UNIVERSIDAD DE ORIENTE
NÚCLEO DE SUCRE
ESCUELA DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICAS
PROGRAMA DE LA LICENCIATURA EN INFORMÁTICA

SOFTWARE PARA LA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL APLICANDO EL ALGORITMO SIMPLEX (CASO FÁCIL)

(Modalidad: Tesis de Grado)

RENAN ALBERTO SALAZAR SALAZAR

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE LICENCIADO EN INFORMÁTICA

CUMANÁ, 2011

SOFTWARE PARA LA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL APLICANDO EL ALGORITMO SIMPLEX (CASO FÁCIL)

APROBADO POR:

_____

Prof. José Lockiby
Asesor

_____

Prof. Manuel Centeno
Jurado

_____

Prof. Hugo Marcano
Jurado

ÍNDICE

Pág.

u

DEDICATORIA

Este trabajo de investigación está dedicado primeramente a Dios todopoderoso, por brindarme cada día un poco de su sabiduría y por ser la luz que alumbra mi camino, apartando de mí todo mal.

En el mundo terrenal, a mi amada madre Rufina Salazar, por ser el pilar central de mi vida, por su amor incondicional y fe en mí, gracias por darme tu apoyo a lo largo de estos años de mi carrera.

Renan A. Salazar S.

AGRADECIMIENTOS

Le agradezco a Dios todopoderoso, por darme la fuerza cada día para continuar hasta el final. A mi madre Rufina Salazar, por servirme de guía y apoyo en todo momento. Al Núcleo de Sucre de la Universidad de Oriente, por permitirme crecer como profesional. Al Prof. José Lockiby, por haberme asesorado satisfactoriamente a lo largo de toda esta investigación. A las profesoras Nancy Ruiz y Lisbeth Fernández, por dedicar su tiempo en la validación del formato para los cuestionarios. A los profesores Manuel Centeno, Armando Anselmi y Manuel Hamana, cuyas sugerencias hicieron posible la culminación de este proyecto. A todas las personas que se tomaron el tiempo para llenar los cuestionarios, así como al personal que labora en la Coordinación del Programa de la Licenciatura en Informática y en la Sala de Computación, gracias por su valiosa colaboración. Y en general, gracias a todas aquellas personas que de una u otra manera prestaron su ayuda incondicional.

LISTAS DE TABLAS

Pág.

LISTAS DE FIGURAS

Pág.

RESUMEN

Se desarrolló un software, denominado JSimplex, para resolver problemas de Programación Lineal (PL) utilizando el Algoritmo Simplex (AS), para el caso fácil. El mismo se construyó siguiendo las fases y principios de la metodología Programación Extrema, bajo un enfoque ágil e iterativo-incremental. En la fase de exploración, se procedió a recabar toda la información necesaria a través de entrevistas, cuestionarios, redacción de las tarjetas Historias de Usuario (HU), revisión bibliográfica y la observación directa, obteniendo así los requisitos funcionales y las necesidades de los usuarios. Adicionalmente, se creó un plan de liberaciones, el cual permitió definir el total de iteraciones realizadas para cada entrega de código. En la fase de gestión, se estableció un ambiente de trabajo acorde a las necesidades y se seleccionó la arquitectura, tecnologías y herramientas a utilizar a lo largo de todo el proceso de desarrollo. La fase de construcción se dividió en cuatro sub-fases: planificación, diseño, codificación y pruebas. En principio, se seleccionaron las tarjetas HU a implementar dentro de cada iteración, para posteriormente llevar a cabo el proceso de modelado de la aplicación y la creación de prototipos. La codificación fue realizada utilizando herramientas de software libre, entre las cuales destacan: *Ubuntu* 10.04 como sistema operativo GNU/Linux, *Java* como lenguaje para la programación de los módulos de la aplicación y *Netbeans* 6.8 como entorno de desarrollo. Al final de cada liberación de código, se llevaron a cabo las pruebas de usuario y de aceptación, las cuales permitieron evaluar si el requerimiento solicitado había sido cumplido. Para la fase de culminación, se procedió a unir cada uno de los módulos desarrollados en las etapas anteriores y además se elaboró la documentación de la aplicación. El software desarrollado permitirá resolver problemas de PL utilizando el AS, caso fácil, ya sea mediante la escritura de modelos en forma algebraica o a través del uso de tablas simplex. Se empleó la descomposición LU (del inglés *Lower-Upper)* para hallar la inversa de la matriz **B**, para proporcionarle estabilidad numérica a la aplicación y se utilizó la regla de Bland para prevenir el fenómeno de ciclaje. Además, el software cuenta con un módulo interactivo-intruccional orientado a reforzar los conocimientos adquiridos en las aulas de clase, especialmente dirigidos a los estudiantes que se inician en el área de la Investigación de Operaciones. También ofrece las posibilidades de observar las operaciones realizadas por el algoritmo y de consultar definiciones y ejemplos sobre temas relacionados con la PL.

INTRODUCCIÓN

El software se ha convertido en un elemento clave de la evolución de los sistemas y productos basados en computadoras, así como en una de las tecnologías más importantes en el ámbito mundial, evolucionado desde ser una herramienta para la solución de problemas especializados y el análisis de información, hasta convertirse en una industria por sí mismo [1]. Este equipamiento lógico e intangible, necesario para hacer posible la realización de una tarea específica, está relacionado con sistemas de todo tipo: de transporte, administrativos, médicos, militares, industriales, de entretenimientos, entre muchos otros.

En la actualidad, el software de computadora está dirigido a un amplio espectro de tecnologías y áreas de aplicación, entre las cuales se pueden mencionar: los Sistemas de Información, las Telecomunicaciones, Redes, Aplicaciones Multimedia, la Robótica, la Simulación y la Investigación de Operaciones (IO). Esta última disciplina tuvo sus primeras actividades formales en Inglaterra durante la Segunda Guerra Mundial, cuando se encomendó a un equipo de científicos ingleses la toma de decisiones acerca de la mejor utilización de materiales bélicos. Al término de la guerra, las ideas formuladas en operaciones militares fueron adaptadas para mejorar la eficiencia y la productividad en el sector civil. Hoy en día, la IO es una herramienta dominante e indispensable para tomar decisiones [2].

En la IO, se tienen diversas áreas de investigación para resolver una gran cantidad de problemas que surgen en la práctica. Sin duda alguna, una de las principales áreas con que cuenta la IO es la denominada Programación Lineal (PL), la cual trata la planeación de las actividades para obtener el resultado que mejor alcance la meta, entre todas las opciones de solución, es decir, la búsqueda de un valor máximo cuando se trata de beneficios; o bien la búsqueda de un mínimo cuando se trata de esfuerzos a

desarrollar [3]. Para alcanzar sus objetivos, la PL utiliza un tipo especial de modelo, el cual describe el problema que pretende solucionar.

Un modelo de PL es un conjunto de expresiones matemáticas que pueden ser planteados de la siguiente forma:

$$\text{Minimizar } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

sujeto a:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ \vdots \quad\quad \vdots \quad\quad & \quad\quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \end{aligned}$$

$$x_1, x_2, \ldots, x_n \geq 0$$

La función que se desea minimizar $Z = c_1x_1 + x_2x_2 + \ldots + c_nx_n$ se conoce como función objetivo, el vector **c** cuyas componentes son los c_j con $j = 1, 2, 3, \ldots, n$, se denomina vector de costos y el vector de decisión **x** de componentes x_j con $j = 1, 2, 3, \ldots, n$, representan las variables de decisión del modelo. La columna posterior a las desigualdades recibe el nombre de vector **b** o vector del lado derecho y representa la disponibilidad de los recursos. Las primeras m restricciones (aquellas del tipo $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n$) reciben el nombre de restricciones funcionales, pudiendo ser del tipo mayor o igual ($\geq$), menor o igual ($\leq$), igual ($=$) o incluso tener restricciones de varios tipos (si se presentan restricciones sólo del tipo menor o igual ($\leq$) se afirma que el modelo de PL es de caso fácil). Los coeficientes a_{ij}, para $i = 1, 2, 3, \ldots, m$, $j = 1, 2, 3, \ldots, n$ son llamados los coeficientes tecnológicos y forman la matriz de restricciones **A**, y por último, las restricciones $x_j \geq 0$, con $j = 1, 2, 3, \ldots, n$, se denominan restricciones de no negatividad [3]. Dependiendo de todas estas características, el análisis y solución de un modelo de PL debe realizarse según una forma determinada.

Existen dos formas para plantear un modelo de PL que resultan particularmente útiles: la forma canónica y estándar. En la primera, las restricciones son de tipo menor o

igual (≤) y todas las variables son no negativas; mientras que en la segunda, las restricciones están expresadas en forma de igualdad y las variables son del tipo no negativas. Esta última forma, constituye un modo particular para resolver un problema de PL, mediante el uso del Algoritmo Simplex (AS) [4].

El AS, también conocido como Método Simplex (MS), es un procedimiento interactivo propuesto por George Dantzig en el año 1945. Este algoritmo parte de una solución básica factible representada por una matriz **B** y busca en cada iteración una matriz **B'**, igualmente factible pero con más posibilidades de obtener el certificado de optimalidad. La diferencia entre las matrices **B** y **B'** está marcada por una única columna, esto quiere decir que en la matriz **B'** está presente una nueva columna (columna entrante) a cambio de otra que estaba en la columna **B** (columna saliente). En cada iteración del algoritmo se determina qué columna entra y qué columna sale, mediante la realización de cálculos matemáticos preestablecidos. El proceso concluye cuando no es posible mejorar la solución encontrada [4].

El MS tradicional puede ser la mejor opción para entender la lógica fundamental del algoritmo; sin embargo, no es la más conveniente para realizar los cálculos necesarios. Para estos casos, es recomendable utilizar una alternativa al AS denominada la Forma Tabular del MS (FTMS) [2].

La FTMS es una versión más cómoda del MS convencional. La idea es exactamente la misma, pero en cambio su representación se realiza a través de tablas sucesivas (obtenidas mediante pivoteos), donde el cambio entre una tabla y otra representa una iteración del algoritmo. Directamente de la tabla se puede obtener toda la información necesaria para desarrollar el método iterativo del AS [5].

Al momento de utilizar el AS, en su forma tradicional o a través de tablas sucesivas, es necesario realizar una serie de procedimientos adicionales, con el objetivo

de evitar ciertos inconvenientes que se puedan presentar al momento de hallar la solución de un problema lineal. Uno de estos inconvenientes es la repetición de la misma secuencia de solución en forma periódica. Este fenómeno se denomina ciclaje y aunque en teoría es posible que el algoritmo cicle indefinidamente a través de una sucesión de bases, muy rara vez han ocurrido en problemas reales [6].

Diversos métodos son utilizados para prevenir el fenómeno de ciclaje, entre los cuales se encuentran: la regla de Bland, el método lexicográfico y la aplicación del algoritmo de descomposición LU (del inglés *Lower-Upper)* sobre columnas específicas de la tabla. Este último es uno de los más adecuado; ya que le otorga estabilidad (desde el punto de vista computacional) a la matriz **A** de restricciones, debido a ello es ampliamente utilizado en diversos proyectos de desarrollo de software algebraicos impulsados por centros especializados de desarrollo en el área de la IO y universidades alrededor del mundo.

En relación a todo lo anteriormente expuesto, se propuso el desarrollo de un software totalmente funcional, multiplataforma, ajustado a las necesidades de los usuarios, así como confiable, intuitivo, fácil de usar y enfocado a hallar la solución a modelos lineales aplicando el AS (caso fácil), para que de esta manera pueda ser utilizado por los estudiantes y personal docente del área de IO de la Universidad de Oriente (UDO).

El presente trabajo de investigación está estructurado en tres capítulos: en el primer capítulo se expone el planteamiento del problema, en el cual se describe la situación actual de los software, utilizados para resolver problemas lineales, con que cuentan las salas de informática del Núcleo de Sucre de la UDO. Posteriormente, se describe el alcance y las limitaciones del proyecto, donde se detallan las funcionalidades desarrolladas, así como las complicaciones que se presentaron. El segundo capítulo abre con el marco teórico, el cual abarca los antecedentes de la investigación, área de

estudio, área de investigación y las bases teóricas que sustentan este trabajo, además de definir la metodología utilizada a lo largo de todo el proceso de desarrollo. En el tercer capítulo se presenta, de forma clara y detallada, cada paso llevado a cabo durante la construcción de la aplicación. Por último, se señalan las conclusiones y recomendaciones derivadas de esta investigación.

CAPÍTULO I
PRESENTACIÓN

1.1 Planteamiento del problema

En las últimas décadas, la revolución de las computadoras ha hecho posible la extensa aplicación de la PL y su MS. Aunque es posible aplicar el AS a mano para resolver problemas lineales, los cálculos necesarios son demasiado tediosos para llevarlos a cabo de manera rutinaria [7]. Por lo tanto, la capacidad de la computadora electrónica digital para realizar operaciones aritméticas, miles o tal vez millones de veces más rápido que los seres humanos, permitió consolidar el MS como técnica idónea para resolver problemas de PL [8].

Uno de los inconvenientes que se puede presentar, cuando se codifican los pasos del AS en una computadora, es la llamada inestabilidad numérica. Debido a que el MS utiliza el concepto de álgebra de matrices durante sus iteraciones es susceptible a que cualquier error en el procesamiento se magnifique, conforme proceda el cálculo [9]. La descomposición LU es una buena alternativa para solventar el problema anterior, para resolver sistemas de ecuaciones y para encontrar la inversa de una matriz, debido a que sus operaciones se basan en el cálculo numérico, el cual a su vez trata de diseñar métodos para aproximar, de manera eficiente, las soluciones de problemas expresados matemáticamente [10].

Adicionalmente, se hace necesario incorporar reglas especiales para prevenir las consecuencias del fenómeno de ciclaje. Robert Bland, matemático estadounidense, sugirió una regla para prevenir dicho fenómeno, al restringir la elección de la variable de entrada y de salida. En esta regla, las variables se escriben primero en cierto orden, por ejemplo x_1, x_2,…, x_n, sin pérdida de generalidad. Luego, de todas las variables no

básicas con $z_j - c_j > 0$, para entrar a la base se elige la de menor índice. De manera semejante, de todos los candidatos para salir de la base, como variable de salida se elige la de menor índice. A pesar de que gran parte de las aplicaciones de software disponibles para resolver problemas lineales, ignoran alguna regla que garantice la no ocurrencia del ciclaje, su codificación resulta relativamente fácil de implementar y a menudo se hace ventajosa desde el punto de vista computacional [4].

Varios paquetes computacionales enfocados a resolver problemas lineales hacen uso de la FTMS. Esto debido a que permite visualizar las operaciones realizadas de una manera práctica, registrando la información esencial en cada iteración del AS. Por su parte, el Modo Tutorial del Método Simplex (MTMS), el cual consiste en resolver paso a paso el problema, señalando las operaciones a realizar y dejando que la aplicación realice los cálculos, está enfocado en proporcionar una retroalimentación inmediata para probar el conocimiento de los detalles de cálculo en cada procedimiento [2].

El contar con herramientas que no sólo resuelvan problemas lineales a través de la FTMS o del MTMS, sino que además muestren las operaciones intermedias realizadas, representa otra alternativa en pro de la comprensión y el manejo más efectivo del MS, especialmente para aquellas personas que se inicien el área de la IO. Hoy en día, el acceso a este tipo de herramientas se encuentra limitado, debido a que en su mayoría son de tipo privativo (requieren el pago de licencias) y dependen de una determinada plataforma. Por tal motivo, las universidades y centros de desarrollo de software han asumido el compromiso de poner a disposición de los investigadores del área, programas computacionales que no requieran el pago de licencias, sean compatibles con diversas plataformas e incluyan métodos especializados en prevenir el fenómeno de ciclaje.

La Licenciatura en Informática del Núcleo de Sucre de la UDO cuenta con salas de computación que funcionan como centro para la investigación, la generación de aprendizaje y la construcción de nuevos conocimientos. En estas salas, los estudiantes y

profesores poseen un ambiente tecnológicamente adecuado que les permite resolver problemas de diversas índoles, dentro del ámbito académico; sin embargo, los paquetes computacionales utilizados para resolver problema lineales: TORA (versión desarrollada en el año 1992), TORA-Win (versión 1.00 del software TORA, basado en el SO Windows) y LINDO (del inglés *Linear Interactive and Discrete Optimizer*), presentan una serie de debilidades que dificultan su normal funcionamiento. Esta situación se ha hecho evidente debido al continuo avance tecnológico, el cual ha socavado su potencial, poniendo al descubierto sus limitaciones.

Mediante la observación directa fue posible constatar que el software TORA sólo se ejecuta bajo MS-DOS (del inglés *Microsoft Disk Operating System*), no posee módulos de ayuda, carece de una interfaz de usuario intuitiva y se encuentra en el idioma inglés. Todo esto trae como consecuencia, que tanto los estudiantes como el personal docente del área de IO, presenten dificultades al momento de hacer uso de dicha aplicación.

En lo que respecta al programa TORA-Win, a pesar de contar con módulos para la inversión de matrices, solución de ecuaciones de PL, modelos de transporte, entre otros [2], la versión disponible en los laboratorios de la Universidad es de tipo limitada y de uso académico, por esta razón carece de varias funcionalidades y cuenta con poca capacidad de procesamiento, en comparación a su equivalente versión privativa.

El software denominado LINDO, resuelve únicamente problemas de hasta doscientos cincuenta (250) restricciones y quinientas (500) variables. Esto, aunado al hecho de que sólo se ejecuta bajo la plataforma Windows, junto con la imposibilidad de actualización debido al pago de licencias que conlleva, representa una limitación importante que influye en su normal funcionamiento.

Durante las prácticas de laboratorio, los cursantes de la asignatura Programación

Lineal, la cual pertenece al área de Optimización, han manifestado su inconformidad con respecto al desempeño de los paquetes disponibles para resolver los problemas dados en clase. Estas opiniones fueron recogidas, tanto por el profesor como por el preparador de dicha asignatura, quienes también han mostrado su preocupación a raíz de esta situación. Esto ha generado la necesidad de evaluar nuevas alternativas, acorde a los tiempos tecnológicos actuales y que estén encaminadas a facilitar el trabajo de los usuarios.

En vista de que en el país todos los órganos y entes de la administración pública nacional deben usar prioritariamente software libre, de acuerdo con el Decreto Nº 3.390 de la Presidencia de la República, se propuso el desarrollo de una aplicación (utilizando herramientas no privativas) para resolver problemas lineales aplicando el AS (caso fácil), la cual se define como una propuesta viable, por incorporar métodos para la prevención del ciclaje y que proporcionan estabilidad numérica a sus operaciones, con el fin de solventar la situación antes mencionada.

1.2 Alcance y limitaciones

1.2.1 Alcance

El software se desarrolló para ser utilizado, tanto por los estudiantes como por el personal docente del área de Optimización del Núcleo de Sucre de la UDO, con el objetivo de proporcionar una herramienta útil para resolver problemas lineales mediante el uso del AS (caso fácil).

Dentro de las funcionalidades de este software destacan: un módulo interactivo orientado a la ejercitación y práctica del estudiante, que se inicia en el área de la IO, con el objetivo de reforzar los conocimientos que adquiere dentro de las aulas de clase; también permite seleccionar entre dos formas para plantear modelos lineales: forma

algebraica y forma tabular; además, cuenta con una interfaz gráfica usable, adaptada a las tecnologías actuales y acorde a los requerimientos manifestados por los usuarios; permite el acceso a material instruccional con información general sobre definiciones, algoritmos y ejemplos de temas relacionados con la PL; adicionalmente, el entorno de la aplicación fue desarrollada para permitir la incorporación de nuevos módulos enfocados a resolver problemas lineales y así ampliar sus funcionalidades en un futuro; y por último, en relación a la portabilidad, este software puede ejecutarse bajo cualquier sistema operativo (SO) que posea una Máquina Virtual de *Java* (JMV), por sus siglas en inglés.

1.2.2 Limitaciones

Durante el desarrollo de este trabajo se confrontaron limitaciones tales como: retrasos en las primeras etapas de la investigación, como consecuencia al no cumplimiento del plan de liberaciones de código, por parte de algunos miembros del equipo de trabajo. Así mismo, se presentaron contratiempos al momento de recabar requerimientos a través de cuestionarios, debido a que parte de la población a objeto de estudio no disponía del tiempo necesario o fue imposible localizarla. Además, se incorporaron varias librerías de código abierto (codificadas por terceros) a la aplicación, para cumplir con los requisitos solicitados por los usuarios, lo cual ocasionó que el ciclo de iteraciones de desarrollo abarcará más tiempo de lo estimado. Por último, se efectuaron reajustes en los tiempos de entregas para algunos módulos del software, debido a modificaciones en varias partes del código, con el objetivo de hacerlos totalmente compatible con la plataforma Windows.

CAPÍTULO II
MARCO DE REFERENCIA

2.1 Marco teórico

2.1.1 Antecedentes de la investigación

Con el fin de establecer el marco de referencia, se llevó a cabo la búsqueda de fuentes de información. Se consultaron algunos de los proyectos realizados dentro del área de Optimización del Núcleo de Sucre de la UDO, los cuales hacen uso de los procedimientos, técnicas y herramientas que forman parte de la IO y, por tal motivo, están relacionados directamente con el objetivo de la presente investigación.

En año 1997, se realizó una investigación por [11], con el objeto de estudiar la demanda de la harina de pescado, utilizando el modelo de PL para mezclas alimenticias de Tarje Hanser. El programa MPLIMA, construido, analizado y aplicado durante dicho trabajo, permite obtener las cantidades óptimas de los ingredientes usados en la preparación de la mezcla y el precio de la misma, siempre y cuando el problema planteado tenga solución. El investigador concluye, que debido a que muchas de las operaciones realizadas en MPLIMA son efectuadas a través de archivos, se ocupa mucho tiempo computacional en la búsqueda de la solución del problema; sin embargo, si el software es ejecutado bajo ambiente Windows95, el tiempo computacional se reduce considerablemente. El tamaño del problema también influye en el desempeño de la aplicación.

El trabajo presentado por [12], en el año 2001, se basó en el desarrollo de un programa computacional, denominado SIPAV. Este software aplica la técnica de la PL a través del procesamiento de datos de la matriz de costos, para resolver, de manera

eficaz, el problema del agente de ventas. Entre las conclusiones derivadas de esta investigación destaca, que la complejidad el algoritmo utilizado es polinomial; sin embargo, cuando el tamaño del problema se hace mayor, el algoritmo pasa a ser del tipo exponencial. En el año 2006, se desarrolló un programa por [13], denominado MBTGC, el cual hace uso de la metahurística búsqueda tabú con intensificación y diversificación, para la solución del problema de coloreado de grafo por vértices. El investigador sostiene, que MBTGCP muestra un desempeño eficiente, ya que obtuvo buenas soluciones con un tiempo de cómputo razonable; sin embargo, dichas soluciones son eficientes sólo para *n* (cantidad de nodos del grafo) mayor que seis (6), las cuales requieren un mayor tiempo computacional, es decir, conforme se incrementa el valor de *n*, el tiempo de cómputo es mayor. Por tal motivo, se concluye en dicho trabajo, que es necesario ajustar de la mejor manera los parámetros del programa, para evitar tiempos excesivos.

