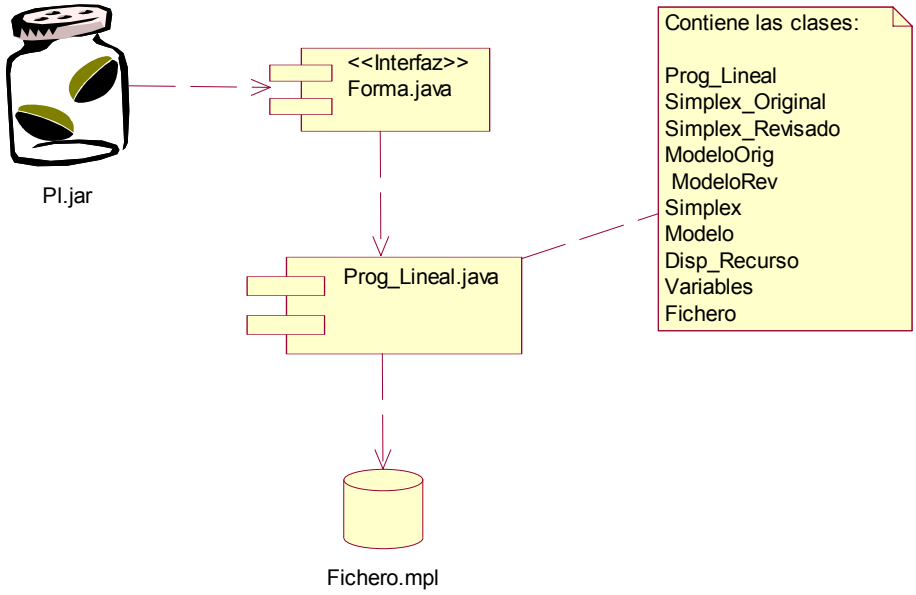
3.9 Diagrama de despliegue

Figura 27 Diagrama de despliegue



3.10 Diagrama de Componentes

Figura 28 Diagrama de componente.



3.11 Conclusiones

En este capítulo se ofrecen aspectos que dentro de la metodología de ingeniería del software empleada, brindan informaciones específicas para el correcto entendimiento de la solución propuesta a partir del análisis. Se definió los actores del sistema, se modeló el Diagrama de Casos de uso el cual representa las funcionalidades del sistema, se definió el Diagrama de Clases que representa la estructura interna del sistema, los diagramas de secuencia con los que se logra delinear el orden de los procesos para la ejecución de cada casos de uso. Se representó la arquitectura física del sistema con el modelo de despliegue, compuesto por un nodo que representa una computadora cliente con la máquina virtual de java y la aplicación propuesta.

Capítulo 4 Estudio de factibilidad

4.1 Introducción.

Para el estudio de factibilidad de este proyecto se utilizará la **Metodología Costo Efectividad (Beneficio),** la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación conjunta de dos factores:

- El costo, que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados
- La efectividad, que se entiende como la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo).

4.2 Efectos Económicos

- Efectos directos
- Efectos indirectos
- Efectos externos
- Intangibles

Efectos directos:

- POSITIVOS:
  - Se permitirá resolver problemas de Programación Lineal medianamente grandes, imposible de hacer a lápiz y papel.
  - Se ganará en velocidad de computo agilizándose estos mediante la ayuda del ordenador mejorando grandemente el tiempo para solucionar un problema.

– Se dejará de utilizar papel y lápiz para resolver los problemas.

➢ NEGATIVOS:

– Para el empleo de la aplicación es imprescindible el uso de un ordenador, aparejado a los gastos que este trae de consumo de corriente eléctrica y mantenimiento.

Efecto indirecto:

– Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de comercializarse.

Externalidades:

– Se contará con una herramienta disponible que apoyará a los administrativos a mejorar la toma de decisiones.

Intangibles:

– En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible.