Cada una de estas investigaciones contribuyó significativamente en el buen desarrollo del presente trabajo. Por ejemplo, considerando las conclusiones recabadas, se tomó la previsión de codiciar sólo aquellas instrucciones necesarias, evitando el mal uso de los recursos computacionales disponibles y procurando disminuir, en la medida de lo posible, el tiempo de cómputo necesario para obtener la solución de los modelos planteados a través del software. Todo esto con el objetivo de que el usuario de la aplicación pueda llevar a cabo su trabajo de manera fácil y eficiente.

2.1.2 Antecedentes de la organización

La UDO, creada según el Decreto-Ley No. 459 de fecha 21 de noviembre de 1958, surge como una universidad moderna, con nueva orientación, adaptada a la realidad social, cultural y económica de la comunidad oriental, partiendo de Sucre, “encrucijada de caminos” y expandiéndose hacia Anzoátegui, Monagas, Bolívar y Nueva Esparta [14].

Desde su concepción, la misión del Núcleo de Sucre es la de ser rector de la educación, cultura, ciencia, formación del recurso humano, creación y difusión de conocimientos a través de sus programas de docencia, investigación y extensión, con el propósito de lograr los cambios científicos, tecnológicos y culturales que se requieren para el desarrollo de la región y del país. Paralelamente, posee la visión de consolidarse como una institución universitaria de excelencia en la docencia, investigación y extensión, que responda eficaz y oportunamente a las exigencias de su entorno y a las demandas de cambios e innovaciones que caracterizan la época [14].

El Núcleo de Sucre de la UDO está conformado por cuatro (4) escuelas, entre las cuales se encuentra la Escuela de Ciencias. Ésta agrupa cinco (5) departamentos y dos (2) programas: Departamento de Bioanálisis, Departamento de Matemáticas, Departamento de Física, Departamento de Química, Departamento de Biología y los programas de Informática y Enfermería. Todas abarcan diversos programas para la profesionalización de los bachilleres provenientes de todo el oriente y el resto del país [14].

Entre las carreras que ofrece el Núcleo de Sucre de la UDO se encuentra la Licenciatura en Informática, la cual tiene como misión formar profesionales con calidad competitiva, compromiso ético y responsabilidad social, capaces de analizar necesidades y oportunidades de mejoría en las organizaciones y de proponer estrategias de solución a través de la investigación, el análisis, diseño, mantenimiento y administración de sistemas de información, que potencien su competitividad [15].

Dentro del mismo orden de ideas, el programa de la Licenciatura en Informática del Núcleo de Sucre de la UDO, busca ser un programa educativo de excelencia y prestigio, que participa en los procesos de transformación socioeconómica, a nivel regional, nacional e internacional; a través de la formación de profesionales en el área de la Informática; mediante la investigación, innovación e impulso al desarrollo

productivo y tecnológico [15].

2.1.3 Área de estudio

El área de estudio de este trabajo es la IO. Esta se define como la rama de la matemática que hace uso de modelos matemáticos y algoritmos con el objetivo de ser usados como apoyo a la toma de decisiones. Busca que las soluciones obtenidas sean significativamente eficientes (en tiempo, recursos, beneficios, costos, entre otros) en comparación a aquellas decisiones tomadas en forma intuitiva [3].

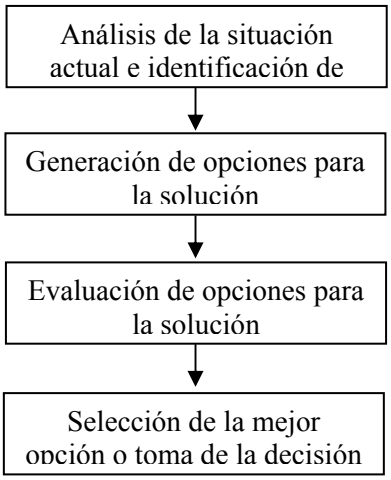
Un elemento principal de la IO son los modelos matemáticos, los cuales representan situaciones del mundo real expresadas mediante ecuación, en otras palabras, son conceptualizaciones abstractas de un problema real a base del uso de letras, número, variables y ecuaciones. Este tipo de modelos son fáciles de manipular y se puede hacer con ellos un gran número de experimentos, además de ser económicos en construir y operar [16]. A pesar de que se deben tener en cuenta factores intangibles o no cuantitativos, la solución de un modelo matemático establece una base para tomar decisiones [2].

La toma de decisiones es el proceso donde se lleva a cabo la selección de una opción o combinación de opciones que resuelven, en cierto sentido, un problema de decisión. Depende de un sin número de factores, entre los que destacan la experiencia del individuo, sus conocimientos, habilidades, actitudes, valores y, en general, su preparación educativa formal. La Figura 1, propuesta por [17], presenta los elementos mínimos que conforman un proceso de toma de decisiones, basado en la ciencia y en el método científico.

Es importante mencionar, que la IO tiene como base el método científico para investigar y ayudar a tomar decisiones sobre los problemas complejos. Este método,

ampliamente utilizado en diferentes áreas de investigación, se define como un procedimiento para descubrir las condiciones en que se presentan sucesos específicos, caracterizado generalmente por ser tentativo, verificable, de razonamiento riguroso y observación empírica [18].

Figura 1. Proceso de toma de decisiones



Además del método científico, la IO cuanta con diversas áreas para resolver modelos matemáticos, dependiendo de sus características, que surgen en la práctica. La programación lineal es una de estas áreas, la cual se diseña para modelos con funciones objetivos y restricciones estrictamente lineales; otra es la programación no lineal, en la que las funciones del modelo son no lineales; la programación entera, en la que las variables toman valores enteros; la programación dinámica, en la que el modelo original se puede descomponer en problemas más pequeños y la programación de red, en la que el problema se puede modelar como una red, entre otras [2].

Una peculiaridad de la mayor parte de las áreas de la IO es que en general las soluciones no se obtienen en forma cerradas, es decir, parecidas a fórmulas. En lugar de ello, se determina mediante algoritmos [2]. Un algoritmo se define como un conjunto

finito de procesos, a su vez finitos y bien definidos, que conducen a un resultado. Por lo general, relacionados con procesos de cálculo matemático [19].

En conclusión, la IO es una ciencia y un arte. Es una ciencia por las técnicas matemáticas que presenta y es una arte, porque el éxito de todos sus procedimientos depende mucho de la creatividad y la experiencia de la persona que las utilice [2].

2.1.4 Área de investigación

Este proyecto se ubica dentro del área de la PL, debido a que el tema central es el desarrollo de un software para la resolución de problemas lineales, utilizando el AS. La PL se define como la técnica de IO encargada del estudio de la optimización (minimización o maximización) de una función lineal, que satisface un conjunto de restricciones lineales de igualdad y/o desigualdad [3].

2.1.4.1 Programación lineal

La PL trabaja con modelos compuestos por tres elementos fundamentales: un conjunto de variables no negativas, que representan a las actividades que van a programarse; una función objetivo, ecuación de primer grado que representa la meta que se desea alcanzar; y un conjunto de funciones lineales, que representan las restricciones del sistema [7]. Este tipo de modelos reciben el nombre de modelos de programación lineal.

Un modelo de programación lineal está en forma estándar, si todas las restricciones son igualdades y todas las variables son no negativas. El AS está diseñado para ser aplicado sólo después de que el modelo se plantea en forma estándar [3]. Se define la forma estándar de un modelo de programación lineal, caso minimización, de la siguiente manera:

$$\text{Minimizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeto a:} \sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1, \ldots, m$$

$$x_j \geq 0 \quad j = 1, \ldots, n$$

En cambio, si todas las variables son no negativas y todas las restricciones son del tipo mayor o igual (≥), se dice que el modelo se encuentra en forma canónica. Se define la forma canónica de un modelo de programación lineal, caso minimización, de la siguiente manera:

$$\text{Minimizar} \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeto a:} \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m$$

$$x_j \geq 0 \quad j = 1, \ldots, n$$

Para obtener la solución a los modelos de PL, se utiliza en AS. Este algoritmo, como procedimiento de resolución de problemas lineales, consiste básicamente en la consideración de sucesivas soluciones factibles básicas, que suponen una mejora en el valor de la función objetivo, hasta obtener la óptima o constatar la existencia de solución ilimitada (o incluso la no existencia de soluciones factibles) [20].

Considere el siguiente modelo lineal:

Minimizar **cx**

sujeto a:

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}$$

donde **A** es una matriz de orden *m* x *n* y **b** es un *m*-vector. A continuación, se presentan

los pasos del AS necesarios para resolverlo:

- Seleccionar una solución básica factible inicial con base **B.**
- Resolver el sistema $\mathbf{Bx_B = b}$ (con solución única $\mathbf{x_B} = \mathbf{B^{-1}\,b} = \bar{\mathbf{b}}$). Sean $\mathbf{x_B} = \bar{\mathbf{b}}$, $\mathbf{x_N} = 0$ y $z = \mathbf{C_B X_B}$**.**
- Resolver el sistema $\mathbf{wB} = \mathbf{c_B}$ (con solución única $\mathbf{w} = \mathbf{c_B B^{-1}}$). Para todas las variables no básicas, calcular $z_j - c_j = \mathbf{wa_j} - c_j$. Sea $z_k - c_k =$ máximo $z_j - c_j$, con $j \in R$, en donde R es el conjunto actual de índices asociados con las variables no básicas. Si $z_k - c_k \leq 0$, entonces el proceso se detiene con la solución básica factible presente como solución óptima. En caso contrario, se continúa con al siguiente paso, con x_k como variable que entra a la base.
- Resolver el sistema $\mathbf{By_k} = \mathbf{a_k}$ (con solución única $\mathbf{y_k = B^{-1}a_k}$). Si $\mathbf{y_k} \leq 0$, entonces el proceso se detiene con la conclusión de que la solución óptima es no acotada.
- Ahora, x_k entra a la base, el índice r de la variable x_{Br}, que sale de la base, se determina mediante el siguiente criterio de la razón mínima.

$$\frac{\bar{b}_r}{y_{rk}} = \text{Mínimo}\left\{\frac{\bar{b}_i}{y_{ik}} \mid y_{ik} > 0, \qquad 1 \leq i \leq m\right\}$$

- Se actualiza la base **B**, donde $\mathbf{a_k}$ reemplaza a $\mathbf{a_{Br}}$, se actualiza el conjunto R de índices y se repite la misma secuencia de pasos [2].

En cada iteración del MS es necesario resolver los siguientes sistemas de ecuaciones lineales: $\mathbf{Bx_B = b}$, $\mathbf{wB} = \mathbf{c_B}$ y $\mathbf{By_k} = \mathbf{a_k}$. Varios procedimientos para resolver y actualizar estos sistemas llevarán a diferentes algoritmos, comprendidos todos dentro del modelo general del MS; sin embargo, es posible representar la solución básica factible con base **B**, utilizando el siguiente *tableau*:

	Z	$\mathbf{x_B}$	$\mathbf{x_N}$	**LD**
Z	1	0	$\mathbf{c_B B^{-1}N - c_N}$	$\mathbf{c_B B^{-1} b}$
$\mathbf{x_B}$	0	**I**	$\mathbf{B^{-1}N}$	$\mathbf{B^{-1}b}$

El anterior *tableau*, no sólo proporciona el valor de la funciones objetivo $\mathbf{c_B B^{-1}b}$, el de las variables $\mathbf{B^{-1}b}$ en el lado derecho y el de los $z_j - c_j$, para las variables básicas, en $\mathbf{c_B B^{-1}N\text{-}c_N}$, sino que también proporciona toda la información necesaria para proceder con el MS [4]. Esta manera particular de plantear un modelo de PL se conoce como la FTMS.

Hoy en día, existe una amplia disponibilidad de paquetes computacionales enfocados a la resolución de problemas lineales, los cuales hacen uso de algunas de las versiones del MS. La mayor parte de estos software de aplicación brindan la posibilidad de hallar la solución a modelos de cientos y hasta miles de variables y/o restricciones, en un tiempo de cómputo razonable y facilitando el análisis posterior de los resultados.

2.1.4.2 Software de aplicación

El software de aplicación consiste en programas independientes que resuelven una necesidad específica. Las aplicaciones en esta área procesan datos cooperando con el usuario en la realización de tareas típicamente humanas, facilitando así la toma de decisiones. Además del procesamiento de datos convencional, el software de aplicación se utiliza para controlar las funciones en tiempo real [1].

Para desarrollar software de aplicación es necesario utilizar algún tipo de lenguaje de programación. Este se define como cualquier lenguaje artificial, utilizado para establecer adecuadamente una secuencia de instrucciones que pueden ser interpretadas y ejecutadas en una computadora. A su vez, los lenguajes de programación se pueden clasificar según su paradigma (modelo que siguen para definir y operar información), entre los que se encuentra el paradigma funcional, lógico, declarativo y orientado a

objetos [21].

Uno de los lenguajes de programación que sigue el paradigma orientado a objetos es *Java*. Este lenguaje es de propósito general y como tal es válido para realizar todo tipo de aplicaciones profesionales. Los programas creados por el compilador de *Java* se pueden ejecutar en una gran variedad de equipos, con diferentes microprocesadores y SO [22].

Los conceptos más utilizados al momento de desarrollar software con *Java* y, en general, en cualquier otro lenguaje de programación que haga uso del paradigma orientado a objetos, se presentan a continuación:

- Objeto: es una cosa, generalmente extraída del vocabulario del espacio del problema o de la solución. Todo objeto tiene un nombre (se le puede identificar), un estado (generalmente hay algunos datos asociados a él) y un comportamiento (se le pueden hacer cosas al objeto) [23].
- Polimorfismo: consigue que un mismo mensaje pueda actuar sobre diferentes tipos de objetos y comportarse de modo distinto. El polimorfismo adquiere su máxima expresión en la extensión de clases, también denominada herencia [24].
- Herencia: es la relación entre clases, mediante la cual una clase hereda toda o parte de la descripción de otra más general. Las instancias heredan todas las propiedades y métodos de la clase a la cual pertenece [23].
- Encapsulamiento: limitación del acceso sin restricciones de un cierto objeto. Los objetos pueden examinar o modificar su propio estado, pero sus métodos de acceso evitan que otros objetos hagan solicitudes o actualizaciones inadecuadas [24].

No hay duda que la Programación Orientada a Objetos (POO) y sus elementos han entrado a formar parte de la corriente principal de la computación, debido a que sus

conceptos son aplicados en una gran variedad de aplicaciones. Es más, el paradigma de objetos se utiliza a escala mundial: en todos los países industrializados se emplean aplicaciones orientadas a objetos e, incluso, muchos países en vías de desarrollo han dado un salto hacia la tecnología orientada a objetos como medio para elevar su base tecnológica [25].

2.1.4.3 Lenguaje unificado de modelado

Para producir software de calidad hay que obtener los requisitos del sistema, contar con una sólida base arquitectónica que sea flexible al cambio, tener un enfoque apropiado y disponer de las herramientas necesarias. De igual manera, es necesario seguir ciertas pautas y no abarcar los problemas de manera somera, con el fin de obtener un modelo que represente suficientemente bien el problema que se ha de abordar [25].

Un modelo es una representación, en cierto medio, de algo en el mismo u otro medio. Capta los aspectos importantes de lo que se modela, desde cierto punto de vista, y simplifica u omite el resto [26].

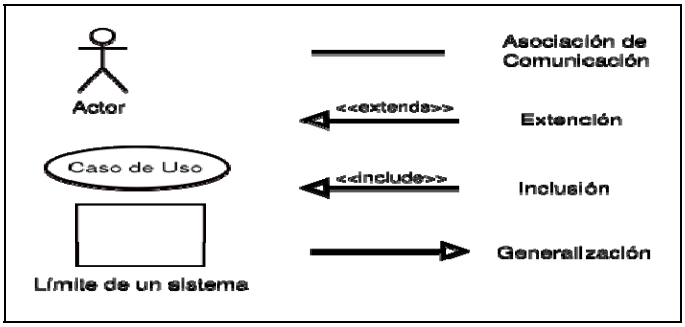
Sin duda alguna, el uso de modelos es la espina dorsal del desarrollo de software de calidad. Se construyen modelos para explicar el comportamiento de un sistema a desarrollar, para comprender mejor ese sistema, para controlar el riesgo y en definitiva para poder atacar problemas que sin el modelado su resolución sería imposible, desde el punto de vista de los desarrolladores y del cliente [25].

El Lenguaje Unificado de Modelado (UML) proporciona, a través de los modelos, una forma estándar de escribir los planos de un sistema. En otras palabras, es un lenguaje gráfico utilizado para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. [25]. Además, está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, los cuales

se agrupan según los aspectos del sistema que modelan. Entre los diagramas que forman parte de UML se encuentran los diagramas de casos de usos, de clases y de secuencias.

Los diagramas de casos de usos describen un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable de valor para un actor. Los diagramas de caso de uso son uno de los tipos de diagramas de UML utilizados para modelar el comportamiento de un sistema, un subsistema o una clase. En la Figura 2, se muestra la simbología utilizada en este tipo de diagramas.

Figura 2. Elementos de un diagrama de casos de uso



La definición de los elementos que forman parte de un diagrama de casos de uso, propuesta por [26], se muestra a continuación:

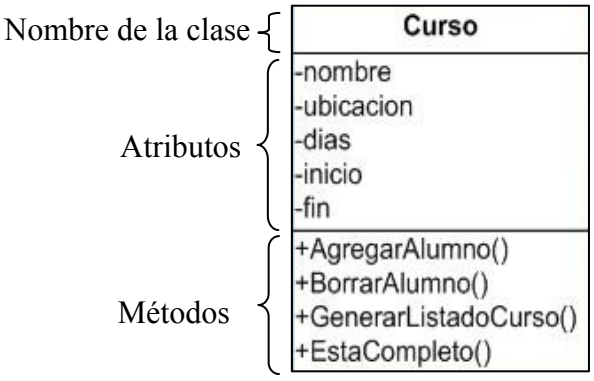
- Actor: idealización de una persona externa, de un proceso o de una cosa que interactúa con un sistema, un subsistema o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema.
- Caso de uso: unidad coherente de funcionalidad, externamente visible, proporcionada por una unidad del sistema y expresada por secuencia de mensajes intercambiados por la unidad del sistema y uno o más actores. El propósito de un caso de uso es definir una pieza de comportamiento coherente, sin revelar la

estructura interna del sistema.

- Límite del sistema: barrera que define el interior y exterior del sistema.
- Asociación de comunicación: línea de comunicación entre un actor y un caso de uso.
- Extensión: inserción de comportamiento adicional en un caso de uso base, que no tiene conocimiento sobre él.
- Inclusión: inserción de comportamiento adicional en un caso de uso, que describe explícitamente la inserción.
- Generalización: una relación entre un caso de uso general y un caso de uso específico, que hereda y añade propiedades a aquél.

Otro de los diagramas que forman parte de UML son los diagramas de clases. Éstos muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Son utilizados para modelar la vista de diseño estática de un sistema. En un diagrama de clases, una clase se representa por un rectángulo en el cual se inscriben tres secciones: en la parte superior se coloca el nombre de la clase; en la intermedia, se representan los atributos que caracterizan a la clase y, en la sección inferior, se listan sus métodos u operaciones [25]. La Figura 3 muestra un ejemplo de la notación del UML, que captura los atributos y métodos de una clase.

Figura 3. Notación UML para representar una clase



Los elementos básicos que forman parte de una clase son: clase, atributo y método, y a continuación, se definen:

- Clase: en la POO, una clase es una colección de objetos que comparten atributos y métodos comunes. Se puede considerar como una plantilla para crear objetos [25].
- Atributo: es una característica concreta de una clase [26].
- Método: algoritmo asociado a un objeto (o a una clase de objetos) constituido generalmente por una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regulan dicha acción y, posiblemente, un valor de salida (o valor de retorno) de algún tipo [25].

Por último, los diagramas de secuencia muestran un conjunto de mensajes, dispuestos en una secuencia temporal. Cada rol en la secuencia se muestra como una línea de vida, es decir, una línea vertical que representa el rol durante cierto plazo de tiempo, con la interacción completa. Los mensajes se muestran como flechas entre las líneas de vida. Su uso es mostrar la secuencia del comportamiento de un caso de uso [26]. En la Figura 4, se ilustra un diagrama de secuencia y sus elementos.

2.2 MARCO METODOLÓGICO

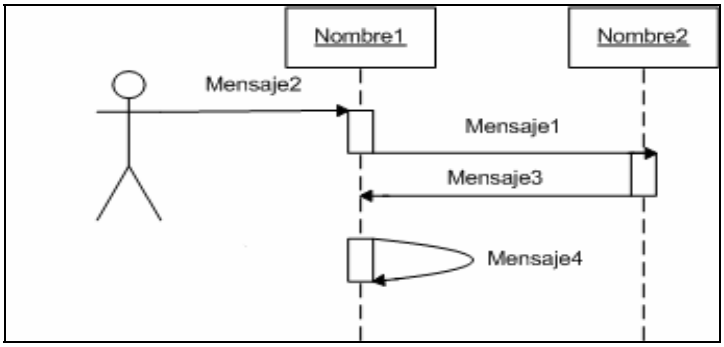
2.2.1 Metodología de la investigación

2.2.1.1 Forma de la investigación

El objetivo del presente proyecto es el desarrollo de un software para la resolución de problemas de PL (caso fácil), principalmente dirigido a los estudiantes de la asignatura Programación Lineal (230-3164). Por lo tanto, esta investigación es de forma aplicada, debido a que se basó en el estudio y en la aplicación de la investigación a

problemas específicos, en circunstancias y características específicas. No se contempla el desarrollo de teorías [19].

Figura 4. Elementos de un diagrama de secuencia



2.2.1.2 Tipo de investigación

La investigación realizada fue de tipo descriptiva, debido a que se basa en la descripción, registro, análisis e interpretación de los procesos enmarcados dentro del desarrollo de cada uno de los módulos de este software. La investigación descriptiva trabaja sobre realidades de hecho y su característica fundamental es la de presentar una interpretación correcta [19].

2.2.1.3 Diseño de la investigación

El diseño de la investigación fue de dos tipos: de campo; porque los datos se tomaron directamente de la realidad; es decir, se recopilaron datos directamente de los sujetos investigados [19], siendo estos los estudiantes y personal docente de la asignatura Programación Lineal (230-3164), sin manipular ni controlar variable alguna; y documental debido a que se recurrió a la utilización de datos secundarios; es decir, obtenidos, elaborados y procesados por terceros.

2.2.1.4 Población y muestra

La población a objeto de estudio fueron los cursantes de las asignaturas del área de IO, para los semestres 2008-I, 2008-II, 2009-I y 2009-II, del Núcleo de Sucre de la UDO. Además, se incluyeron a los profesores que habían dictado la asignatura Programación Lineal (230-3164), para estos semestres. Todas las personas encuestadas poseen conocimientos sobre las técnicas y herramientas utilizadas para resolver problemas lineales. Este requisito era imprescindible, debido a que se buscaba determinar cuáles módulos se iban a desarrollar y qué observaciones y/o recomendaciones se debían tener en cuenta al momento de construir la aplicación. En definitiva, la población para la presente investigación abarcó un total de setenta y tres (73) personas.

La muestra de la población estuvo integrada por un total de cincuenta y dos (52) personas, las cuales fueron consultadas, en su mayoría, dentro de las instalaciones de la Universidad, utilizando diversos instrumentos para la recolección de datos. El objetivo principal fue recolectar la mayor cantidad de opiniones y/o recomendaciones, que permitiesen contribuir en el buen desarrollo de la aplicación.

2.2.1.5 Instrumentos de recolección de datos

A continuación, se exponen las técnicas y herramientas implementadas para este proyecto: revisión bibliográfica y consultas de páginas Web enmarcadas en diversos temas de la PL (definiciones, algoritmos, métodos para resolver modelos, entre otros), lo cual permitió establecer el soporte teórico de la investigación; entrevistas dirigidas a los profesores del área de Optimización del Núcleo de Sucre de la UDO, a fin de obtener información útil que permitiera encaminar y facilitar el proceso de desarrollo; redacción de las tarjetas HU, las cuales sirvieron para capturar los requerimientos funcionales y no funcionales de cada uno de los módulos que conforman el software;

cuestionarios presentados a los estudiantes de la asignatura Programación Lineal (230-3164), con el objetivo de conocer las necesidades de los usuarios; y por último, reuniones programadas con el cliente, en las cuales se evaluaron todas las alternativas disponibles al momento de codificar los requisitos capturados en papel. Todo esto permitió tener una perspectiva clara sobre las necesidades a satisfacer.

2.2.2 Metodología del área aplicada

En el año 2001, Kent Beck y otros 16 notables desarrolladores, escritores y consultores (conocidos como la "Alianza Ágil") firmaron el "Manifiesto para el desarrollo ágil de software", el cual establece:

> Hemos descubierto mejores formas de desarrollar software al construirlo por nuestra cuenta y ayudar a otros a hacerlo. Por medio de este trabajo hemos llegado a valorar: a los individuos y sus interacciones sobre los procesos y las herramientas; al software en funcionamiento sobre la documentación extensa; a la colaboración del cliente sobre la negociación del contrato; y a la respuesta al cambio sobre el seguimiento de un plan [1].

Tomando en consideración los principios previamente citados, se procedió a dar inicio a un proceso de desarrollo ágil de software. La comunicación continua con el cliente y usuarios fue de suma importancia, ya que permitió conocer las características y funcionalidades requeridas para la aplicación; se priorizó la entrega temprana de los módulos sobre una documentación detallada, pero sin desaprovechar las ventajas que ofrece UML como técnica para el modelado de software; se enfrentó con coraje los cambios en los requerimientos, los cuales se presentaron durante todo el proceso de construcción del software; por último, se ajustaron las fechas de liberación de código en algunas iteraciones de desarrollo, para cumplir con los nuevos requerimientos solicitados.

A lo largo de todo el proceso de esta investigación, se utilizó La metodología Programación Extrema (XP), por sus siglas en inglés, como guía para la construcción de la aplicación. Sin duda alguna, XP es uno de los métodos ágiles más populares, ampliamente aceptado y utilizado por desarrolladores alrededor del mundo. Se define como una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito de los proyectos, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios [27]. En el Anexo 1, se muestra un resumen de los valores que se deben tomar en cuenta al momento de aplicar esta metodología. A continuación, se exponen las fases de la metodología XP.

2.2.2.1 Exploración

La actividad de exploración comenzó redactando varias tarjetas HU, las cuales fueron usadas para capturar los requerimientos del software. Una vez escritas las tarjetas, se efectuaron una serie de reuniones entre el equipo de desarrollo y el cliente, para establecer un plan de liberaciones y definir cuáles tarjetas HU conformaban cada incremento. Dada la naturaleza cambiante de los requerimientos de gran parte de los proyectos de desarrollo de software y siguiendo las recomendaciones dadas por XP, se llevó a cabo la redacción de las tarjetas HU en paralelo a todas las fases del ciclo de construcción de la aplicación.

2.2.2.2 Gestión

Durante esta fase se estableció un ambiente de trabajo adecuado para el equipo de desarrollo. Esto permitió sentar las bases para dar inicio al ciclo de programación en pares y, al mismo tiempo, motivó a todas las personas involucradas a trabajar en forma

constante, para así alcanzar los objetivos propuestos. Una vez establecido el ambiente de trabajo, se procedió a seleccionar la arquitectura, tecnología, herramientas y prácticas utilizadas durante todo el proceso de desarrollo del software.

2.2.2.3 Construcción

Está compuesta por cuatro sub-fases: planificación, diseño, codificación y pruebas, que juntas conforman ciclos de desarrollo más cortos, lo cual es una de las ideas centrales de XP. Se realizaron un total de nueve (9) iteraciones durante esta fase, las cuales permitieron incluir las funcionalidades capturadas durante todo el ciclo de desarrollo de la aplicación.

2.2.2.3.1 Planificación

Esta actividad abarcó la redacción y/o recopilación de un conjunto de tarjetas HU, las cuales señalan las características y la funcionalidad requeridas para cada incremento efectuado. En el Anexo 2, se muestra el proceso de planificación propuesto por [28] y llevado a cabo durante esta fase.

2.2.2.3.2 Diseño

Se siguió el paradigma de diseño denominado KISS (del inglés *Keep It Small and Simple*, "Mantenlo tan sencillo como sea posible"), propuesto por XP, el cual ofreció una guía para la implementación de cada tarjeta HU. Adicionalmente, se construyeron una serie de prototipos operacionales. Estas versiones preliminares de la aplicación permitieron clarificar ciertos requerimientos difíciles de diseñar. Por último, se llevó a cabo una refactorización de código, la cual se define como el proceso de cambiar un sistema de software de tal manera que no altere el comportamiento externo del código y que mejore su estructura interna [30].

Cabe resaltar, que fue incluida una fase de modelado bajo UML durante esta etapa de desarrollo. Esto como el fin de facilitar el proceso de visualización y especificación de las funciones de la aplicación, antes de su construcción, y al mismo tiempo, contribuir al proceso de documentación para revisaciones futuras.

2.2.2.3.3 Codificación

XP sugiere que después de redactar las tarjetas HU y realizar el trabajo de diseño preliminar, el equipo de trabajo no debe moverse hacia la codificación, sino que debe desarrollar un conjunto de pruebas que ejerciten cada una de las HU a incluirse en el lanzamiento actual [30]. Por tal motivo, se escribieron un conjunto de pruebas de unidad, las cuales permitieron establecer algunos de fallos que se pudiera presentar durante la ejecución de la aplicación.

Un concepto clave durante la actividad de codificación es la programación en pares. XP recomienda que dos personas trabajen juntas en una estación de trabajo de computadora, para crear código de una tarjeta HU. Esto proporciona un mecanismo para la resolución de problemas en tiempo real y el aseguramiento de la calidad en las mismas condiciones [30]. A pesar de ello, no fue posible cumplir con dicha práctica, debido a que aunque se conformó un equipo trabajo, con el objetivo de construir los diferentes módulos de la aplicación, no fue posible concretar dicha idea, lo cual impidió llevar a cabo las prácticas enfocadas al trabajo colectivo de un grupo de desarrollo.

2.2.2.3.4 Pruebas

Se llevaron a cabo dos tipos de pruebas: las pruebas de unidad y las pruebas de aceptación. La primera fue implementada por el desarrollador, permitiendo comprobar si se alcanzaba la operatividad necesaria para la versión definitiva del incremento y la segunda fue especificada junto al cliente, tomando en consideración las características

generales que se deseaban, la funcionalidad del sistema y los elementos visuales.

2.2.2.4 Culminación

Se dio comienzo a esta fase una vez que todas las HU fueron implementadas. Entonces, el proceso de integración de cada uno de los módulos desarrollados se inició. En esta fase no se hicieron grandes cambios en la arquitectura, el diseño, ni en el código del software, en cambio se tomaron en cuenta otros aspectos también importantes como lo son la interfaz general y la interacción entre el software y el usuario final. Por último, se procedió a realizar toda la documentación del software. La Figura5, propuesta en [28], resume el proceso de desarrollo de software bajo la metodología XP.

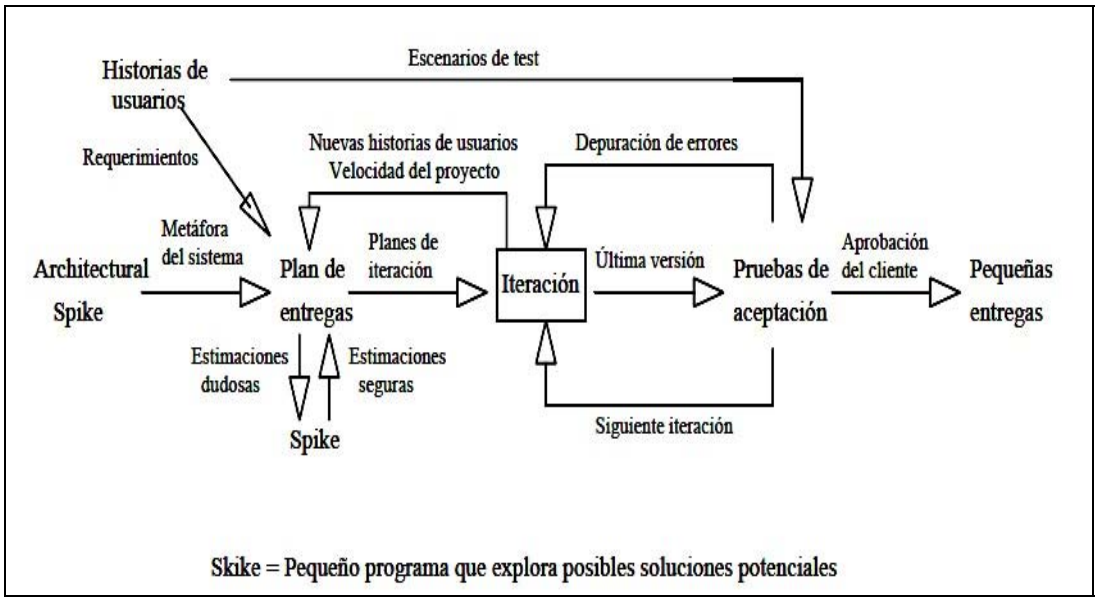


Figura 5. Proceso de desarrollo de software con XP

CAPÍTULO III
DESARROLLO

3.1 Fase de exploración

3.1.1 Revisión de materiales bibliográficos y publicaciones en Internet

La fase de exploración comenzó con una revisión bibliográfica de libros, documentos y publicaciones disponibles en la Web que contemplaban temas relacionados con la PL. El objetivo fue establecer la terminología a utilizar a lo largo de todo el proceso de desarrollo e identificar las técnicas, algoritmos y procedimientos utilizados para resolver problemas lineales.

3.1.2 Aplicación de entrevistas y cuestionarios

El objetivo de esta actividad fue analizar el contexto en el cual se iba a construir el software. Para ello se entró en contacto con la población a la que está dirigida la aplicación.

Por medio de entrevistas realizadas a profesores del área de Optimización del Núcleo de Sucre de la UDO, para el semestre 2009-II, se logró: definir formalmente la problemática existente, proponer posibles soluciones, identificar las limitaciones del proyecto y conocer las necesidades a satisfacer. Todo esto ayudó a sentar las bases para la construcción de un software enfocado a resolver problemas lineales utilizando el AS (caso fácil).

Adicionalmente, se llevó a cabo el proceso de redacción y posterior aplicación de cuestionarios, orientados a los estudiantes de la asignatura Programación Lineal (230-

3164), para el semestre 2009-II, con el objetivo de conocer su punto de vista en relación a los paquetes computaciones que utilizan para resolver problemas lineales. En el Apéndice A, se muestran los formatos utilizados, tanto para las entrevistas dirigidas a los profesores como para los cuestionarios aplicados a los estudiantes.

Con el objetivo de obtener resultados confiables fue necesaria la validación de las preguntas para las entrevistas y cuestionarios efectuados. Para ello, se consultaron a tres (3) profesores del Núcleo de Sucre, los cuales avalaron que estas herramientas cumplían con los parámetros suficientes para ser utilizado en la presente investigación. En el Apéndice B, se muestran las cartas que certifican la validez del formato utilizado para la captura de los requerimientos.

Una vez finalizado el proceso de consultas, se procedió a analizar cada una de las respuestas obtenidas. Los resultados de este análisis son presentados a continuación:

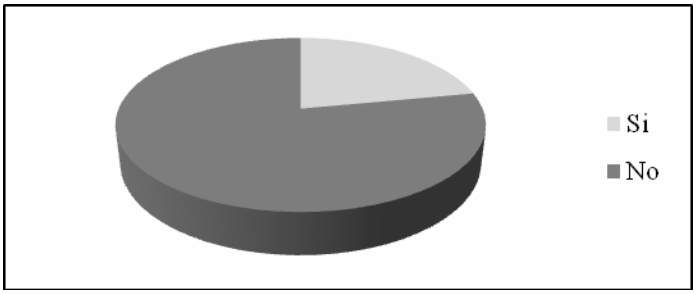
En el Cuadro Nº 5, se muestra la pregunta número 1 del cuestionario junto con los resultados obtenidos.

CUADRO NÚMERO 1 Resultados obtenidos de la pregunta número 1 del cuestionario.

Nº	**Pregunta**	**Alternativas**	**Frecuencia**	**Porcentaje**
1	¿Cursa Ud. actualmente alguna asignatura del área de la Investigación de Operaciones?	Si	11	22%
		No	39	78%

La Figura 6 muestra, en forma gráfica, los resultados obtenidos para la pregunta número 1 del cuestionario.

Figura 6. Resultados gráficos obtenidos de la pregunta número 1 del cuestionario



En la respuesta número 1, representada por la Figura 6, el 22% de las personas encuestadas estaban cursando alguna asignatura del área de la IO, para el semestre 2009-II; mientras que un 78% no lo hacían. Es importante mencionar, que las personas que respondieron afirmativamente a esta pregunta, representan a todos los estudiantes de la asignatura Programación Lineal (230-3164), para dicho semestre académico.

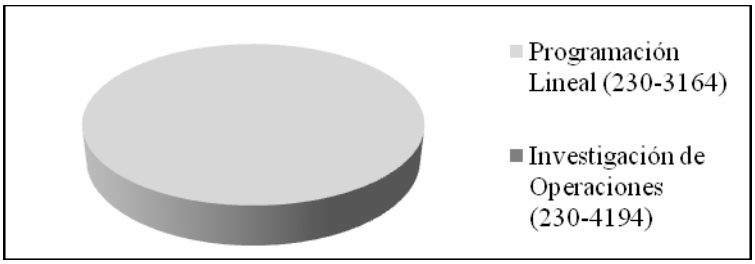
En el Cuadro Nº 2, se presentan los resultados obtenidos para la pregunta número 2.

CUADRO NÚMERO 2 Resultados obtenidos de la pregunta número 2 del cuestionario.

Nº	**Pregunta**	**Alternativas**	**Frecuencia**	**Porcentaje**
2	De ser afirmativa la respuesta anterior, ¿cuál asignatura está cursando?	Programación Lineal (230-3164)	11	100%
		Investigación de Operaciones (230-4194)	0	0%

Los resultados gráficos obtenidos de la pregunta número 2, se presentan en la Figura 7.

Figura 7. Resultados gráficos obtenidos de la pregunta número 2 del cuestionario

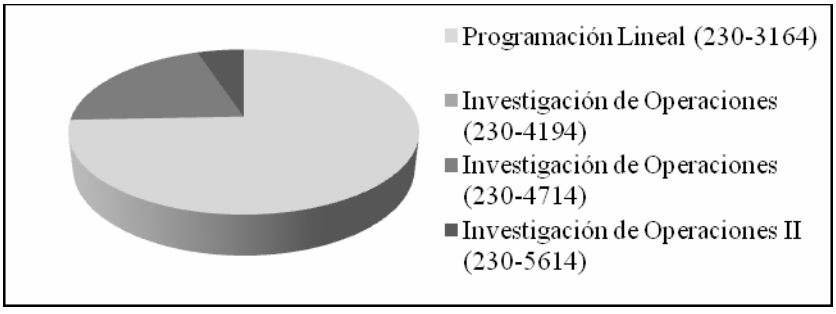


Los resultados obtenidos muestran que la asignatura Programación Lineal (230-3164), era la única materia del área de Optimización que cursaban los estudiantes encuestados.

En el Cuadro Nº 3, se muestra la pregunta número 3 junto con los resultados obtenidos.

La Figura 8 muestra una gráfica correspondiente a los resultados obtenidos para la pregunta número 3.

Figura 8. Resultados gráficos obtenidos de la pregunta número 3 del cuestionario



CUADRO NÚMERO 3 Resultados obtenidos de la pregunta número 3 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
3	De ser negativa la respuesta a la pregunta número 1, ¿cuáles de estas asignaturas, del área de la Investigación de Operaciones, cursó en semestres anteriores.	Programación Lineal (230-3164)	29	74,36%
		Investigación de Operaciones (230-4194)	0	0%
		Investigación de Operaciones (230-4714)	8	20,51%
		Investigación de Operaciones II (230-5614)	2	5,13%

En la Figura 8, se puede observar que un 74,36% de los encuestados ya habían cursado la asignatura Investigación de Operaciones (230-4714) al momento de responder; mientras que un 20,51%, habían cursado la materia Programación Lineal (230-3164). Sólo un 5,13% de los encuestados cursaron la electiva Investigación de Operaciones II (230-5614). Ninguno de los estudiantes encuestados cursó Investigación de Operaciones (230-4194).

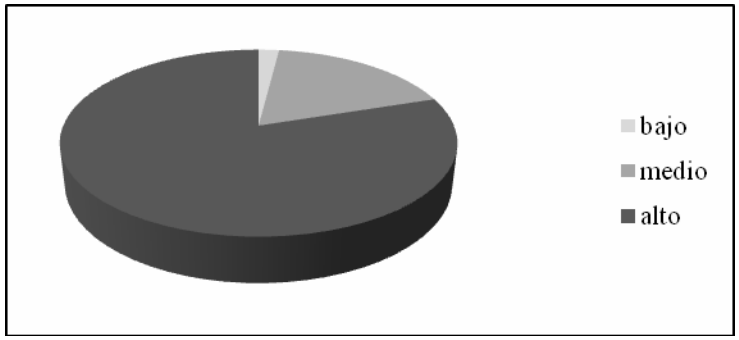
En el Cuadro Nº4, se observan los datos obtenidos en relación a la pregunta número 4.

CUADRO NÚMERO 4 Resultados obtenidos de la pregunta número 4 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
4	¿Cuál es el nivel de dificultad que Ud. considera presentan las asignaturas del área de Investigación de Operaciones?	bajo	1	2%
		medio	9	18%
		alto	40	80%

La Figura 9 muestra, en forma gráfica, los resultados obtenidos de la pregunta número 4.

Figura 9. Resultados gráficos obtenidos de la pregunta número 4 del cuestionario



El 80% de los estudiantes encuestados afirman que las asignaturas del área de IO poseen un alto nivel de dificultad; mientras que un 9% sostiene que el nivel de dificultad es medio. Sólo un 2% señala que el nivel de dificultad es bajo.

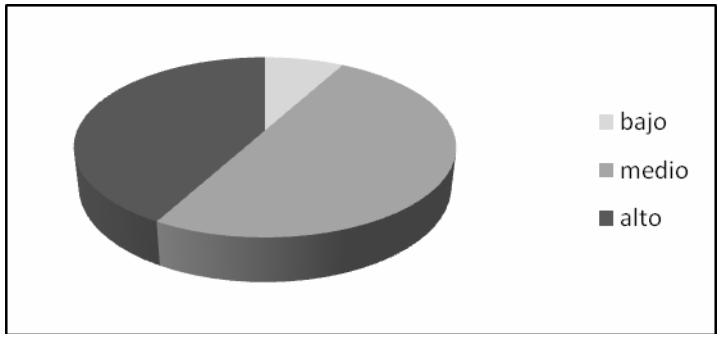
En el Cuadro Nº 5, se muestran los datos obtenidos en relación a la pregunta número 5.

CUADRO NÚMERO 5 Resultados obtenidos de la pregunta número 5 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
5	¿Cuál es el nivel de dificultad que Ud. considera posee el aprendizaje del Algoritmo Simplex?	bajo	4	8%
		medio	25	50%
		alto	21	42%

La representación gráfica de los resultados obtenidos se muestra en la Figura 10.

Figura 10. Resultados gráficos obtenidos de la pregunta número 5 del cuestionario



El resultado obtenido muestra que sólo el 8% de los encuestados afirman que el nivel de dificultad del aprendizaje del Algoritmo Simplex es bajo; mientras que un 92%, expresa que el nivel de dificultad está entre medio y alto.

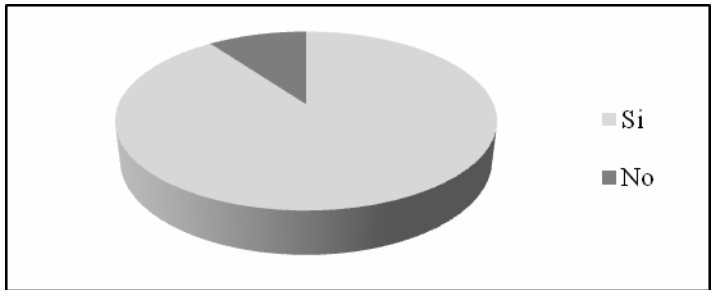
En el Cuadro Nº 6, se presentan los datos obtenidos en relación a la pregunta número 6.

CUADRO NÚMERO 6 Resultados obtenidos de la pregunta número 6 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
6	¿Considera que la Programación Lineal es importante para resolver problemas de la vida diaria?	Si	45	90%
		No	5	10%

La Figura 11 muestra una gráfica correspondiente a los resultados obtenidos para la pregunta número 6.

Figura 11. Resultados gráficos obtenidos de la pregunta número 6 del cuestionario



En los resultados de la pregunta número 6, representado por la Figura 11, el 90% de los encuestados afirmaron que el estudio de la Programación Lineal es útil para resolver problemas cotidianos. Dentro de este aspecto cabe destacar, que una gran parte de los encuestados recomendaron incluir ejemplos de problemas de la vida diaria, que pudieran ser consultados en cualquier momento, desde de la aplicación. Este requerimiento fue tomado en cuenta e incorporado dentro de los requisitos del software.

Uno de los principales temas a tratar fue las características que poseen los paquetes utilizados para resolver problemas lineales, disponibles en los laboratorios de

la Universidad. El objetivo era ayudar a definir cuáles módulos se debían incorporar y qué características debían poseer. Sin duda alguna, el contar con la opinión de los usuarios finales es fundamental para el desarrollo de todo software, por tal motivo se incluyó la pregunta número 7 en este cuestionario, la cual se muestra en el Cuadro Nº 7.

CUADRO NÚMERO 7 Pregunta número 7 del cuestionario.

Nº	**Pregunta**
7	Dentro del contenido programático de las asignaturas del área de Investigación de Operaciones, se utilizan un conjunto de herramientas para resolver problemas lineales, las cuales son: TORA, TORA-Win y LINDO. ¿Cuál es su opinión acerca del funcionamiento y características de estos paquetes computacionales?

A cada bachiller encuestado se le asignó un identificador, lo cual permitió llevar un control sobre cada test llevado a cabo. Siguiendo este método, se muestra en el Cuadro Nº 8, algunas de las respuestas obtenidas a la pregunta número 7, junto con el número del encuestado correspondiente.

CUADRO NÚMERO 8 Algunas respuestas obtenidas de la pregunta número 7 del cuestionario.

Encuestado Nº	**Respuesta**
2	La ventaja del software LINDO radica en que se puede escribir el problema y además resolverlo de forma eficiente.
12	El software TORA-Win es fácil de usar.