A fin de medir con precisión los efectos, deberán considerarse tres situaciones:

- **SITUACIÓN SIN PROYECTO**

Para llevar acabo la solución del modelo matemático sin proyecto debemos seguir los siguientes pasos:

1. Construcción del modelo matemático.

En la construcción del modelo se deben seguir los siguientes pasos:

a) Definición de las variables del modelo.

b) Construcción del sistema de restricciones.

c) Planteamiento de la función objetivo.

2. Derivar una solución del modelo.

Para la solución del modelo matemático debemos seguir los siguientes pasos:

a) Estandarizar el modelo, convertir las desigualdades en igualdades.

b) Escribir la tabla inicial para el método simplex.

c) Verificar si estamos en presencia de una solución óptima, comprobando si existe un caso especial de solución.

d) Calcular los valores de Zi – Ci, y seleccionar la variable que entra en la base.

e) Calcular y seleccionar la variable que sale de la solución.

f) Calcular los nuevos coeficientes de la tabla simplex.

Esto se pude realizar de la siguiente manera:

Fila del pivote:

Nueva fila del pivote = (Vieja fila del pivote) / (Pivote)

Resto de las filas:

Nueva fila = (Vieja fila) -(Coeficiente de la vieja fila en la columna de la variable entrante) x (Nueva fila del pivote)

g) Escribir la nueva tabla simplex.

h) Verificar si estamos en presencia de una solución óptima finalizar, en caso contrario pasamos al paso 2, inciso d).

Nota: La solución se hace más trabajosa si el modelo tiene gran cantidad de variables y restricciones.

- **OPTIMIZADA SIN PROYECTO**

  La solución optimizada del modelo matemático sin proyecto cumple estrictamente con lo antes reflejado en la situación sin proyecto, debido a que estos pasos responden a un algoritmo matemático optimizado.

- **SITUACIÓN CON PROYECTO**

Para llevar acabo la solución del modelo matemático con proyecto debemos seguir los siguientes pasos:

1. Construcción del modelo matemático.

En la construcción del modelo se deben seguir los siguientes pasos:

a) Definición de las variables del modelo.
b) Construcción del sistema de restricciones.
c) Planteamiento de la función objetivo.
d) En el menú Opciones, seleccionar “Nuevo Modelo”.
e) Insertar el modelo en el sistema.

2. Solucionar modelo matemático.

a) En el menú Operaciones, seleccionar “Solucionar Modelo”.
b) Seleccionar el método de solución, en casos de ser el método simplex original, seleccionar el tipo de solución.
c) Seleccionar, “Solucionar”.

4.3 Beneficios Y Costos Intangibles en el proyecto

COSTOS:

- Resistencia al cambio.

BENEFICIOS:

- Mejora en la calidad de la información por la integridad, oportunidad de la información y confiabilidad.
- Mayor comodidad de los usuarios
- Mejor imagen de la institución.

4.4 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar Una Ficha De Costo de un Producto Informático [Dra. Ana Ma. Gracia Pérez, UCLV].

Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

Costos en Moneda Libremente Convertible:

- **Costos Directos**.
  1. Compra de equipos de cómputo: No procede.
  2. Alquiler de equipos de cómputo: No procede.
  3. Compra de licencia de Software: No procede.
  4. Depreciación de equipos: $ 60.78.
  5. Materiales directos: No procede.

  Total: $ 60.78.
- **Costos Indirectos**.
  1. Formación del personal que elabora el proyecto: No procede.
  2. Gastos en llamadas telefónicas: No procede.
  3. Gastos para el mantenimiento del centro: No procede.
  4. Know How: No procede.
  5. Gastos en representación: No procede.

  Total: $0.00.
- **Gastos de distribución y venta.**
  1. Participación en ferias o exposiciones: No procede.
  2. Gastos en transportación: No procede.
  3. Compra de materiales de propagandas: No procede.

  Total: $0.00.

Costos en Moneda Nacional:

- **Costos Directos.**
  1. Salario del personal que laborará en el proyecto: $100.00.
  2. El 12% del total de gastos por salarios se dedica a la seguridad social: No procede.
  3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.
  4. Gasto por consumo de energía eléctrica: $ 5.94.