CUADRO NÚMERO 8 Continuación.

Encuestado Nº	Respuesta
6	El software LINDO presenta una interfaz más amigable, si lo comparamos con los otros paquetes disponibles.
25	El software TORA es práctico porque trabaja con tablas simplex.
28	El software LINDO usa símbolos que usualmente vemos en otros programas (uso de metáforas ya conocidas)

En el Cuadro Nº 9, se muestra la pregunta número 8 del cuestionario, junto son los resultados obtenidos.

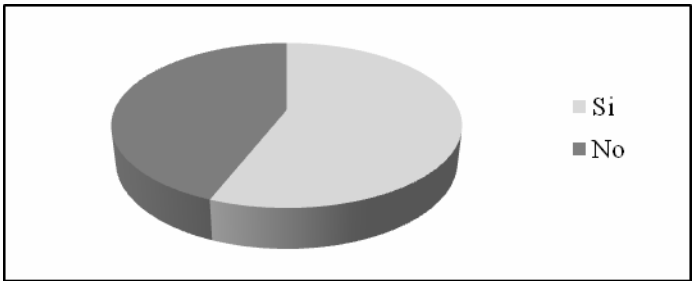
CUADRO NÚMERO 9 Resultados obtenidos a la pregunta número 8 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
8	¿Presentó algún inconveniente al usar alguno de estos paquetes computacionales?	Si	28	56%
		No	22	44%

En la Figura 12 se muestra, en forma gráfica, los resultados obtenidos de la pregunta número 8 del cuestionario.

Los resultados obtenidos para la pregunta número 8, muestran que el 56% de los encuestados presentaron inconvenientes al momento de utilizar los software para resolver problemas lineales; mientras que un 44% afirman que no presentaron ningún inconveniente. Algunas de las opiniones recabadas se muestran en el Cuadro Nº 10.

Figura 12. Resultados gráficos obtenidos de la pregunta número 8 del cuestionario



CUADRO NÚMERO 10 Algunas respuestas obtenidas de la pregunta número 8 del cuestionario.

Encuestado Nº	**Respuesta**
1	El software TORA es difícil de manejar, la solución de los problemas que se introducen son un poco difícil de entender. Además trabaja bajo MS-DOS.
4	Es difícil de manejar; ya que sólo se puede utilizar el teclado para desplazarse por las celdas. Posee una interfaz anticuada.
5	Algunas de estas aplicaciones no reflejan los principios de usabilidad al momento de ocurrir un error, ya que no indican una posible causa del problema y mucho menos cómo resolverlo.
8	No permiten conocer qué método usan internamente para resolver los problemas.
24	Sólo pueden ser utilizados en la plataforma Windows.
25	El software LINDO es tedioso. Tuve problemas para entender la gramática para codificar las instrucciones y eso conllevaba a errores que no son propios del problema planteado.
32	El software LINDO es muy completo, pero debería tener las tablas para la inserción de datos.

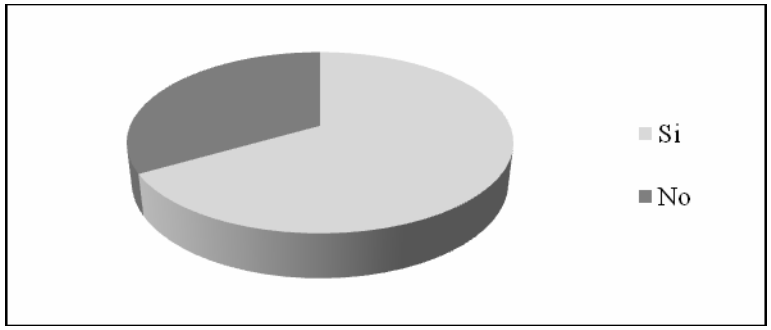
En el Cuadro Nº 11, se muestra la pregunta número 9 del cuestionario junto con los resultados obtenidos.

CUADRO NÚMERO 11 Resultados obtenidos de la pregunta número 9 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
9	¿Los paquetes computacionales, utilizados para resolver problemas lineales, le han ayudado a reforzar los métodos vistos en clase?	Si	32	64%
		No	18	36%

La Figura 13 muestra, en forma gráfica, los resultados obtenidos de la pregunta número 9.

Figura 13. Resultados gráficos obtenidos de la pregunta número 9 del cuestionario



En la pregunta 9, representada por la Figura 13, el 64% de los encuestados afirman que los software que utilizan para resolver problemas lineales los ayudan a comprender los métodos explicados en clase; mientas un 36% opinan que no.

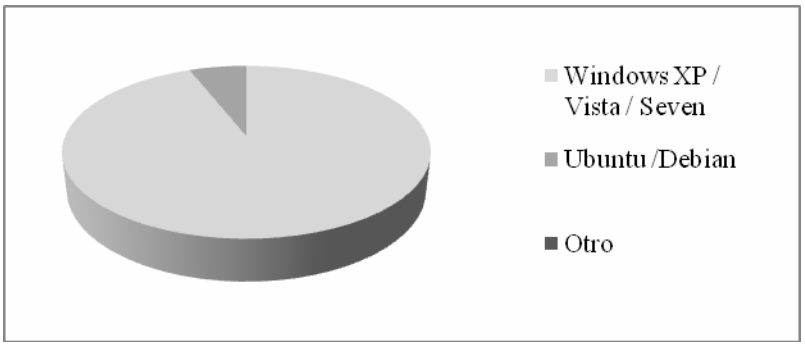
En el Cuadro Nº 12, se observan los datos obtenidos en relación a la pregunta número 10.

CUADRO NÚMERO 12 Resultados obtenidos a la pregunta número 10 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
10	¿Cuál sistema operativo Ud. utiliza con mayor frecuencia?	Windows XP / Vista / Seven.	47	94%
		Ubuntu /Debian.	3	6%
		Otro.	0	0%

La representación gráfica de los resultados obtenidos se muestra en la Figura 14.

Figura 14. Resultados gráficos obtenidos de la pregunta número 10 del cuestionario



En lo que respecta a los resultados a la pregunta número 10, la Figura 14 muestra que el 94% de los encuestados utilizan algunas de las versiones del SO Windows como plataforma de trabajo; en cambio, sólo un 6% señala que utilizan las distribuciones libres *Ubuntu* o *Debian*. Este dato fue tomado en cuenta al momento de seleccionar *Java* como lenguaje base de la aplicación, debido a que posee la característica de poder

ejecutar sus aplicaciones en cualquier SO que cuente con una JMV.

En el Cuadro Nº 13 se muestra la interrogante número 11.

CUADRO NÚMERO 13 Pregunta número 11 del cuestionario.

Nº	**Pregunta**
11	¿Cuáles características considera Ud. debe poseer un software, para resolver problemas lineales, dirigido a los estudiantes de la Licenciatura en Informática de la UDO?

El Cuadro Nº 14 reúne algunas de las respuestas obtenidas de la pregunta número 11.

CUADRO NÚMERO 14 Algunas respuestas obtenidas de la pregunta número 11 del cuestionario.

Encuestado Nº	**Respuesta**
5	Debe ser usable e intuitivo, además de presentar los resultados en un lenguaje más natural.
9	Debe mostrar alternativas y sugerencias para resolver los problemas y que sea en español.
10	Sin duda alguna, una interfaz fácil de manejar y un manual de usuario para ayudarnos.
21	Debe ser multiplataforma, orientado a eventos, que esté acorde a las nuevas tecnologías y con una interfaz usable y amigable, además debe arrojar resultados precisos.

CUADRO NÚMERO 14 Continuación.

Encuestado Nº	Respuesta
25	Interfaz sencilla, agradable, intuitiva y explicativa con cada paso de la solución del problema.
28	Que esté asentado específicamente a problemas relacionados con el área y que ayude a entender mejor los resultados obtenidos.
45	Debería ser multiplataforma, sería lo mejor, o por lo menos que se pueda usar tanto en Windows como en software libre

En el Cuadro Nº 15 se muestra la pregunta número 12.

CUADRO NÚMERO 15 Pregunta número 12 del cuestionario.

Nº	Pregunta
12	¿Cuál es su opinión acerca de las herramientas de apoyo a los procesos de enseñanza y de aprendizaje?

El Cuadro Nº 16 reúne algunas de las respuestas, aportadas por los encuestados, a la interrogante número 12.

CUADRO NÚMERO 16 Algunas respuestas obtenidas a la pregunta número 12 del cuestionario.

Encuestado Nº	Respuesta
2	Mejoran la comprensión de las materias y mientras más material existe de este tipo, más fácil será adquirir los conocimientos.

CUADRO NÚMERO 16 Continuación.

Encuestado Nº	**Respuesta**
14	Es importante, ya que si en algún momento no podemos contar con lo que el profesor da en el aula de clases, podemos utilizar estas herramientas para ponernos al día con la materia.
19	Es importante contar con un software de este tipo, ya que nos permite cambiar la forma tradicional y poco interactiva de aprender una materia.
23	Radica en que aportan una ayuda colateral a la adquirida en el aula de clases y fomentan el aprendizaje autodidáctico en el estudiante.
25	Contar con estos software, permiten practicar los ejercicios fuera del horario de clase, en caso de interrupciones de actividades académicas u otra circunstancia anormal.
32	La tecnología ayuda a mejorar significativamente el desarrollo del proceso de enseñanza, motivo por el cual, contar con este software permite que la clase sea mejor.

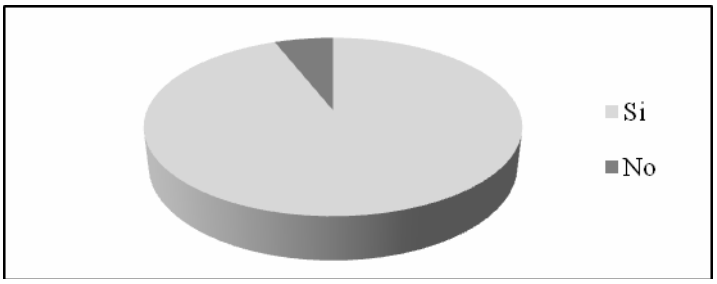
En el Cuadro Nº 17, se muestra la pregunta número 13 del cuestionario junto a los resultados obtenidos.

CUADRO NÚMERO 17 Resultados obtenidos a la pregunta número 13 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
13	¿Considera necesaria la inclusión de un módulo de ejercitación y práctica, que ayude a reforzar los conocimientos que adquiere en clase?	Si	47	94%
		No	3	6%

En la Figura 15, se muestran a través de una gráfica, los resultados obtenidos para la pregunta número 13 del cuestionario.

Figura 15. Resultados gráficos obtenidos de la pregunta número 13 del cuestionario



En la respuesta 13, representada en la Figura 15, el 94% de los encuestados consideran necesario incluir un módulo de ejercitación y práctica; mientras que sólo un 3% afirma que no es necesario. En este caso se puede observar, que la mayoría de los estudiantes encuestados están de acuerdo en desarrollar dicho módulo, por lo tanto, se incluyó dentro de los requisitos de la aplicación.

En el Cuadro Nº 18, se muestra la pregunta número 14 junto con los resultados

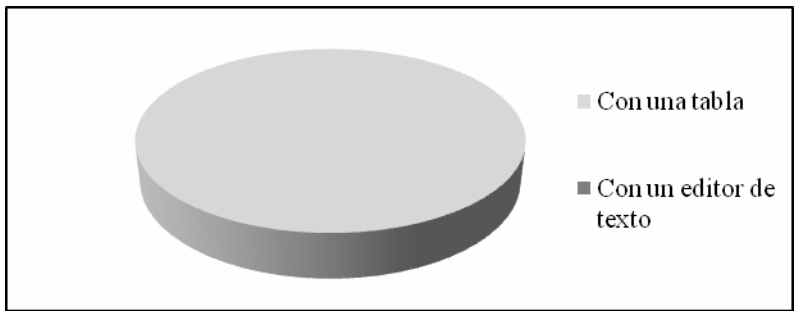
obtenidos.

CUADRO NÚMERO 18. Resultados obtenidos de la pregunta número 14 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
14	¿Qué forma considera Ud. más recomendable para introducir los datos de un problema lineal?	Con una tabla	50	100%
		Con un editor de texto	0	0%

La Figura 16 muestra, en forma gráfica, los resultados obtenidos para la pregunta número 14.

Figura 16. Resultados obtenidos de la pregunta número 16 del cuestionario



En la respuesta número 14, representada en la Figura 16, el 100% de las personas encuestadas opinaron que les era más cómodo plantear modelos lineales a través del uso de tablas simplex. A pesar de estos resultados, se tomó en cuenta, que los problemas con cientos de variables y/o restricciones son fáciles de manejar con la ayuda de un editor; por lo tanto, se decidió que la aplicación contara con ambas formas para trabajar.

En el Cuadro Nº 19, se muestra la pregunta número 15 del cuestionario y los

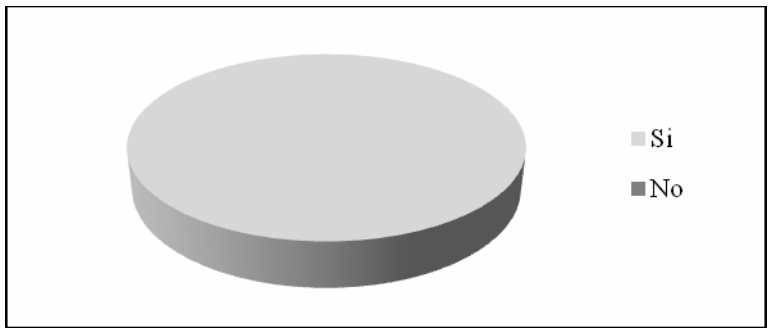
resultados obtenidos.

CUADRO NÚMERO 19 Resultados obtenidos a la pregunta número 15 del cuestionario.

Nº	Pregunta	Alternativas	Frecuencia	Porcentaje
15	¿Le gustaría contar con un software, ajustado a los requerimientos actuales, para resolver problemas lineales?	Si	50	100%
		No	0	0%

La Figura 17 muestra, en forma gráfica, los resultados obtenidos para la pregunta número 15.

Figura 17. Resultados obtenidos de la pregunta número 15 del cuestionario



En la respuesta 15, representada en la Figura 17, el 100% de las personas encuestadas aprobaron el desarrollo de una nueva aplicación enfocada a resolver problemas de PL, utilizando el AS (caso fácil).

Adicionalmente, en el Cuadro Nº 20, se muestra varias recomendaciones dadas

por los bachilleres encuestados, al momento de finalizar el proceso de captura de información.

CUADRO NÚMERO 20 Resultados obtenidos de la pregunta número 16 del cuestionario.

Encuestado Nº	**Respuesta**
2	Tomar en cuenta que la asignatura es bastante abstracta y que existe una mezcla de materias que son antecedentes a ella.
18	Que los resultados sean bastante sencillos de entender y que nos muestren, en forma clara, la solución del problema y otros puntos importantes.
20	Que el software sea implementado, para que los estudiantes tengan otra herramienta de apoyo a su aprendizaje.
23	Sugiero que se incorpore un tutorial, para orientar al usuario en el manejo del software y que lo guie a través de los ejercicios.

3.1.3 Identificación del perfil de los usuarios finales

3.1.3.1 Principiante

Se refiere a los estudiantes que se inician en la asignatura Programación Lineal (230-3164). Éstos poseen conocimientos en el manejo de software orientados a diversas áreas de la informática como: programación, gestión de base de datos y diseño Web. Debido a esto, están familiarizados con las operaciones básicas disponibles en todo software orientado a eventos (abrir un documento, copiar y pegar texto seleccionado, manejo básico de ventanas, entre otras). Por otro lado, y según los requerimientos recabados, la mayor parte de ellos requieren de asesoramiento en el planteamiento de

modelos lineales, además de poder contar con una documentación dentro del software que contemple conceptos relacionados con la PL.

3.1.3.2 Regular

Corresponde a los profesores del área de Optimización. Éstos se destacan por hacer uso frecuente de diversos software orientados a resolver problemas matemáticos de diversa índole, por tal motivo, requieren que la aplicación desarrollada ofrezca flexibilidad en la entrada de datos y una presentación detallada de los resultados.

3.1.3.3 Usuario modelador

El software tiene un sólo tipo de usuario, denominado Modelador. Éste puede tener características de usuario principiante o regular, pero ambos poseen el mismo acceso a todas las funcionalidades desarrolladas.

3.1.4 Redacción de las tarjetas HU

El proceso de redacción de las tarjetas HU se llevó a cabo de forma continua durante todo el proceso de desarrollo de la aplicación. Esto permitió estimar el esfuerzo asociado a cada incremento del software, así como también, planificar el trabajo desde un punto de vista técnico. Adicionalmente, se realizaron reuniones con el cliente para definir, en forma clara, cada uno de los procedimientos a seguir durante el proceso de llenado. En las mismas, se presentó la necesidad de definir qué información debería contener cada tarjeta y en qué forma se iba a representar esta información.

Se decidió seguir las recomendaciones dadas por [31] para el diseño de las tarjetas HU. De esta manera, se adoptó el modelo INVEST, el cual describe los atributos que debe poseer una tarjeta HU para facilitar el proceso de desarrollo ágil de software. En el

Cuadro Nº 21 se muestra un resumen de este modelo.

CUADRO NÚMERO 21 Descripción del modelo INVEST.

Atributo	**Descripción**
Independientes	Cada tarjeta HU debe ser independiente de otra (en la medida de lo posible). Las dependencias entre las tarjetas conllevan a que la planificación, priorización y la estimación del esfuerzo sea mucho más difícil.
Negociables	Todas las tarjetas HU son negociables. No son contratos escritos. Cada tarjeta es sólo una breve descripción de la historia, los detalles se resuelven durante las reuniones.
Valoradas por los clientes o usuarios	Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos.
Estimables	Es importante que el desarrollador pueda estimar (o al menos tener una idea) del tiempo necesario para codificar una tarjeta HU.
Pequeñas	Una buena historia debe ser pequeña en esfuerzo, por lo general no más de dos (2) semanas de programación ideal.
Verificables	Toda tarjeta HU debe ser verificable. Si la historia pasa las pruebas, entonces ha sido implementada con éxito.

El proceso de captura de la información de cada tarjeta fue realizado utilizando una plantilla en digital y no con fichas de papel como es habitual. Esto con el objetivo de minimizar los gastos asociados a la impresión. En la Figura 18, se muestra la estructura de las tarjetas creada para la captura de los requerimientos.

Figura 18. Formato de una tarjeta HU

Historia de Usuario	
Número:	**Puntos asignados:**
Nombre historia:	
Prioridad del cliente: (alta/media/baja)	**Riesgo en desarrollo:** (alto/medio/bajo)
Descripción:	
Notas:	

A cada tarjeta HU se le asignó un total de puntos, desde uno (1) hasta cinco (5), lo cual indica el grado de complejidad del requerimiento capturado y necesario para su implementación. De esta manera, se siguió el criterio para la ponderación sugerido por [31], lo cual señala que, por ejemplo, si una historia posee un total de cuatro (4) puntos, significa que requiere el doble de tiempo para ser codificada con respecto a otra que sólo tenga dos (2) puntos asignados. Además, las tarjetas que poseían características semejantes fueron agrupadas, considerando el grado de prioridad dado por el cliente y las observaciones que se presentaron durante el proceso de llenado, conformando así una entrega o incremento del software. En el Apéndice C, se muestran las tarjetas HU recabadas a lo largo de todo el proceso de desarrollo.

A medida que se iban capturando los requerimientos haciendo uso de las tarjetas, estas se dividían en tareas más simples, teniendo presente que el tiempo asociado a cada tarea no podría sobrepasar tres (3) días de programación ideal. En el Apéndice D, se presentan, en forma de tabla, todas las tareas creadas durante el desarrollo de este proyecto.

3.1.5 Elaboración del plan de liberaciones

Los puntos asignados a cada tarjeta HU permitieron establecer la velocidad del proyecto y, en consecuencia, el número de iteraciones a realizar. En el Cuadro 22, se muestra el nombre de cada tarjeta HU recabada, según el orden en que fueron implementadas, junto a sus puntos asignados.

CUADRO NÚMERO 22 Tarjetas HU y sus puntos asignados.

Nombre de la tarjeta	**Puntos asignados**
Iniciar la aplicación	5
Crear ventana	4
Plantear modelo en forma algebraica	3
Copiar, cortar y pegar texto	3
Plantear modelo en forma tabular	4
Ingresar parámetros del modelo	3
Hallar la solución (modelo en forma tabular)	5
Hallar la solución (modelo en forma algebraica)	5
Hallar la solución en forma guiada	5
Consultar definiciones, ejemplos y ejercicios propuestos	3
Acceder a una función desde la barra de menú	2
Acceder a una función desde la barra de herramientas	2
Buscar y reemplazar texto	3
Deshacer y rehacer escritura	3
Guardar modelo	3
Imprimir modelo	3
Abrir archivo	3
Exportar a formato .pdf	2

CUADRO NÚMERO 22 Continuación.

Nombre de la tarjeta	**Puntos asignados**
Generar operaciones del Algoritmo Simplex	3
Revisar la sintaxis del modelo	4
Guardar resultados y operaciones	2
Seleccionar orden de las tablas	4
Consultar manual de usuario	3
Cambiar tamaño de fuente (Barra de Zoom)	1
Cambiar color de fuente	1
Cambiar tipo de fuente	1
Distribuir ventanas	1
Consultar información del proyecto	1
Salir de la aplicación	2

La forma de ordenar las tarjetas permitió cumplir con una de las recomendaciones de la metodología, la cual señala que se deben implementar aquellos requerimientos que tuviesen mayor valor (prioridad para el cliente). De esta manera, se estableció el valor de 10 (diez) para la velocidad del proyecto, lo cual significa que cada iteración podría abarcar sólo aquellas tarjetas que sumasen diez (10) puntos o menos. Una vez definido este valor, se procedió a agrupar las tarjetas HU para conformar pequeños ciclos de desarrollo. En el Cuadro Nº 23, se muestra el plan de liberaciones elaborado durante la construcción de la aplicación.

CUADRO NÚMERO 23 Plan de liberaciones.

Iteración	Nombre de la historia	Total de puntos
1	Iniciar la aplicación	9
	Crear ventana	
2	Plantear modelo en forma algebraica	10
	Copiar, cortar y pegar texto	
	Plantear modelo en forma tabular	
3	Ingresar parámetros del modelo	8
	Hallar la solución (modelo en forma tabular)	
4	Hallar la solución (modelo en forma algebraica)	10
	Hallar la solución en forma guiada	
5	Consultar definiciones y ejemplos	
	Acceder a una función desde la barra de menú	
	Acceder a una función desde la barra de herramientas.	10
	Buscar y reemplazar texto	
6	Deshacer y rehacer escritura	
	Guardar modelo	9
	Imprimir modelo	
7	Abrir archivo	
	Exportar a formato .pdf	8
	Generar operaciones del Algoritmo Simplex	
8	Revisar la sintaxis del modelo	
	Guardar resultados y operaciones	10
	Seleccionar orden de las tablas	

CUADRO NÚMERO 23 Continuación.

Iteración	Nombre de la historia	Total de puntos
9	Consultar manual de usuario Cambiar tamaño de fuente (Barra de Zoom) Cambiar color de fuente Cambiar tipo de fuente Distribuir ventanas Consultar información del proyecto Salir de la aplicación	10

Cabe destacar que a pesar de llevar a cabo varios ajustes en los tiempos de entregas (debido a la redacción de nuevas tarjetas HU o por la modificación de otras), se pudo establecer un ritmo de trabajo constante, cumpliendo así con cada uno de los requisitos solicitados por el cliente.

3.2 Fase de gestión

3.2.1 Establecer el ambiente de trabajo

Uno de los requisitos de XP es que el cliente siempre debe estar disponible. No sólo para ayudar al equipo de desarrollo, sino para ser parte de este también. Todas las fases de un proyecto de XP requieren comunicación con el cliente, preferiblemente cara a cara, en el sitio [27].

Tomando en consideración que la comunicación entre el cliente y el desarrollador es un factor de suma importancia en un proyecto que hace uso de XP, se estableció un ambiente de trabajo adecuado para llevar a cabo el proceso de construcción de la

aplicación. El laboratorio de Optimización del Núcleo de Sucre de la UDO fue el lugar seleccionado como centro para llevar a cabo las reuniones programadas. El objetivo principal fue el propiciar un clima de trabajo que animara a las personas a opinar y participar en forma espontánea. Adicionalmente, se tomaron en consideración un conjunto de recomendaciones dadas por la propia metodología, al momento de distribuir el espacio de trabajo.

3.2.2 Selección de la arquitectura, herramientas y tecnologías

Se seleccionó *Java* como lenguaje para la programación de los módulos, *Netbeans* 6.8, como entorno principal para construirla aplicación; SDK, como paquete para desarrollar código *Java; Bluefish*1.0.7, como editor de código HTML; *OpenProj* 1.4, como herramienta de gestión de proyectos; *Umbrello* 2.4.2, como herramienta para crear y editar diagramas de UML; *OpenOffice* 3.2, como editor de texto; GIMP, como procesador de imágenes y *Ubuntu* 10.04, como SO.

3.3 Fase de construcción

3.3.1 Planificación

Esta fase comenzó coordinando un conjunto de reuniones con el cliente para la creación y/o recopilación de las tarjetas HU, necesarias para dar inicio al ciclo de iteraciones. Además, se analizó la información suministrada por los estudiantes de la asignatura Programación Lineal (230-3164), a través de cuestionarios aplicados durante las primeras etapas del proceso de desarrollo (ver Apéndice A), logrando establecer los requerimientos funcionales y las tareas que los usuarios han de realizar con la ayuda del software.

A través esta fase, se pudo definir formalmente los primeros requerimientos de la

aplicación, los cuales fueron establecidos bajo la supervisión directa del cliente. A continuación se listan las pautas tomadas en cuenta para la planeación y posterior diseño de la aplicación:

- Construir una aplicación que presente una interfaz intuitiva y fácil de usar.
- Seguir un estándar para la codificación de la aplicación, el cual facilite la comprensión del código (por parte de terceros), con el objetivo que obtener un producto ampliable, es decir, que sea posible incorporarle nuevos módulos a futuro.
- Debe permitir plantear (en forma algebraica y tabular) modelos de PL y que los resuelva utilizando el AS (caso fácil), de manera eficiente.
- La aplicación debe contar con una documentación completa, que incorpore definiciones y ejemplos relacionados con la PL.
- Se recomienda que la aplicación sea multiplataforma, pues así será posible abarcar a usuarios que usen diferentes SO.
- El usuario de la aplicación debe poder observar las operaciones realizadas durante el proceso de búsqueda de la solución de los modelos.
- La aplicación debe contar con opciones que le permitan al usuario crear documentos, guardarlos, recuperarlos y exportarlos a formato .pdf, entre otras funcionalidades que considere necesarias.
- En lo que respecta a la vista gráfica de la aplicación, esta debe ser semejante a la que presentan las aplicaciones para resolver problemas lineales, utilizadas en el área de Optimización del Núcleo de Sucre de la UDO.
- La gramática para plantear modelos debe ser simple y semejante a la que se utiliza dentro de las aulas de clases. Adicionalmente, se debe permitir el ingreso de comentarios, aunque estos deben ser ignorados por el mecanismo utilizado para validar la sintaxis del modelo.
- Los datos introducidos por teclado deben estar colocados en la pantalla de modo que puedan ser leídos como un libro, es decir, de izquierda a derecha y de arriba

abajo.

- Se debe validar que los modelos planteados por el usuario posean los componentes básicos que conforman un modelo lineal: el tipo de problema (minimización o maximización), la función objetivo (función lineal), el conjunto de restricciones y el vector **b** (vector de recursos).
- Para la entrada de datos por medio de tabla simplex, se debe chequear que el valor que contenga cada una de las celdas sea el adecuado. Si el usuario comete algún error, se debe notificar por pantalla para que pueda rectificar la operación.
- El usuario de la aplicación podrá resolver modelos lineales paso a paso, es decir, donde sea él quien seleccione la variable de entrada y la variable de salida, en cada iteración del algoritmo, y la aplicación sólo se limite a verificar la selección y a realizar las operaciones necesarias.
- En el módulo de ejercitación y práctica, la aplicación debe congratular al usuario si éste acierta en la selección de las variables. De igual manera, debe alentarlo a continuar si se equivoca en su selección.
- La aplicación debe poseer la opción de cambiar el número de cifras significativas para los datos de la solución de los modelos.
- Se deben codificar los mecanismos necesarios para prevenir el fenómeno del ciclaje y atascamiento. Además, la aplicación debe señalar si se está en presencia de tablas degeneradas y/o si existen soluciones óptimas alternativas en alguna iteración del algoritmo.

3.3.1.1 Modelado de la aplicación

Se cumplió con la recomendación propuesta por XP y relacionada directamente con el proceso de modelado de software, la cual expresa textualmente:

Desarrollar software que funcione más que conseguir una buena documentación. La regla a seguir es “no producir documentos a menos que sean necesarios de forma

inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental [32].

Según la recomendación anterior, se decidió realizar un proceso de modelado sencillo pero útil, que permitiera definir exactamente lo que se esperaba de la aplicación, identificar las posibles clases del sistema y establecer el orden de los eventos dentro del mismo. De igual manera, se planteó como objetivo principal, contribuir al proceso de documentación del software, permitiendo la visualización rápida de las funcionalidades creadas, a los desarrolladores que se incorporen en un futuro al proyecto.

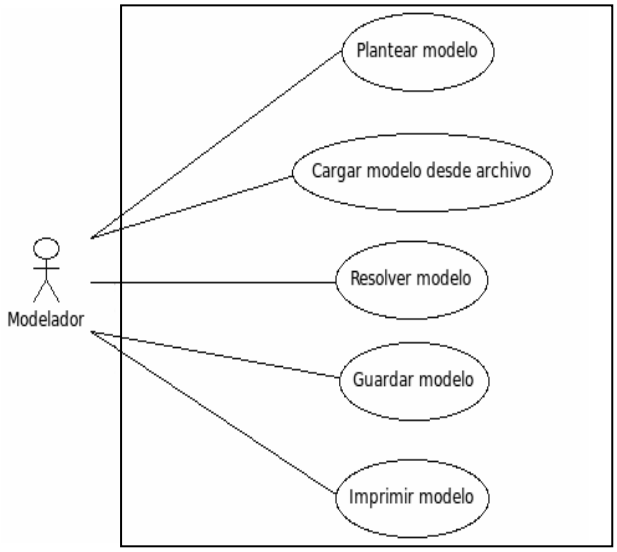
3.3.1.1.1 Diagrama de casos de usos

Tomando en cuenta toda la información recabada durante la fase de exploración, se logró definir los requerimientos iníciales de la aplicación. Dichos requerimientos fueron representados a través de diagramas de casos de uso, permitiendo especificar el comportamiento deseado para la aplicación desde el punto de vista del usuario. Al mismo tiempo, se pudo establecer una visión general de las funcionalidades a desarrollar. En la Figura 19, se muestra el diagrama de casos de uso preliminar de la aplicación.

A medida que se iban capturando nuevos requerimientos, los productos obtenidos fueron evolucionando, hasta alcanzar un modelo general de los escenarios que se han de presentar durante el manejo de la aplicación. La Figura 20 representa el diagrama de casos de uso final para el software. En el mismo, se observan dos elementos principales: el actor y los casos de uso. El actor, denominado Modelador, es la persona encargada de manipular el software, plantear y resolver modelos lineales a través de la aplicación, consultar la documentación, entre otras funcionalidades; mientras que los casos de usos describen un conjunto de acciones que el sistema ejecuta y que produce un determinado

resultado que es de interés para el actor [25]. La descripción detallada de los casos de uso se muestra en el Apéndice E.

Figura 19. Diagrama de casos de uso preliminar



3.3.1.1.2 Diagrama de clases

Al igual que la construcción de los diagramas de casos de uso, el proceso de identificar las clases de la aplicación se llevó a cabo de manera continua a través de todas las iteraciones del proceso de desarrollo. En principio, se realizó un diseño preliminar de las posibles clases, métodos y atributos, y la manera de cómo éstas iban a interactuar dentro de la aplicación. Durante este desarrollo se fueron refinando los diagramas de casos de usos y éstos sirvieron para depurar las clases, obteniendo como producto final el modelo de datos. En la Figura 21, se presenta el diagrama de clases de la aplicación.

Figura 20. Diagrama de casos de uso final

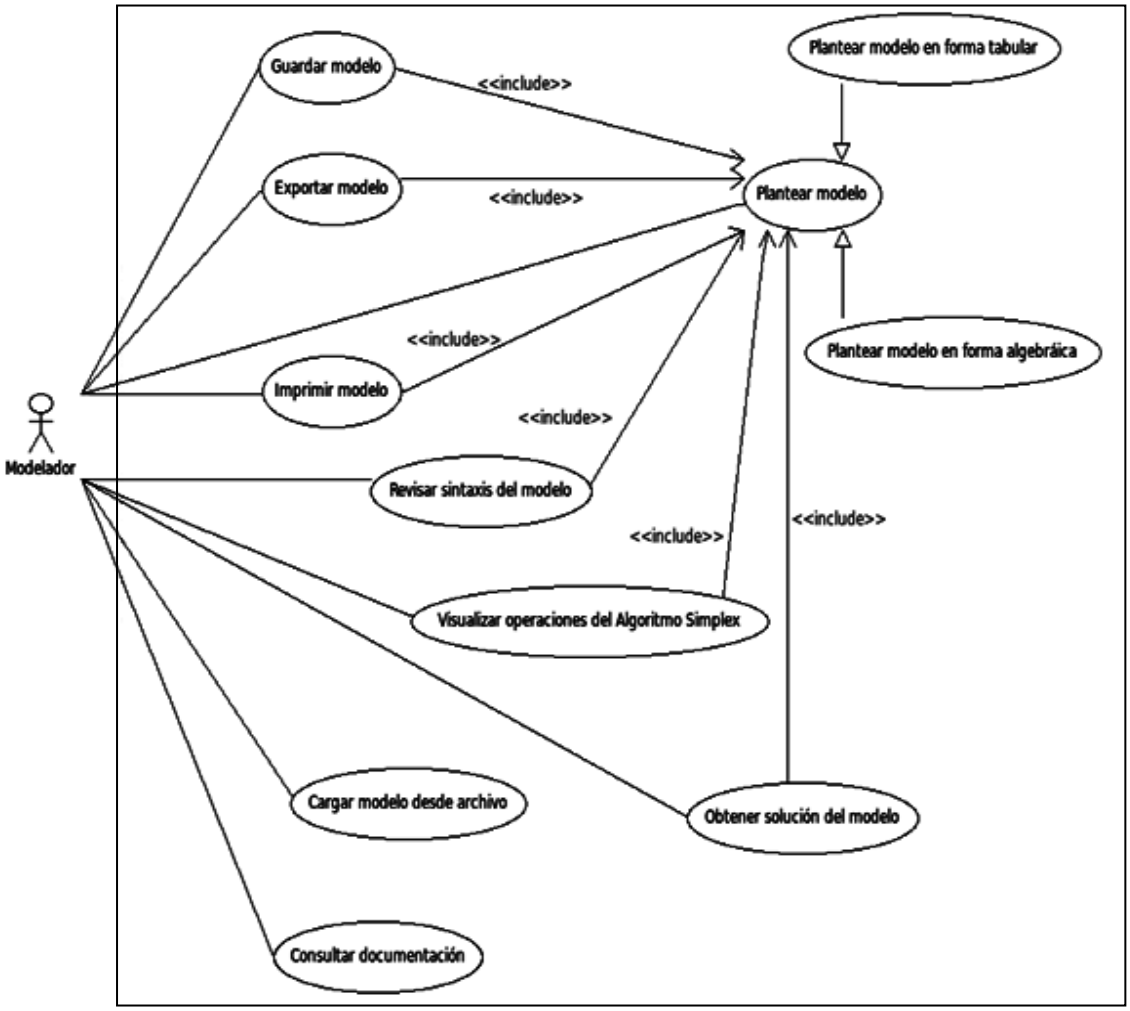
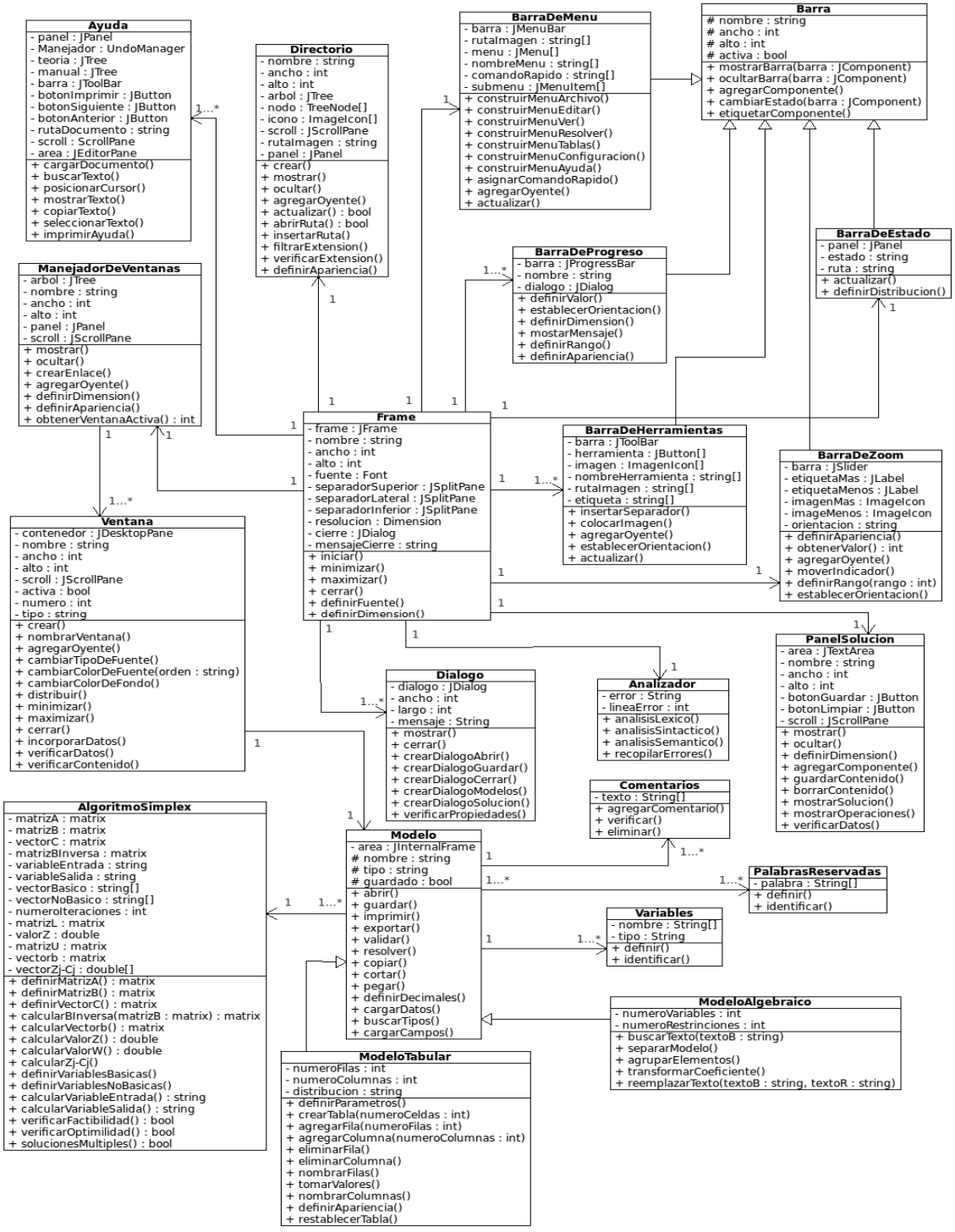


Figura 21. Diagrama de clase de la aplicación



En el Cuadro Nº 24, se presenta una breve descripción de cada una de las clases que componen la vista estática de la aplicación. La descripción de sus métodos se

muestra en el Apéndice F.

CUADRO NÚMERO 24 Listado de clases de la aplicación.

Clase	**Descripción**
Frame	Clase que permite al modelador interactuar con todos los elementos de la aplicación.
Barra	Esta clase reúne los atributos y las operaciones básicas de los componentes del tipo de datos barra.
BarraDeMenu	Con esta clase el modelador puede manipular los elementos que componen la barra de menú de la aplicación.
BarraDeHerramientas	Proporciona un componente útil para mostrarlas acciones de uso más común de la aplicación.
BarraDeEstado	Esta clase permite manipular los elementos que componen la barra de estado de la aplicación.
BarraDeProgreso	Clase encargada de indicar, en forma gráfica, el avance del proceso de una operación dentro de la aplicación.
BarraDeZoom	Componente que permite seleccionar, gráficamente, un valor al deslizar el indicador dentro de un intervalo acotado. Permite cambiar el tamaño de la fuente de las ventanas de la aplicación.
Directorio	Esta clase permite acceder al directorio de la máquina donde se ejecuta la aplicación.
ManejadorDeVentanas	Esta clase controla cada ventana creada por el modelador.

CUADRO NÚMERO 24 Continuación.

Clase	Descripción
Ventana	Representa el área donde el modelador puede plantear un modelo lineal.
Dialogo	Representa las ventanas de diálogo utilizadas por la aplicación para interactuar con el usuario.
Modelo	Esta clase reúne los atributos y las operaciones básicas para manipular los modelos que plantee el usuario dentro de la aplicación.
ModeloAlgebraico	Esta clase posee los mecanismos necesarios para que el usuario de la aplicación pueda plantear modelos lineales en forma algebraica.
ModeloTabular	Esta clase posee los mecanismos necesarios para que el usuario de la aplicación pueda plantear modelos lineales en forma tabular.
AlgoritmoSimplex	Esta clase permite hallar la solución de los modelos lineales que plantee el usuario de la aplicación.
Analizador	Permite realizar el proceso de validación de la sintaxis de los modelos lineales que plantee el usuario dentro de la aplicación.
Variables	Esta clase representa las variables de los modelos lineales planteados por el usuario de la aplicación.
Comentarios	Esta clase representa los comentarios de los modelos lineales planteados por el usuario de la aplicación.
PalabrasReservadas	Representa las palabras que forman parte de la definición de los modelos planteados por el usuario de la aplicación.

CUADRO NÚMERO 24 Continuación.

Clase	Descripción
PanelSolucion	Representa el área donde se muestran los datos de la solución de los modelos lineales.
Ayuda	Representa todo el texto de ayuda con que cuenta la aplicación.

3.3.1.1.3 Diagrama de secuencias

Una vez definidos todos los modelos iníciales fue necesario contar con un medio que permitiese describir, en forma gráfica, la interacción entre los distintos objetos que componen la aplicación y, de igual manera, que favoreciera el entendimiento de cada una de las funcionalidades de los módulos a desarrollar. Por tal motivo, se llevó a cabo la construcción de diagramas de secuencias.

Cada diagrama de secuencia desarrollado representa el curso normal de eventos de un caso de uso. Esto propició la creación de diagramas sencillos, pero útiles, que facilitaran el proceso de codificación. En el Apéndice G, se pueden apreciar los diagramas de secuencias del software.

3.3.1.2 Diseño

En XP, el diseño sigue de manera rigurosa el principio de mantener todo los más simple posible. Siempre se prefiere un diseño simple respecto a una presentación más compleja y así propiciar una guía de implementación para una HU como está escrita, ni más ni menos [1]. Una vez concluida la fase de planificación, se dio inicio al diseño de cada uno de los componentes el software. Cabe destacar, que todo el proceso de diseño se caracterizó por llevarse a cabo de una manera progresiva, evitando crear un diseño

altamente complejo que quisiera abarcar todos los aspectos posibles de una sola vez. A continuación se describen las actividades realizadas durante esta fase.

3.3.1.2.1 Diseño de prototipos

Al momento de construir software es algo muy común que no se conozca con certeza cómo abordar algún requerimiento manifestado por el cliente, en estos casos resulta útil la creación de prototipos [29]. Siguiendo las recomendaciones dadas por XP, se llevó a cabo la construcción de prototipos, los mismos permitieron: evaluar todas las opciones disponibles en relación a la formulación de los modelos lineales, familiarizarse con el lenguaje de programación *Java*, probar las primeras funciones desarrolladas y clarificar los requisitos del usuario.

La Figura 22 corresponde al módulo enfocado a resolver problemas lineales escritos en forma algebraica. En la misma es posible apreciar el uso de pestañas para acceder a las áreas de texto, además, de la utilización de menús y barras de herramientas para organizar las diferentes funcionalidades. La Figura 23 representa el prototipo para el módulo de tabla simplex. Aquí se muestra la manera de representar una tabla que ofrece el lenguaje de programación *Java*, junto a una serie de botones para manipular su dimensión.