5. Gastos en llamadas telefónicas: No procede.
6. Gastos administrativos: No procede.

- **Costos Indirectos.**

1. Know How: $ 108,75.

Total: $ 214.69.

Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. Dentro de esta metodología la técnica de punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes el análisis se basa exclusivamente en los costos. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en horas empleado para resolver un problema de Programación Lineal y la variable sería la complejidad del problema para lo cual tenemos cinco valores.

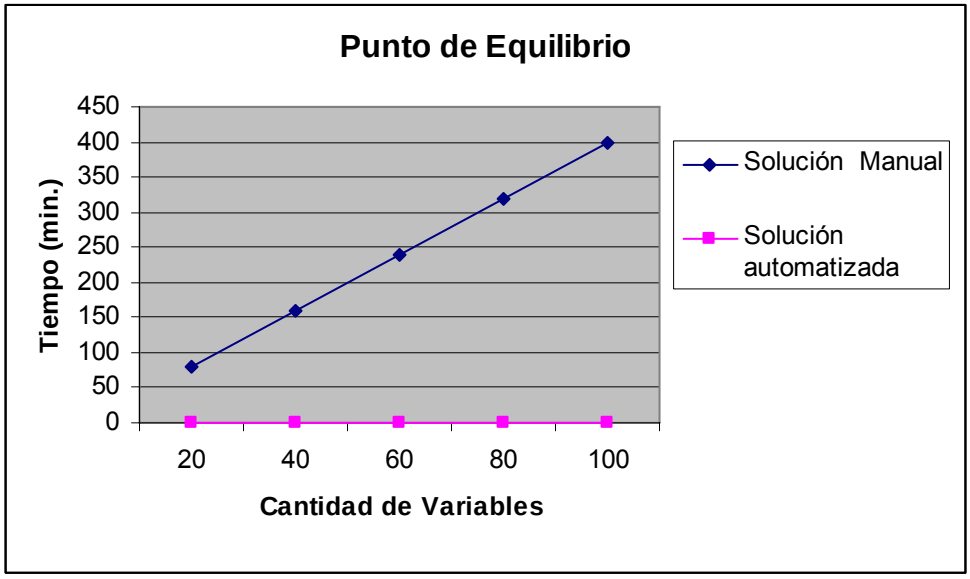
Valores de la variable (Solución manual):

- 1 problemas de hasta 20 variables. (80 min.)
- 1 problemas de hasta 40 variables. (160 min.)
- 1 problemas de hasta 60 variables. (240 min.)
- 1 problemas de hasta 80 variables. (320 min.)
- 1 problemas de hasta 100 variables. (400 min.)

Valores de la variable (Solución con el software):

- 1 problemas de hasta 20 variables. (0.1 min.)
- 1 problemas de hasta 40 variables. (0.2 min.)
- 1 problemas de hasta 60 variables. (0.3 min.)
- 1 problemas de hasta 80 variables. (0.4 min.)
- 1 problemas de hasta 100 variables. (0.5 min.)

Figura 29 Punto de equilibrio.



Teniendo en cuenta los resultados reflejados en la gráfica en cuanto al Punto de Equilibrio queda demostrada la factibilidad del sistema evidenciado por la relación entre la coplejidad del problema (cantidad de variables y restricciones) y el tiempo que demora la solución del mismo de foma manual y automatizada.

4.5 Conclusiones

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizó los efectos económicos, los beneficios y costos intangibles, así como se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 60.78 CUC. y $ 214.69 MN demostrándose la conveniencia de la elaboración del sistema.

Conclusiones

La investigacion realizada parte del problema de la insuficiente integración de las herramientas informáticas relacionadas con la investigación de operaciones disponibles dificultando la formación de los modos de actuación profesional en la toma de decisiones.

Al hacerse un estudio bibliográficos de los referentes teóricos sobre el problema en cuestión y diagnosticar la situación actual que presenta los sistemas informáticos al no facilitar la integración de las herramientas de la investigación de operaciones. Cosideramos necesario la eleaboracion una propuesta de un Sistema informático para la solución de problemas de programación lineal facilitando empotrarse en un sistema integrador que incluya otras ramas de la Investigación de Operaciones como la programación en enteros, programación no lineal, planeación de proyectos, problemas de redes etc.

Por otra parte se cuenta con una herramienta multiplataforma ampliando asi su compatibilidad con otros sistemas operativos.

El sistema fue implementado sobre herramientas patentizadas por licencia GNU GPL, por lo que garantiza su uso sin costos adicionales de licencias.

Constituye un material didáctico y metodológico para los profesores que imparten la disciplina de Investigación de Operaciones.

Recomendaciones

Con el objetivo de perfeccionar y beneficiarnos con el sistema recomendamos:

1. Proponer el sistema a los profesores que imparten la asignatura de investigación de operaciones, como una herramienta más para la enseñanza de la programación lineal.
2. Realizar un estudio más profundo de este sistema en vista a perfeccionarlo en nuevas versiones del software.
3. Crear la ayuda y documentación en línea, para que el usuario cuente con una amplia documentación.
4. Realizar las pruebas concernientes a la compatibilidad de la aplicación con Sistema Operativo que soporten Java.

Referencias bibliográficas

[1]. Schein EH. Process consultation. Cambridge: Addison-Wesley Publishing Company, 1988. pp. 81.

[2]. Orozco Silva E, García Díaz I. Del dato a la decisión: la gestión de información en un sector específico. Caso de estudio BIOTEC. Ciencias de la Información 1992; 23(2):75-82.

[3]. Ponjuán Dante G. Gestión de información en las organizaciones: principios, conceptos y aplicaciones. Santiago de Chile: Cecapi, 1998. pp. XXII.

Bibliografia

1. Charles A. Gallaghher, H.J.W., Métodos cuantitativos para la toma de decisiones en administración Parte 1ra. Felix Varela La Habana 2005 ed. Programación lineal: Solución por el método simplex, pág. 199.

2. Dr. Sc. Ramon Rodríguez Betancourt, C.D.M.A.G., Programación Matemática para Economista Tomo1, 1992, La Habana.

3. Jacobson, G.B., James Rumbaugh., El Proceso Unificado de Desarrollo Software. 1999: Addison Wesley.

4. Lenguaje de programación C en http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C, (17/01/2008).
5. C++ en http://es.wikipedia.org/wiki/C%2B%2B, (17/01/2008).
6. Delphi en http://es.wikipedia.org/wiki/Delphi, (17/01/2008).
7. Lenguaje de programación Java en http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java, (17/01/2008).

Glosario de términos

Modelo primal: Es el modelo matemático conformado por variables de decisión, función objetivo, restricciones y condición de no negatividad.

Modelo dual: Es el modelo matematico obtenido a partir del modelo primal,

Pivote: Hace referencia al elemento que se encuentra en la intercepción entre la columna de la variable que entra en la base y la fila de la variable que sale de la base, en la tabla simplex.

Columna pivote: Columna a la que pertenece el elemento pivote.

Fila pivote: Fila a la que pertenece el elemento pivote.

TIC (Tecnologías de la Información y la Comunicación): Esta expresión engloba el conjunto de tecnologías que conforman la sociedad de la información: informática, Internet, multimedia, etcétera, y los sistemas de telecomunicaciones que permiten su distribución.

Multihilo: Multihilo se refiere a que dos o más tareas se ejecutan "aparentemente" a la vez, dentro de un mismo programa.

Software Libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

GNU GPL: Conjunto de programas desarrollados por la Fundación por el Software Libre; es de uso libre.

Diagrama de clases (class ***diagram):*** diagrama de objetos que describe las clases en forma de esquema, patrón o plantilla, de muchas de las posibles instancias de datos.

UML: Es el lenguaje de modelado de sistemas de software más conocido en la actualidad; es el estándar internacional aprobado por la OMG (Object Managment Group). UML son un grupo de especificaciones de notación orientadas a Objeto, las cuales están compuesta por distintos diagramas, que representan las diferentes etapas del desarrollo de un proyecto de software.