3.3.1.2.2 Diseño de la interfaz de usuario

El objetivo principal del diseño de la interfaz fue definir un conjunto de objetos gráficos: menús, íconos, barras de herramientas, comandos abreviados (*shortcut-keys*), entre otros, que le permitirán al usuario final realizar su trabajo de forma fácil y eficiente. En esta fase, se tomaron en cuenta los estándares de diseño de interfaz de usuario para así contribuir, entre otros aspectos, a la usabilidad general del software.
Figura 22. Prototipo de la interfaz de usuario

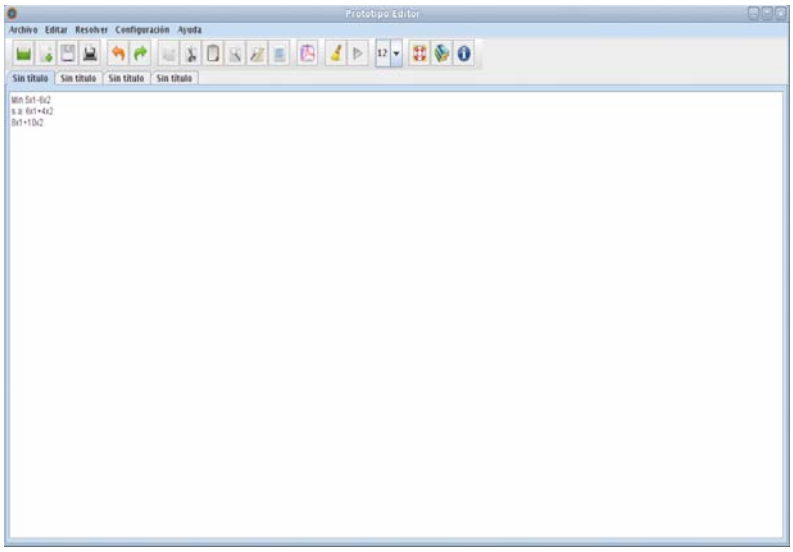
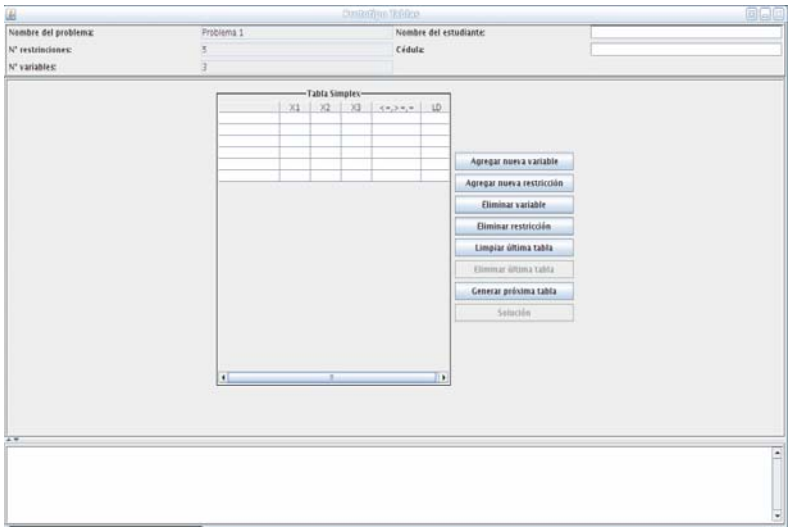


Figura 23. Prototipo del módulo de tabla simplex

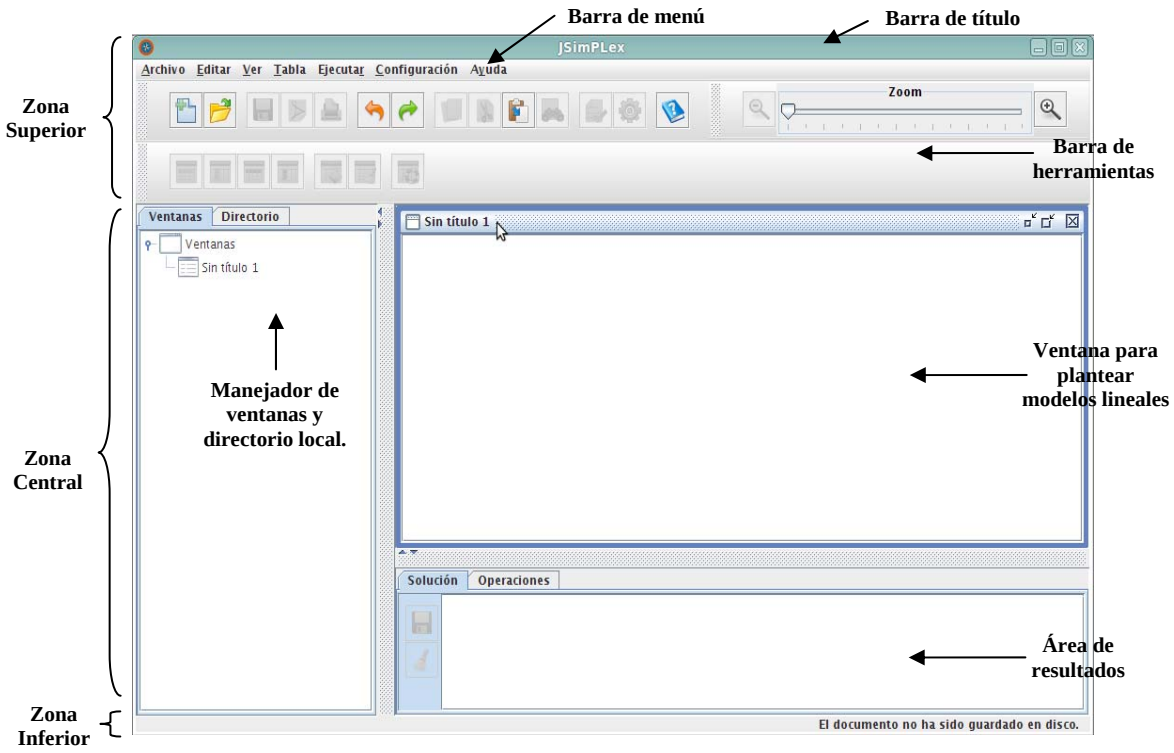


Se siguieron las pautas señaladas por XP para alcanzar un diseño de interfaz simple, pero eficiente, complementándose con las tres reglas de oro para el diseño de interfaz de usuario creadas por Theo Mantel y citadas en [1]. Estos principios proporcionaron una guía para la creación de una interfaz fácil de usar y reconocer, que

les permitirá a los usuarios llevar a cabo sus tareas con el mínimo esfuerzo requerido.

La interfaz del software está dividida en tres (3) zonas: zona superior, zona central y zona inferior. En la Figura 24, se muestra la ventana principal de la aplicación y las partes que la componen.

Figura 24. Interfaz de usuario de la aplicación



A continuación se hace una descripción de la interfaz general del software:

- La zona superior se encuentra dividida en dos áreas. En la primera se ubica la barra de título, donde se localiza el nombre de la aplicación junto con los botones de minimizar, maximizar y cerrar la ventana principal. En la segunda área, se encuentran las barras de menú y de herramientas, las cuales agrupan todas las operaciones disponibles para que el usuario pueda interactuar con la aplicación.

- La zona central está compuesta por tres (3) áreas distribuidas proporcionalmente. El área izquierda está divida a su vez en dos paneles: el primero contempla un área destinada a la visualización rápida de cada una de las ventanas de trabajo creadas por el usuario y el segundo muestra el directorio local de la máquina donde se está ejecutando la aplicación. El área derecha corresponde a la ventana creada para plantear modelos lineales, esta incorpora las operaciones más utilizadas para manejar texto plano: copiar, cortar, pegar, seleccionar texto, entre otras. Debajo de esta última, se encuentra el área de resultados, la cual está dividida en dos paneles: el primero está destinado a mostrar la solución de los modelos lineales planteados por el usuario y en el segundo se listan las operaciones del AS realizadas por el software.

- En la zona inferior de la aplicación se muestra la barra de estado, la cual ofrece información sobre las ventanas creadas y el estado del documento de trabajo activo.

El uso de menús le permitirá al usuario de la aplicación seleccionar entre una serie de opciones para realizar una determinada tarea. Estos menús reúnen todas las acciones u operaciones disponibles y sus ítems incorporan el uso de teclas de acceso rápido y de caracteres mnemónicos. Adicionalmente, los comandos usados con mayor frecuencia fueron agrupados en una barra de herramientas. También se construyó una barra de desplazamiento para ajustar el tamaño de la fuente de la ventana de trabajo activa. Dicho mecanismo realiza un seguimiento continuo para ajustar su estado, dependiendo del área donde esté trabajando el usuario. En la Figura 25, se ilustra la zona superior del software compuesta por las barras de menú y de herramientas.

Figura 25. Zona superior de la interfaz del software

3.3.1.2.3 Módulo para resolver problemas lineales escritos en forma algebraica.

A través de este módulo, el usuario de la aplicación podrá plantear modelos lineales ingresando, por teclado, los tres componentes básicos de todo modelo de PL, los cuales son: las variables de decisión que se tratan de determinar; el objetivo o meta, el cual se busca optimizar y las restricciones, que se deben satisfacer. Además, tendrá la posibilidad de escribir comentarios al final de cada instrucción, procurando así un mejor entendimiento del modelo por parte de terceros.

Para minimizar los posibles errores que pueden suscitarse al momento de plantear un modelo, se diseñó un mecanismo capaz de evaluar la sintaxis de cada instrucción ingresada por el usuario. Primeramente, se definió la gramática del programa fuente (en este caso, la gramática abarca cada elemento que forma parte del planteamiento de un modelo de PL) indicando así, todos sus componentes léxicos (*tokens)*, la estructura que describe su sintaxis y las especificaciones semánticas.

En los lenguajes de programación, el léxico lo constituyen todos los elementos individuales del lenguaje, denominados frecuentemente *tokens*. Así son *tokens*: las palabras reservadas del lenguaje, los símbolos que denotan los distintos tipos de operadores, identificadores (de variables, funciones, procedimientos, tipos, entre otros), separadores de sentencias y otros símbolos empleados en las distintas construcciones de un lenguajes [33]. En el Cuadro Nº 26, se utiliza la notación BNF para representar los elementos utilizados para plantear modelos lineales.

CUADRO NÚMERO 26 Notación BNF del lenguaje creado para plantear modelos lineales.

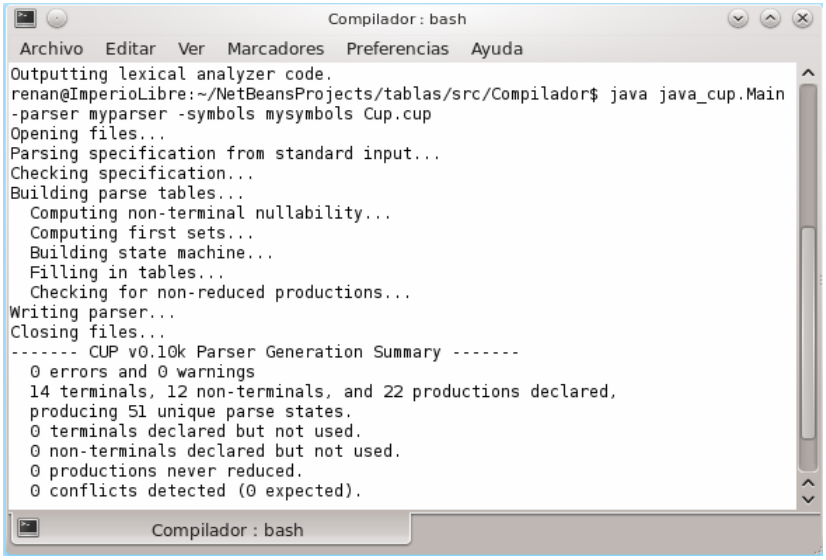
Elemento	**Representación**
<modeloLineal> : : =	<tipo> <función_objetivo> <fin_sentencia> <condición> <restricción>
<tipo> : : =	min | max | minimizar | maximizar | minimice | maximice
<función_objetivo> : : =	<expresión> | - <expresión>
<expresion> : : =	<termino> | <termino> + <expresion> | <termino> - <expresion>
<termino> : : =	<identificador> | <numero> <identificador>
<identificador> : : =	<cadena> | <cadena> <numero>
<restriccion> : : =	<expresion> <desigualdad> <numero> ; | - <expresion> <desigualdad> <numero> ; | <restriccion> <expresion> <desigualdad> <numero> ; | <restriccion> - <expresion> <desigualdad> <numero> ;
<numero> : : =	entero | decimal | fraccion;
<decimal> : : =	<cifra> . <cifra>
<fracción> : : =	(<cifra> / <cifra>)

CUADRO NÚMERO 26. Continuación.

Elemento	Representación
<cifra> : : =	[0-9]+
<cadena> : : =	[a-z A-Z]+
<desigualdad> : : =	<=
<condicion> : : =	sa: | st: | s.t: | s.a:

Una vez finalizado el proceso de definición de la gramática, se procedió a depurarla mediante la compilación de sus librerías. Para ello, se utilizaron las herramientas que proporciona el entorno de desarrollo *Java*, junto con la consola del SO *Ubuntu*. En la Figura 26, se aprecian los resultados obtenidos durante el proceso de compilación.

Figura 26. Proceso de compilación del analizador de la gramática



3.3.1.2.4 Módulo para resolver problemas lineales escritos en forma tabular.

La FTMS utiliza una tabla simplex para desplegar, de manera resumida, el sistema de ecuaciones que llevan a la solución básica factible en ese momento. Para esta solución, cada variable en la columna de la izquierda es igual al número correspondiente en la columna de la derecha (y las variables que no aparecen son iguales a cero). Cuando se realiza la prueba de optimalidad o una iteración, los únicos números relevantes son los que están a la derecha de la columna **Z** (que incluyen el número del renglón derecho) [2]. En la Figura 27, se presenta un ejemplo de un modelo lineal planteado a través de tabla simplex.

Figura 27. Modelo lineal planteado a través de tablas simplex

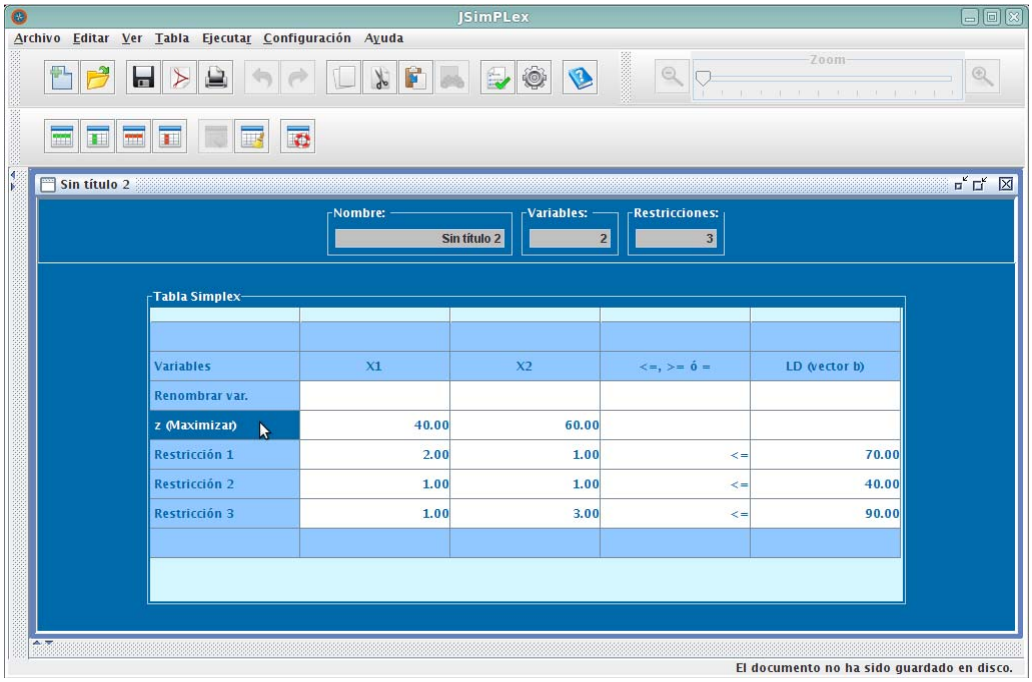
Forma tabular								
Variable básica	Ec. núm.	Coeficiente de:						Lado derecho
		Z	x_1	x_2	x_3	x_4	x_5	
Z	(0)	1	−3	−5	0	0	0	0
x_3	(1)	0	1	0	1	0	0	4
x_4	(2)	0	0	2	0	1	0	12
x_5	(3)	0	3	2	0	0	1	18

Basado en la FTMS, se diseñó un módulo enfocado a resolver problemas lineales haciendo uso de tablas simplex. El mismo presenta una interfaz semejante a la mayoría de los paquetes computacionales creados para tal fin. Esto permitirá que los estudiantes, profesores del área de Optimización, del Núcleo de Sucre de la UDO, se familiaricen más rápidamente con su modo de uso.

Se incorporaron varias funcionalidades tales como: el uso de menús contextuales, la posibilidad de cambiar la dimensión de la tabla sin perder los valores cargados hasta ese momento, la validación de la estructura del modelo, entre otras, las cuales permitirán que el usuario de la aplicación realice su trabajo de forma rápida y confiable.

En la Figura 28 se observa la interfaz de usuario creada para el módulo de tabla simplex.

Figura 28. Interfaz final del módulo de tablas simplex



3.3.1.2.5 Diseño del módulo de ejercitación y práctica

Este módulo funciona básicamente de la siguiente manera: el usuario de la aplicación plantea un modelo lineal, de caso fácil, mediante el uso de tablas simplex. El software presenta, en una nueva ventana, la tabla inicial de la solución del modelo. De aquí en adelante, el modelador señala la variable de entrada y la variable de salida en cada iteración. La aplicación evalúa la selección realizada (tomando en cuenta los criterios del AS, previamente codificados). Si la selección es la adecuada, realiza los cálculos necesarios y muestra la siguiente tabla, en caso contrario, notifica el error por pantalla. El usuario entonces, tiene la posibilidad de corregir su selección y revisar las

observaciones dadas por la aplicación.

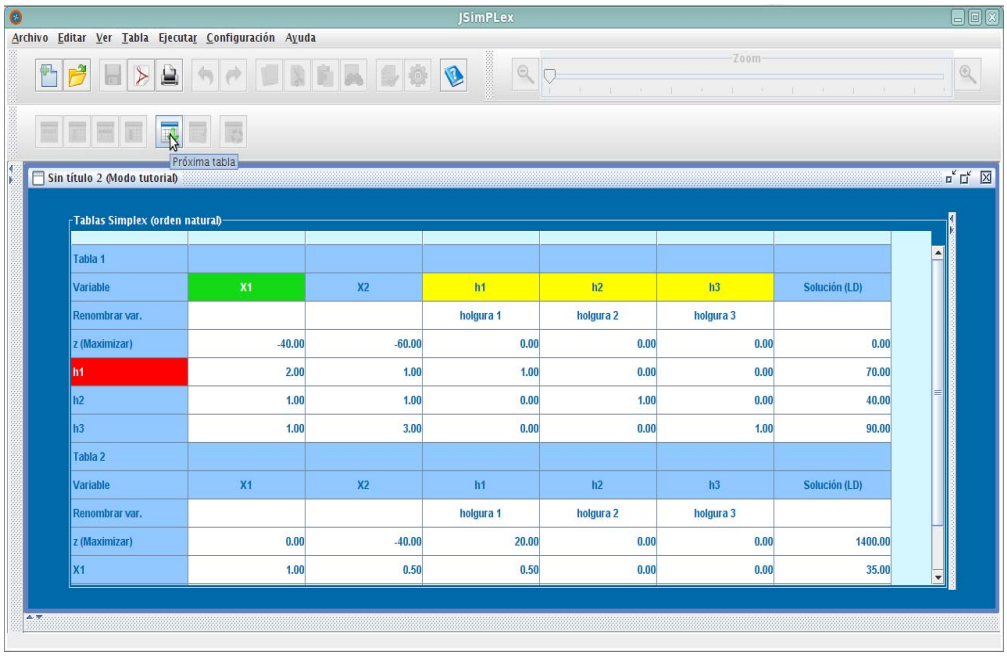
Para el desarrollo de este módulo, hubo la necesidad de incorporar ciertos aspectos didácticos y pedagógicos que facilitaran y garantizaran la satisfacción de la necesidad. Por tal motivo, se decidió seguir las pautas dadas en [34], en lo referente a los conceptos de retroalimentación y motivación que deben estar presentes en las aplicaciones utilizadas como apoyo a procesos de aprendizaje.

Un aspecto importante que surgió en medio del diseño del módulo fue el concepto de retroalimentación. Esta se traduce en mostrar el efecto de lo que hizo el usuario, independientemente si es correcto o no, para que éste sea quien analice lo que ha pasado y tome decisiones al respecto [34]. De esta manera, el modelador tendrá dos posibles escenarios: el primero, si su selección fue la correcta, se le mostrará la siguiente tabla (que representa una nueva iteración del AS) y podrá continuar con el ejercicio; y un segundo escenario, en caso de no acertar en su selección, se mostrará el vector **b** resultante de esa operación, la columna seleccionada o la observación de que no se cumplió con algún criterio del AS (según sea el caso). Así podrá entonces, analizar lo sucedido y continuar hasta hallar la solución del modelo.

El poder mantener motivado al usuario, cuando éste resuelva un problema, fue otro punto a considerar en medio del desarrollo de este módulo. Hay que mantener motivados a los usuarios para que el trabajo que se tenga con la aplicación sea efectivo y de provecho [34]. Por tal motivo, se diseñó un mecanismo que permitiese mostrar un mensaje dependiendo del desenvolvimiento del usuario. De esta manera, si acierta en su selección al primer intento, se le mostrará un mensaje congratulándolo; si ha cometido algún error en su selección, se le alentará a continuar y si el error es reiterativo, la aplicación le invitará a consultar la documentación que esta ofrece y así, pueda disipar sus dudas y continuar.

El objetivo principal de este módulo fue permitirles a los estudiantes que se iniciaron en el área de la IO, reforzar los conocimientos que adquirieron en clase; además, de ofrecer la posibilidad de utilizar tablas simplex para identificar distintos escenarios que se presentan al momento de resolver problemas lineales, como por ejemplo: identificar el tipo de solución del modelo, determinar si una tabla es simplex, verificar si se está en presencia de soluciones óptimas alternativas y/o de tablas degeneradas y comprobar que existen varios caminos que permiten encontrar la solución de un modelo lineal. En la Figura 29, se muestra una imagen correspondiente a la interfaz del módulo de ejercitación y práctica.

Figura 29. Interfaz final del módulo de ejercitación y práctica



3.3.1.2.6 Diseño del módulo de consultas

Otro requerimiento solicitado por los cursantes de las asignaturas del área de IO fue contar con un módulo para consultar definiciones, terminologías básicas y ejemplos

relacionados con la PL, específicamente, la resolución de problemas lineales de caso fácil. Este módulo tiene como objetivo principal permitir al estudiante complementar su estudio formal de esta disciplina.

Dada la importancia de contar con información fidedigna, se tomó la decisión de incluir sólo aquellas definiciones provenientes de una fuente apropiada, siendo esta libros de autores reconocidos del área de IO. Además, toda la información fue organizada de manera que pudiese ser localizada en forma intuitiva por parte del usuario de la aplicación.

Otros aspectos a destacar son: la incorporación de imágenes acordes a cada uno de los temas expuestos, la posibilidad de realizar una búsqueda de algún término en específico, la opción para imprimir parte o toda de la información consultada por el usuario y por último, una serie de recomendaciones de libros y sitios Web que abracan temas como la PL, sus métodos, la formulación de modelos lineales, entre otros. En la Figura 30, se puede apreciar la interfaz general desarrollada para el módulo de consultas.

3.3.1.3 Codificación

Una vez concluida la fase de diseño, se dio inicio a la etapa de codificación. Se siguió el estándar de codificación para el lenguaje de programación *Java* sugerido en [35], el mismo expone una sintaxis sencilla pero útil para escribir programas en *Java* a través de un conjunto de reglas para la selección de los tipos de datos, declaración clases, métodos y variables, manejo de excepciones, forma de documentar el software, entre otras, obtenido así un código homogéneo y fácil de entender, que facilita el proceso posterior de detención y corrección de errores.

Es importante mencionar, que se utilizaron librerías en *Java* codificadas por

terceros, las cuales permitieron incorporar una serie de funcionalidades adicionales solicitadas por los usuarios durante la captura inicial de requerimientos. En el Cuadro Nº 27, se señala cada una de las librerías utilizadas, junto con una breve descripción de las mismas.

Figura 30. Interfaz del módulo de consultas

CUADRO NÚMERO 27 Librerías utilizadas durante el proceso de construcción de la aplicación.

Biblioteca	**Descripción**
iText	Biblioteca *Open Source* para crear y manipular archivos PDF, RTF y HTML en *Java*, distribuida bajo la *Mozilla Public License* con la LGPL como licencia alternativa.

CUADRO NÚMERO 27. Continuación.

Biblioteca	**Descripción**
Jama	Paquete de álgebra lineal básica para *Java*. Proporciona clases para construir y manipular matrices del tipo de dato real. Posee la suficiente funcionalidad para llevar a cabo rutinas cotidianas, empaquetado de una manera natural y entendible por personas no expertas.
JLex	Herramienta desarrollada en *Java* que recibir un archivo de entrada (archivo .lex) dentro del cual se declaran expresiones regulares y *tokens* válidos para un lenguaje que se desean reconocer. Dicho de otro modo, Jlex es una herramienta para generar analizadores léxicos.
CUP	Generador de analizadores sintácticos para *Java*.
JaveHelp	Expansión de *Java* que facilita la programación de las ventanas de ayuda en las aplicaciones informáticas.

Durante la fase de codificación se empleó un conjunto de herramientas que facilitaron, en gran medida, el proceso de construcción de los diferentes módulos de la aplicación. En las Figuras 31 y 32, se muestran la interfaz de los editores utilizados, *Netbeans* 6.8 y *Bluefish* 1.0.7. Ambas aplicaciones permitieron realizar un trabajo de codificación ordenado y eficiente, facilitando así el posterior proceso de refactorización de código.

Durante esta fase se utilizaron diferentes tipos de archivos, cada uno cumple una función diferente dentro de la aplicación. En el Cuadro Nº 28, se muestran las extensiones de los archivos empleados, junto con una breve descripción de las mismas.

CUADRO NÚMERO 28 Elementos utilizados en la aplicación.

Extensión	Descripción
*.java	Archivos que contienen el código desarrollado en *Java*.
*.class	Archivos de una clase generado por el compilador de *Java*.
*.html	Archivos para la elaboración de páginas Web.
*.lex	Archivos que contienen las especificaciones de entrada para el analizador léxico LEX.
*.cup	Archivos que contienen las especificaciones de entrada para el analizador sintáctico CUP.
*.jmh	Fichero de mapa para *JavaHelp*. Contiene las direcciones relativas de los componentes usados en el módulo de bases teóricas.
*.hs	Fichero de configuración de *JavaHelp*.

Figura 31. Interfaz del IDE *Netbeans* 6.8

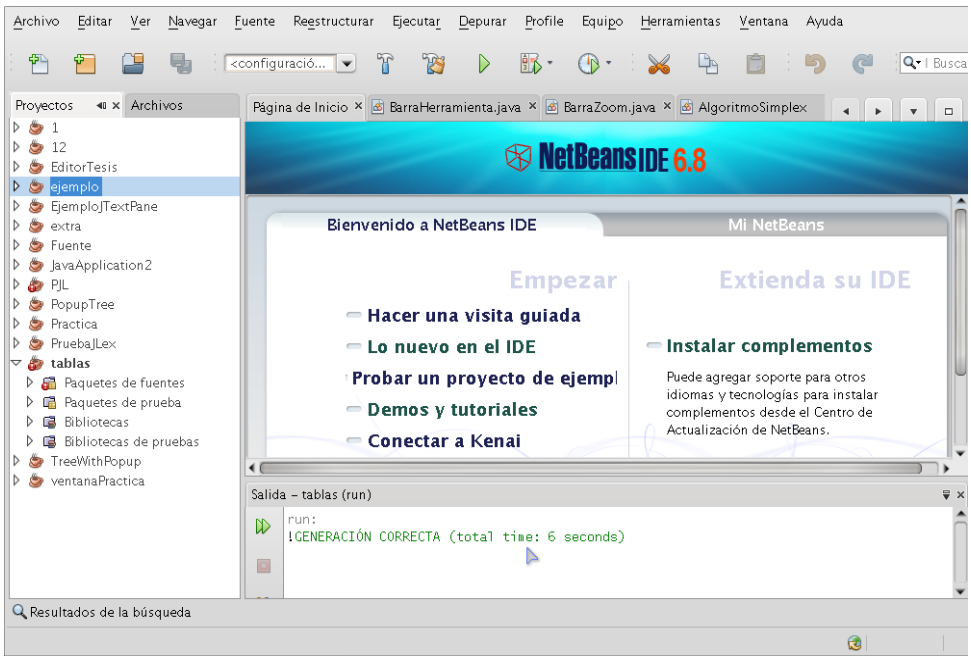
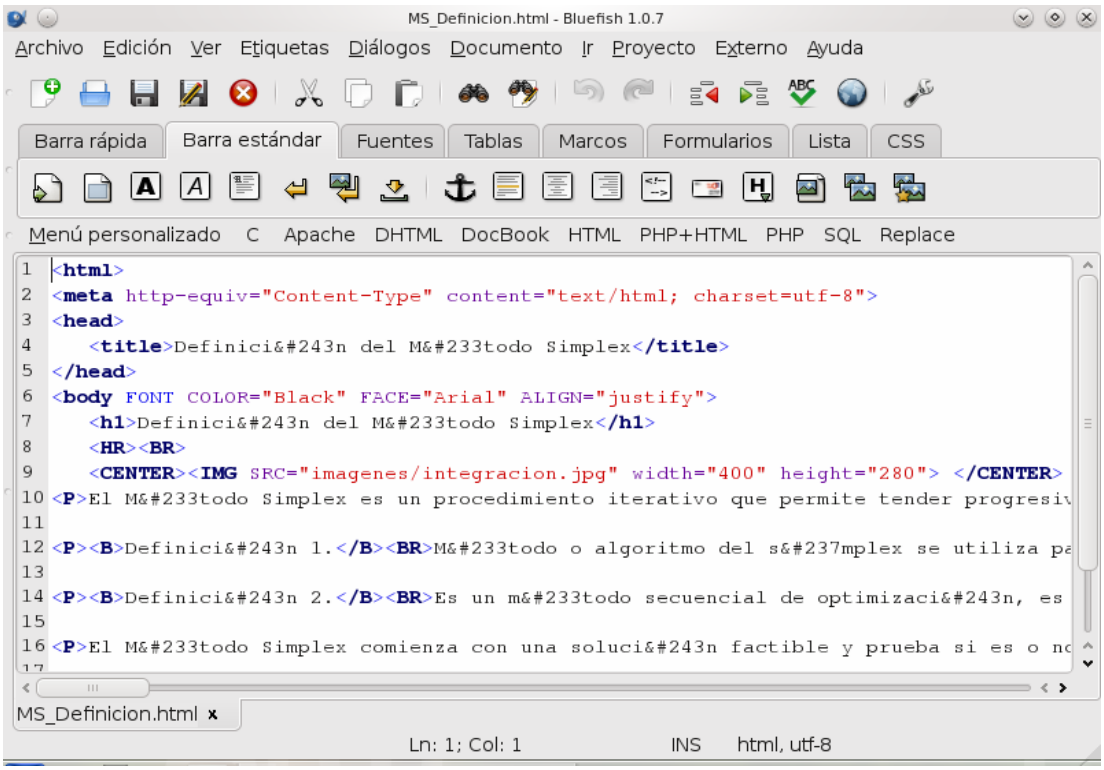


Figura 32. Interfaz del software *Bluefish* 1.0.7



En lo que respecta a la codificación del AS, se tomaron en consideración varios fenómenos que ocurren al momento de utilizar este algoritmo. El objetivo principal fue que la aplicación mostrara los datos de la solución de los modelos de la forma más clara y precisa posible.

La degeneración provoca varias dificultades conceptuales y computacionales en la PL. En ausencia de degeneración, el MS termina de manera finita, ya sea produciendo una solución básica factible óptima o bien comprobando no acotamiento. Sin embargo, cuando ocurre un pivote degenerado, simplemente se pasa de una base a otra, en donde ambas representan al mismo punto extremo solución. A medida que se repite el proceso es concebible que se tome otro pivote degenerado, dando por resultado el mismo punto extremo con una diferente representación básica. Por consiguiente, es posible aunque

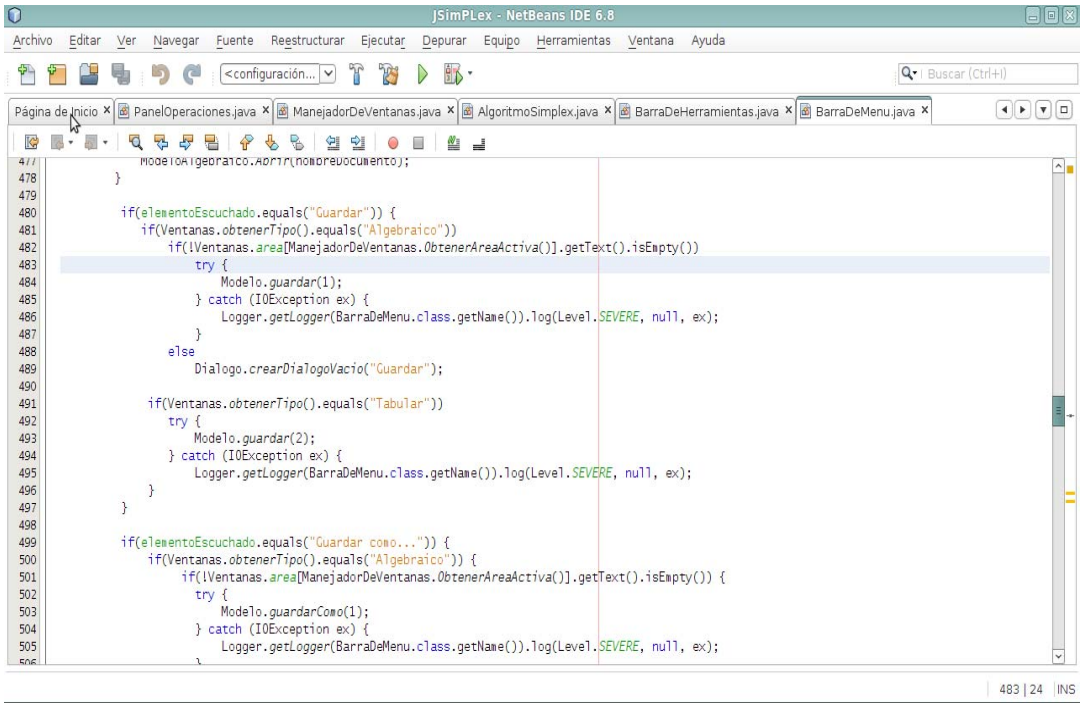
improbable, que el proceso permanezca en un punto extremo no óptimo y pivotee a través de una sucesión de bases asociada $B_1, B_2,..., B_t$, en donde $B_t = B_1$. Si la misma sucesión se usa una y otra vez, entonces el proceso estará siempre en un ciclo entre las bases $B_1, B_2,..., B_t = B_1$, sin llegar a la solución óptima [2]. Con el objetivo de evitar las consecuencias del ciclaje, se decidió incorporar la regla de Bland al proceso de codificación del AS.

El tratamiento de excepciones fue otro de los aspectos a considerar al momento de codificar cada una de las rutinas del software. *Java* ofrece al programador mecanismos que permiten tomar medidas para evitar que la aplicación culmine de manera inesperada. En primera instancia fue necesario identificar los bloques de código donde se podría presentar una posible excepción, ya sean excepciones de escritura o lectura de archivos, conversiones numéricas, manejo de arreglos, entre otras. Posteriormente, se incluyó dentro de estos bloques los métodos *try-catch*, ellos cumplieron el rol de manejador de excepciones, permitiendo capturarlas y tratarlas de manera eficiente. Finalmente, se llevó a cabo un conjunto de pruebas rápidas a fin de verificar si los métodos funcionaban correctamente. A manera de ejemplo, en la Figura 33, se presenta un fragmento de código de la aplicación donde se implementa el manejo de excepciones.

3.3.1.4 Pruebas

Abarcaron las pruebas unitarias y las pruebas de aceptación. Ambas fueron escritas durante el proceso de llenado de las tarjetas HU, permitiendo así, identificar cada escenario antes de escribir el código necesario para su implem0entación. El objetivo fue el de minimizar la probabilidad de que ocurriese algún error al momento de ejecutar la aplicación.

Figura 33. Fragmento de código donde se implementa el manejo de excepciones



3.3.1.4.1 Pruebas unitarias

Estas pruebas fueron ejecutadas por el desarrollador, de forma rápida y continúa, justo antes de implementar una nueva característica del software. El objetivo fue el localizar errores en la semántica y sintaxis de los textos, errores de vinculación en los menús y botones y, en general, aspectos relacionados con la interfaz gráfica de la aplicación. Si se conseguía algún error, se procedía a corregirlo y realizar nuevamente la prueba. En lo que respecta al manejo de las mismas, se siguieron las recomendaciones citadas en [36].

Cabe destacar que durante la ejecución de estas pruebas se aplicó una refactorización de código, a fin de mejorar las instrucciones escritas hasta ese momento. Una vez que el código cumplía con las expectativas, se procedía a aplicar las pruebas de aceptación.

3.3.1.4.2 Pruebas de aceptación

Fueron redactadas junto con el cliente y permitieron evaluar, desde su punto de vista, si el requerimiento solicitado cumplía con los parámetros establecidos en las tarjetas HU. Esto contribuyó a una mayor confianza en el código, ya que se probaba casi inmediatamente después de escribirse, facilitando la tarea de adaptarlo a nuevos cambios si era necesario.

Es importante mencionar, que sólo se crearon pruebas de aceptación para aquellos requerimientos donde era necesario una entrada de datos por parte del usuario, a fin de validar si los resultados obtenidos eran los correctos. Se utilizó el formato sugerido en [37] debido a su sencillez, tanto al momento de crearlas como de aplicarlas. En el Apéndice H, se presentan cada una de las pruebas de aceptación realizadas durante el desarrollo de la aplicación.

3.3.1.4.3 Validez de los resultados y tiempo computacional de la aplicación

Fue necesario comprobar la eficiencia de la aplicación desarrollada, específicamente en lo que respecta a la solución de problemas de PL. Para esto, se seleccionaron de [2] cincuenta (50) modelos lineales, de caso fácil. Cada uno de ellos fueron planteados y resueltos, utilizando tanto el software construido durante esta investigación como el paquete computacional LINDO (versión 6.1). Los resultados obtenidos se compararon entre sí, para verificar su similitud. En algunos casos, los datos de la solución no coincidieron, entonces se realizó una revisión del código y se verificó cada instrucción escrita hasta ese momento. Luego de hacer las correcciones necesarias, se procedió a plantear y resolver el modelo, nuevamente. Al final, ambas aplicaciones arrojaron los mismos resultados, para cada uno de los modelos utilizados.

Adicionalmente, durante la etapa de refactorización de código, se realizaron

pruebas para observar el tiempo computacional de la aplicación. Se utilizó el módulo de tablas simplex para generar, en forma aleatoria, los datos de los modelos. Los valores para la matriz **A** y el vector **C** fueron del tipo real, con dos (2) decimales y variaban desde menos cien (-100) hasta cien (100) y para el vector **b,** sólo se permitieron valores positivos. Se calculó el promedio de los tiempos necesarios para hallar su solución y el total de iteraciones realizadas. En el Cuadro 29 se presentan los resultados obtenidos.

CUADRO NÚMERO 29 Tiempos de respuesta de la aplicación.

Orden de la tabla	**Total de elementos**	**Iteraciones necesarias (promedio)**	**Tiempo necesario (promedio)**
10x10	200	6,8 iteraciones	0,009359304 s
50x50	5000	24,2 iteraciones	0,064508116 s
10x100	20000	45,2 iteraciones	0,227060339 s
200x200	80000	136 iteraciones	3,853781723 s
300x300	180000	117,8 iteraciones	10,087378364 s
500x500	500000	226,6 iteraciones	2,102796681 min
750x750	1125000	312,4 iteraciones	11,053517306 min
750x1000	1500000	332,4 iteraciones	11,466414989 min
750x1500	2250000	474,2 iteraciones	17,240001016 min

Cabe señalar que, las pruebas fueron realizadas en una computadora con las siguientes características: SO *Ubuntu* 10.04, procesador *Dual-Core* a 2.60 GHz, 1 Gb de RAM, disco duro de 120 GB y JMV versión 6 *Update* 27.

3.4 Fase de culminación

3.4.1 Integración de los módulos

Una vez que se producía una liberación de código, se iniciaba el proceso de integración de las nuevas funcionalidades al resto del sistema. De este modo, todas las partes del código fueron fusionadas en etapas tempranas de desarrollo, permitiendo detectar errores de vinculación. Una vez integrado un nuevo módulo, se aplicaban las pruebas nuevamente para comprobar que la comunicación entre las partes fuese armoniosa y que lograran, en conjunto, hacer su trabajo correctamente. Paralelamentemente, se realizaron varios procesos de refactorización de código, logrando así aumentar la eficiencia del sistema al someterlo continuamente a un proceso de depuración de sus componentes, eliminando funciones repetidas y demás código innecesario.

3.4.2 Elaboración de la documentación del software

Se basa en la realización del manual de usuario como soporte a la aplicación desarrollada. Este documento fue redactado de la forma más clara, sencilla y completa posible, pudiendo ser consultado en cualquier momento por el usuario dentro de la aplicación. En el Apéndice I se muestra el manual de usuario.

CONCLUSIONES

La aplicación desarrollada permitirá hallar la solución a problemas lineales de caso fácil, utilizando el AS. Además, tanto la regla de Bland (codificada para la prevención del fenómeno de ciclaje) como la técnica de descomposición LU, permitirán contar con un software computacionalmente estable.

La aplicación presenta una interfaz general de fácil uso, esto debido, entre otros factores, a que los componentes de la barra de menú y herramientas poseen imágenes representativas a las funciones que desempeñan. De igual manera, la incorporación de teclas de acceso rápido y caracteres mnemónico permitirán que el usuario de la aplicación pueda realizar su trabajo de forma simple y eficiente.

Se resolvieron modelos de hasta setecientos cincuenta (750) variables originales por mil quinientas (1500) restricciones. Cuando estos valores se incrementan, la aplicación requiere más tiempo de cómputo para realizar su trabajo; sin embargo, es posible disminuir este tiempo a medida que se disponga de una computadora con mayor capacidad de procesamiento.

El enfoque ágil de desarrollo de software representa una alternativa viable para la puesta en marcha de proyectos de investigación y desarrollo de software, donde no se tengan claramente definidos los requerimientos iníciales y/o donde estos sean de naturaleza cambiante. Una interacción continua y directa entre desarrolladores y clientes es un atributo clave para alcanzar el éxito.

Las fases, técnicas y herramientas propuestas por la metodología XP y empleadas para esta investigación, se adaptaron a los requerimientos recabados a lo largo de todo del proceso de desarrollo, esto permitió entre otros aspectos, el diseño de una interfaz

simple, fácil de usar y acorde a las necesidades manifestadas por los usuarios.

Al ser independiente del ciclo de desarrollo, UML pudo adaptarse de forma acertada al proyecto. Además, gracias a las ventajas que ofrece dicho lenguaje, el modelado de cada uno de los módulos fue completo, permitiendo localizar posibles fallas en el diseño antes de su implementación.

RECOMENDACIONES

Construir e incorporar al software los módulos para resolver problemas lineales de caso difícil y mediante el método gráfico.

Poner a disposición de los estudiantes de la licenciatura en informática el código ejecutable del software (.jar), para que pueda ser descargado vía Web.

Ampliar la capacidad general de la aplicación, para que permita resolver modelos lineales con un mayor número de variables y/o restricciones.

Realizar revisiones periódicas al software, que permitan mantener actualizada la información teórica que este ofrece, así como incluir nueva documentación si así se amerita.

BIBLIOGRAFÍA

1. Pressman, R. 2005. *Ingenieria de software. Un enfoque práctico.* Sexta edición. Editorial McGraw-Hill. Madrid.

2. Taha, H y otros. 2004. *Investigación de operaciones*. Séptima edición. Editorial McGraw-Hill. México DF.

3. Arreola, J. y Arreola, A. 2003. *Programación Lineal: una introducción a la toma de decisiones cuantitativa*. Cengage Learning Editores. México DF.

4. Bazaraa, M. 1999. *Programación lineal y flujo de redes.* Segunda edición. Editorial imusa. México DF.

5. Ayuso, M. 2007. *Introducción a la Programación Lineal.* Facultad de Ciencias de la UNAM. México DF.

6. Moya, M. 2003. *Investigación de operaciones 1*. Tercera edición. Editorial Universidad Estatal a Distancia. San José, Cora Rica.

7. Lieberman, G. y Frederick H. 2004. Investigación de Operaciones. Séptima edición. Editorial McGraw-Hill. México DF.

8. Gomollón, F.1996. *Ejercicios de Investigación de Operaciones*. Editorial ESIC. Madrid.

9. Pilar, J. y Ruiz, A. 2003. *Investigación operativa para ingenieros*. Editorial Universidad Politécnica de Valencia. Madrid.

10. Sala, R. y Mocholi, M. 1993. *Programación Lineal: metodologías y problemas*. Editorial Tébar Flores SL. Albacete, España.

11. Cordero, A. Hueso, J. y Terregosa, J. 2004. *Calculo numérico: teorías y problema.* Editorial Universidad Politécnica de Valencia. España.

12. Lockiby, J. 1997. Análisis de la demanda de harina de pescado en un modelo de programación lineal. Trabajo de pregrado. Departamento de Matemáticas, Universidad de Oriente, Cumaná.

13. Anselmi, A. 2001. Una solución al problema de agentes de venta a través del procesamiento de datos. Trabajo de pregrado. Departamento de Matemáticas,

14. Kazanjian, V. 2006. Software para resolver el problema del coloreado de un grafo, utilizando la metaheurística búsqueda tabú con intensificación y diversificación. Trabajo de pregrado. Programa de la Licenciatura en Informática, Universidad de Oriente, Cumaná.

15. UDO-Sucre. SF. "El Núcleo de Sucre de la Universidad de Oriente". "Sucre.udo". http://www.sucre.udo.edu.ve/index.php?option=com_content&task=category&sectionid=23&id=62&Itemid= 14 (14/10/2010).

16. UDO-Sucre. SF. "Misión y visión del programa de la Licenciatura en Informática de la UDO, "Sucre.udo". http://licinformatica.sucre.udo.edu.ve/mision_vision.php (18/08/2011).

17. Prawda, J. 2004. *Métodos y modelos de investigación de operaciones*. Noriega Editores. México DF.

18. Cierra, D. 2004. *Métodos cualitativos para la toma de decisiones*. Ediciones Gestión 2000. España.

19. Tamayo y Tamayo, M. 2003. *El Proceso de la Investigación Científica*. Cuarta edición. Editorial Limusa S.A. México D.F.

20. Balbas, A. y Gil, J. 1990. *Programación Matemática.* Segunda edición. Editorial AC. Madrid.

21. Martines, F y Martíbn, G. 2003. *Introducción a la programación estructurada en C*. Editorial Maite Simón. Valencia, España.

22. Graham, I. 1996. *Métodos Orientados a objetos.* Editorial Addison Wesley. Wilmington, Delaware, EEUU.

23. Deitel, H. 2004. *Cómo programar en C y Java.* Editorial Prentice Hall. México D.F.

24. Joyanes, L. y Fernández, M. *Java 2 Manual de programación*. Editorial Mc Graw Hill. Madrid.

25. Alarcón, R. 2000. *Diseño Orientado a Objetos con UML.* Editorial Grupo Eidos. Madrid.

26. Rumbaugh, J y otros. 2000. *El lenguaje unificado de modelado. Manual de*

referencia. Editorial Addison Wesley. Madrid.

27. Wells, D y otros. 2009. "The Rules of Extreme Programming". "Extreme programming: A Gentle Introduction". <http://www.extremeprogramming.org/ rules.html> (11/08/2009).

28. Canós, J. Letelier, P. y Penadés, C. 2003. *Metodologías ágiles en el desarrollo de software. Actas. Taller realizado en el marco de las VIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2003*. Universidad Politécnica de Valencia. Alicante, España.

29. Calabria, L. y Píriz, P. 2003. *Metodología XP. Cátedra de Ingeniería de Software*. Facultad de Ingeniería de la Universidad ORT. Montevideo, Uruguay.

30. Campos, M. 2004. *La Programación Extrema: conceptos y prácticas.* Universidad Tecnológica Nacional Facultad Regional. Buenos Aires, Argentina.

31. Cohn, M. 2004. *User stories applied: for agile software development*. Editorial Addison Wesley. Boston, MA.

32. Hazzan, O. y Dubinsky, Y. 2008. *Agile software engineering*. Editorial Springer. EEUU.

33. Rojas, S. y Mata, M. 2005. *Compiladores: traductores y compiladores con Lex/Yacc, JFlex/cup y JavaCC*. Universidad de Málaga. España.

34. Galvis, A. 1992. *Ingeniería de Software Educativo*. Ediciones Uniandes.

35. Gosling, J. Joy, B. Steele, G. y Bracha, G. 2005. *The Java. Language specification.* Addison Wesley. California, USA.

36. Beck, K. 1999. *Extreme Programming explained*.Addison-Wesley, USA.

37. Sommerville, I. 2005. *Ingeniería de software*. Séptima edición. Editorial Addison Wesley. Madrid.

APÉNDICES

Apéndice A. Formato de las entrevistas y cuestionarios

Formato de la entrevista

Entrevista elaborada con la finalidad de obtener los requerimientos iníciales, que permitan el desarrollo de un software funcional y enfocado a resolver problemas de Programación Lineal, utilizando el Algoritmo Simplex (caso fácil). Dicho software estará dirigido a los estudiantes y personal docentes del área de Optimización de la Universidad de Oriente. Las preguntas realizadas se muestran en el Cuadro A1.

CUADRO A1. Preguntas formuladas durante la entrevista.

Nº	**Preguntas**
1	¿Qué opinión tiene Ud. sobre el contenido programático de las asignaturas del área de Optimización?
2	¿Qué apreciación considera Ud. tienen sus alumnos con respecto a la materia Programación Lineal (230- 3164)?
3	En su opinión, ¿cuál es el estado actual de los paquetes computacionales utilizados para resolver problemas lineales, disponibles en los laboratorios de informática?
4	Describa de manera general, los pasos para resolver un problema lineal.
5	Durante las prácticas de laboratorio, ¿sus estudiantes le han comentado sobre alguna debilidad con los software utilizados para resolver problemas lineales?
6	¿Con qué frecuencia Ud. hace uso de estos paquetes?

CUADRO A1. Continuación.

Nº	Pregunta
7	Desde su punto de vista, ¿cuál sería el objetivo principal del desarrollo de un nuevo software que resuelva problemas lineales?
8	¿Cree Ud. que los paquetes disponibles en los laboratorios, para resolver problemas lineales, ayudan a reforzar los conceptos dados en clase?
9	¿Cuáles beneficios considera Ud. traería la implementación de un nuevo software para resolver problemas lineales?
10	Señale, ¿qué nuevas características se pueden incorporar a un software para resolver problemas lineales?
11	¿Le gustaría incluir un módulo teórico en un software que resuelva problemas lineales?
12	¿Qué recomendaciones sugiere Ud. para el desarrollo de un nuevo software que resuelva problemas lineales?

Formato del cuestionario

Encuesta realizada con la finalidad de recabar información para diagnosticar la necesidad del desarrollo de un software, para resolver problemas lineales utilizando el Algoritmo Simplex (caso fácil). Dicho software estará dirigido especialmente a los estudiantes y profesores del área de Optimización de la UDO.

En el Cuadro A2, se muestran una serie de planteamientos, los cuales fueron utilizados para detectar el grado de dificultad que el encuetado considera presentan las asignaturas del área de Optimización, así como la opinión con respecto a las herramientas, con que cuenta la UDO, para apoyar los procesos de enseñanza y de aprendizaje.

CUADRO A2. Preguntas del cuestionario.

Nº	Pregunta
1	¿Cursa Ud. actualmente alguna asignatura del área de la Investigación de Operaciones? Si [ ] No [ ]
2	De ser afirmativa la respuesta anterior, ¿cuál asignatura está cursando? [ ] Programación Lineal (230-3164) pensum nuevo. [ ] Investigación de Operaciones (230-4194) pensum nuevo (electiva).
3	De ser negativa la respuesta a la pregunta número 1, ¿cuáles de estas asignaturas, del área de la Investigación de Operaciones, cursó en semestres anteriores? [ ] Programación Lineal (230-3164) pensum nuevo. [ ] Investigación de Operaciones (230-4194) pensum nuevo (electiva). [ ] Investigación de Operaciones (230-4714) pensum viejo. [ ] Investigación de Operaciones II (230-5614) pensum viejo (electiva).
4	¿Cuál es el nivel de dificultad que Ud. considera presentan las asignaturas del área de Investigación de Operaciones? Bajo [ ] Medio [ ] Alto[ ]
5	¿Cuál es el nivel de dificultad que Ud. considera posee el aprendizaje del Algoritmo Simplex? Bajo [ ] Medio [ ] Alto[ ]
6	¿Considera que la Programación Lineal es importante para resolver problemas de la vida diaria? Si [ ] No [ ]

CUADRO A2. Continuación

Nº	Pregunta
7	Dentro del contenido programático de las asignaturas del área de Investigación de Operaciones, se hace uso de un conjunto de herramientas para resolver problemas lineales, las cuales son: TORA, TORA-Win y LINDO. ¿Cuál es su opinión acerca del funcionamiento y características de estos paquetes computacionales?
8	¿Presentó algún inconveniente al usar alguno de estos paquetes computacionales? Si [ ] No [ ]
9	¿Los paquetes computacionales, utilizados para resolver problemas lineales, le han ayudado a reforzar los métodos vistos en clase?
10	¿Cuál sistema operativo Ud. utiliza con mayor frecuencia? Windows XP / Vista / Seven [ ] Ubuntu /Debian [ ] Otro: _____
11	¿Cuáles características considera Ud. debe poseer un software, para resolver problemas lineales, dirigido a los estudiantes de la carrera de la Licenciatura en Informática de la UDO?
12	¿Cuál es su opinión acerca de las herramientas utilizadas de apoyo para los procesos de enseñanza y de aprendizaje?
13	¿Considera necesaria la inclusión de un módulo de ejercitación y práctica, que ayude a reforzar los conocimientos que se adquieren en clase? Si [ ] No [ ]
14	¿Qué forma considera Ud. más recomendable para introducir los datos de un problema lineal? Con una tabla [ ] Con un editor de texto[ ]
15	¿Le gustaría contar con un software, ajustado a los requerimientos actuales, para resolver problemas lineales? Si [ ] No [ ]

Apéndice B. Cartas de validación del cuestionario

En las figuras B1, B2 y B3, se muestran las cartas de validación para el cuestionario aplicado durante la fase de recolección de requerimientos. En ellas se avala que dicho instrumento cumple con los parámetros suficientes para ser utilizado en esta investigación.

Figura B1. Carta de validación del cuestionario firmada por el Prof. Armando Anselmi

VALIDACIÓN

Por medio de la presente certifico que he revisado exhaustivamente el instrumento para la recolección de datos y, por cumplir con los parámetros suficientes, lo valido para que sea utilizado en el Trabajo de Grado intitulado "Software para la resolución de problemas de programación lineal aplicando el Algoritmo Simplex (caso fácil)", que le servirá al Br. Renan Salazar para optar al título de Licenciado en Informática.

Prof. Armando Anselmi

Figura B2. Carta de validación del cuestionario firmada por la Profa. Lisbeth Fernández

VALIDACIÓN

Por medio de la presente certifico que he revisado exhaustivamente el instrumento para la recolección de datos y, por cumplir con los parámetros suficientes, lo valido para que sea utilizado en el Trabajo de Grado intitulado "Software para la resolución de problemas de programación lineal aplicando el Algoritmo Simplex (caso fácil)", que le servirá al Br. Renan Salazar para optar al título de Licenciado en Informática.

Profa. Lisbeth Fernández

Figura B3. Carta de validación del cuestionario firmada por la Profa. Nancy Ruiz

VALIDACIÓN

Por medio de la presente certifico que he revisado exhaustivamente el instrumento para la recolección de datos y, por cumplir con los parámetros suficientes, lo valido para que sea utilizado en el Trabajo de Grado intitulado "Software para la resolución de problemas de programación lineal aplicando el Algoritmo Simplex (caso fácil)", que le servirá al Br. Renan Salazar para optar al título de Licenciado en Informática.

Profa. Nancy Ruiz

Apéndice C. Historias de usuario recabadas

Las tarjetas HU son la forma de capturar las funcionalidades requeridas para construir software, bajo la metodología XP. En ellas, se describen brevemente las características que el sistema debe tener, desde la perspectiva del cliente. En los Cuadros C1, C2 hasta la C29, se muestran las tarjetas HU capturadas durante el desarrollo de la aplicación.

CUADRO C1. Historia de usuario Iniciar la Aplicación.

Número	1
Nombre	Iniciar la Aplicación.
Puntos asignados	5
Prioridad del cliente	Alta
Riesgo de desarrollo	Alto
Descripción	La aplicación carga todos los componentes y muestra el avance del proceso en forma gráfica. Al finalizar, se debe desplegar la ventana principal de la aplicación.
Notas	El tiempo de carga debe ser razonable. La interfaz de la aplicación debe adaptarse a la resolución de la pantalla.

CUADRO C2. Historia de usuario Acceder a una Función desde la Barra de Menú.

Número	2
Nombre	Acceder a una Función desde la Barra de Menú.
Puntos asignados	2

CUADRO C2. Continuación

Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	La aplicación contará con una barra de menú, la cual agrupará todas las funcionalidades disponibles para el usuario.
Notas	Los componentes más utilizados deben contar con una combinación de teclas de acceso rápido. Usar íconos representativos para los elementos de los menús.

CUADRO C3. Historia de usuario Acceder a una Función desde la Barra de Herramientas.

Número	3
Nombre	Acceder a una Función desde la Barra de Herramientas.
Puntos asignados	2
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	La aplicación contará con una barra de herramientas, la cual agrupará las funcionalidades más utilizadas por el usuario.
Notas	Las imágenes utilizadas deben ser de alto contraste y corresponderse a la función que representan. El orden de los elementos en la barra de herramientas debe ser el mismo en la barra de menú.

CUADRO C4. Historia de usuario Cambiar Tamaño de Fuente (Barra de Zoom).

Número	4
Nombre	Cambiar Tamaño de Fuente (Barra de Zoom).
Puntos asignados	1
Prioridad del cliente	Baja
Riesgo de desarrollo	Bajo
Descripción	El usuario podrá cambiar el tamaño de la fuente de la ventana donde se plantean los modelos.
Notas	El indicador de la barra de zoom debe ajustarse, automáticamente, cuando el usuario cambie de ventana. Esta función sólo estará disponible para las ventanas que contengan modelos escritos en forma algebraica.

CUADRO C5. Historia de usuario Crear Ventana.

Número	5
Nombre	Crear Ventana.
Puntos asignados	1
Prioridad del cliente	Alta
Riesgo de desarrollo	Medio
Descripción	El usuario podrá crear una o varias ventanas dentro de la aplicación. Éstas tendrán un área de texto (para plantear modelos en forma algebraica) o una tabla (para plantear modelos en forma tabular)

CUADRO C5. Continuación.

Notas	Cada ventana debe poseer un nombre por defecto. El usuario podrá cambiarlo si lo desea. Todas las ventanas deben poseer las opciones de maximizar, minimizar y cerrar. Si el contenido de la ventana excede el límite, debe activarse el uso de una barra de desplazamiento.

CUADRO C6. Historia de usuario Plantear Modelo en Forma Algebraica.

Número	6
Nombre	Plantear Modelo en Forma Algebraica.
Puntos asignados	3
Prioridad del cliente	Alta
Riesgo de desarrollo	Bajo
Descripción	El usuario podrá plantear modelos en forma algebraica dentro de la ventana creada para tal fin.
Notas	El usuario podrá agregar comentarios durante la definición del modelo.

CUADRO C7. Historia de usuario Cambiar Color de Fuente

Número	7
Nombre	Cambiar Color de Fuente
Puntos asignados	1
Prioridad del cliente	Baja

CUADRO C7. Continuación.

Riesgo de desarrollo	Bajo
Descripción	El usuario de la aplicación podrá seleccionar el color de la fuente para escribir los modelos lineales.
Notas	El color por defecto para el texto será el negro.

CUADRO C8. Historia de usuario Cambiar Tipo de Fuente.

Número	8
Nombre	Cambiar Tipo de Fuente.
Puntos asignados	1
Prioridad del cliente	Baja
Riesgo de desarrollo	Bajo
Descripción	Cuando el usuario seleccione esta opción, la aplicación debe recopilar todas las fuentes disponibles en la máquina donde se ejecute la aplicación y mostrarlas para que el usuario haga su selección.
Notas	El tipo de fuente por defecto será Times New Roman.

CUADRO C9. Historia de usuario Copiar, Cortar y Pegar Texto.

Número	9
Nombre	Copiar, Cortar y Pegar Texto.
Puntos asignados	1
Prioridad del cliente	Alta
Riesgo de desarrollo	Bajo

CUADRO C9. Continuación.

Descripción	La aplicación debe poseer las opciones para copiar, cortar y pegar texto. El objetivo es facilitar la tarea de plantear modelos al usuario, permitiéndole traer texto desde otro documento, sin necesidad de reescribirlo.
Notas	Estas funciones podrán ser activadas utilizando: Ctrl+C, para copiar; Ctrl+X, para cortar y Ctrl+V, para pegar.

CUADRO C10. Historia de usuario Buscar y Reemplazar Texto.

Número	10
Nombre	Buscar y Reemplazar Texto.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	La aplicación contará con la opción para localizar una cadena de texto dentro de los datos de los modelos que plantee el usuario. Además, el usuario tendrá disponible la opción para reemplazar una frase o palabra por otra.
Notas	Si no hubo resultados en la búsqueda, la aplicación debe notificarlo por pantalla. Si al realizar la búsqueda se llega al final del texto, la aplicación debe preguntar si se desea reiniciar la búsqueda desde el principio.

CUADRO C11. Historia de usuario Deshacer y Rehacer Escritura.

Número	11
Nombre	Deshacer y Rehacer Escritura.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	Cuando el usuario selecciona una de estas opciones, la aplicación verificará la pila de procesos y, si es posible, realizará la acción solicitada.
Notas	Cada ventana debe manejar su propia pila de procesos. Estas opciones sólo estarán disponibles para las ventanas que posean modelos escritos en forma algebraica.

CUADRO C12. Historia de usuario Guardar Modelo.

Número	12
Nombre	Guardar Modelo.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	Esta opción permite guardar un modelo en un archivo de texto plano, para que así el usuario de la aplicación pueda recuperarlo en un futuro.

CUADRO C12. Continuación.

Notas	El usuario podrá seleccionar la ruta donde se creará el archivo. El modelo debe guardarse en un archivo de texto plano (de extensión .txt). Se debe evitar, en lo posible, guardar un archivo en blanco. Si es el caso, la aplicación deberá notificarlo por pantalla.

CUADRO C13. Historia de usuario Imprimir Modelo.

Número	13
Nombre	Imprimir Modelo.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	Esta opción permite imprimir, en físico, un modelo que plantee el usuario dentro de la aplicación.
Notas	La aplicación debe verificar que exista una impresora conectada y disponible para realizar esta tarea. Se debe evitar, en lo posible, imprimir documentos en blanco. Si es el caso, la aplicación deberá notificarlo por pantalla.

CUADRO C14. Historia de usuario Abrir Archivo.

Número	14
Nombre	Abrir Archivo.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	Esta opción permite abrir un archivo desde la aplicación.
Notas	Sólo será posible abrir archivos de texto plano (con extensión .txt) La aplicación debe verificar si el archivo contiene un modelo lineal. Si no es el caso, mostrará un mensaje notificando sobre esta situación.

CUADRO C15. Historia de usuario Plantear Modelo en Forma Tabular.

Número	15
Nombre	Plantear Modelo en Forma Tabular.
Puntos asignados	4
Prioridad del cliente	Alta
Riesgo de desarrollo	Medio
Descripción	El usuario podrá plantear un modelo de Forma Tabular dentro de una ventana creada para tal fin.

CUADRO C15. Continuación.

Notas	El usuario debe seleccionar previamente el número de filas y columnas de la tabla. Éstas representan el total de variables y de restricciones que posee el modelo. Se podrá cambiar la dimensión de la tabla sin perder los datos ingresados hasta ese momento. Las filas y columnas de la tabla deben estar etiquetadas con un nombre significativo.

CUADRO C16. Historia de usuario Exportar a Formato .pdf

Número	16
Nombre	Exportar a Formato .pdf.
Puntos asignados	2
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	El usuario podrá crear un archivo con extensión .pdf que contenga un modelo lineal planteado dentro de la aplicación.
Notas	Se debe permitir al usuario adjuntar al documento los datos de la solución del modelo y de las operaciones realizadas por el algoritmo. Se debe evitar, en lo posible, crear un archivo en blanco.

CUADRO C17. Historia de usuario Distribuir Ventanas.

Número	17
Nombre	Distribuir Ventanas.
Puntos asignados	1
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	El usuario podrá ordenar automáticamente las ventanas abiertas, ya sea en mosaico horizontal, vertical o en cascada. El objetivo es facilitarle la búsqueda y/o visualización de un modelo en particular.
Notas	Sin observaciones.

CUADRO C18. Historia de usuario Ingresar Parámetros del Modelo.

Número	18
Nombre	Ingresar Parámetros del Modelo.
Puntos asignados	3
Prioridad del cliente	Alta
Riesgo de desarrollo	Bajo
Descripción	Una vez seleccionado el tipo de modelo a plantear, el usuario introduce el nombre, el número de variables y el número de restricciones del problema.

CUADRO C18. Continuación.

Notas	Si el modelo es de forma algebraica, la aplicación solicitará un nombre para identificar la ventana. Si el modelo es de forma tabular, la aplicación solicitará además del nombre, el número de variables y el número de restricciones del problema. Si el usuario no ingresa un nombre, la aplicación define uno automáticamente, de la forma: "Sin título" acompañado del número de la ventana que corresponda.

CUADRO C19. Historia de usuario Hallar Solución (Modelo en Forma Tabular).

Número	19
Nombre	Hallar Solución (Modelo en Forma Tabular).
Puntos asignados	5
Prioridad del cliente	Alta
Riesgo de desarrollo	Alto
Descripción	El usuario podrá, haciendo uso de la aplicación, resolver un modelo escrito en forma tabular. La aplicación tomará los datos de cada celda, creará las estructuras de entrada y realizará los procedimientos necesarios para resolver el problema.

CUADRO C19. Continuación.

Notas	Estructuras de entrada: matriz **A**, matriz **B**, vector **b**, vector **C**. La aplicación debe chequear que no existan celdas vacías. La aplicación debe chequear que el modelo planteado sea de caso fácil. Los datos que se deben reflejar en la solución son: nombre del problema, tiempo necesario para hallar la solución (en segundos), tipo de problema, valor objetivo, vector $\mathbf{X_B}$, número de variables originales, número de variables de holgura y, por último, si se estuvo en presencia de tablas degeneradas o soluciones óptimas alternativas. Se debe señalar, para cada iteración del algoritmo: la variable que entra a la base, la variable que sale de la base y las variables que forman la base.

CUADRO C20. Historia de usuario Hallar Solución (modelo en Forma Algebraica).

Número	20
Nombre	Hallar Solución (Modelo en Forma Algebraica).
Puntos asignados	5
Prioridad del cliente	Alta
Riesgo de desarrollo	Alto

CUADRO C20. Continuación.

Descripción	El usuario podrá, haciendo uso de la aplicación, resolver un modelo escrito en forma algebraica. La aplicación tomará los datos del modelo, creará las estructuras de entrada y realizará los procedimientos necesarios para resolver el problema.
Notas	Estructuras de entrada: matriz **A**, matriz **B**, vector **b**, vector **C**. Si una variable del modelo no tiene coeficiente, la aplicación le asignará el valor de uno (1). Los datos que se deben reflejar en la solución son: nombre del problema, tiempo necesario para hallar la solución (en segundos), tipo de problema, valor objetivo, vector $\mathbf{X_B}$, número de variables originales, número de variables de holgura y, por último, si se estuvo en presencia de soluciones degeneradas o de soluciones óptimas alternativas.

CUADRO C21. Historia de usuario Hallar Solución de un Modelo en Forma Guiada.

Número	21
Nombre	Hallar Solución de un Modelo en Forma Guiada.
Puntos asignados	5
Prioridad del cliente	Alta
Riesgo de desarrollo	Alto

CUADRO C21. Continuación.

Descripción	El usuario podrá, una vez planteado un modelo en forma tabular, resolverlo paso a paso. El usuario seleccionará la variable de entrada y la variable de salida en cada iteración del algoritmo. La aplicación validará que la operación a ejecutar sea la correcta, si es así, realizará los cálculos necesarios y mostrará la siguiente tabla. El objetivo es permitirle al usuario de la aplicación reforzar los conocimientos que adquiere dentro de las aulas de clases, en lo que respecta a la solución de modelos escritos en forma tabular.
Notas	Opción disponible sólo para modelos en forma tabular. Si el usuario se equivoca en su selección, la aplicación debe notificarlo por pantalla. Si se equivoca más de tres (3) veces, se mostrará un mensaje animándolo a consultar la documentación del software. La aplicación debe alentar al usuario cuando haga una selección correcta. Cuando no existan candidatos para entrar o para salir de la base, la aplicación debe notificarlo por pantalla.

CUADRO C22. Historia de usuario Generar Operaciones del Algoritmo Simplex.

Número	22
Nombre	Generar Operaciones del Algoritmo Simplex.
Puntos asignados	3
Prioridad del cliente	Media

CUADRO C22. Continuación.

Riesgo de desarrollo	Medio
Descripción	El usuario podrá seleccionar la opción de mostrar las operaciones que realice la aplicación, cuando resuelva los modelos lineales.
Notas	Los datos de las operaciones deben reflejar las iteraciones del algoritmo. El usuario podrá guardar estos datos en un archivo digital, si lo desea.

CUADRO C23. Historia de usuario Revisar la Sintaxis del Modelo.

Número	23
Nombre	Revisar la Sintaxis del Modelo.
Puntos asignados	4
Prioridad del cliente	Media
Riesgo de desarrollo	Alto
Descripción	Los modelos que plantee el usuario, desde la aplicación, deben estar escritos correctamente. Para esto, el usuario dispondrá de la opción para validar si el modelo cumple con la sintaxis pre-establecida.
Notas	La aplicación debe notificar por pantalla si el modelo lineal cumple o no con la sintaxis. Si se encuentra algún error durante el proceso de análisis, la aplicación debe señalar el presunto error e indicar la línea donde se originó.

CUADRO C24. Historia de usuario Guardar Resultados y Operaciones.

Número	24
Nombre	Guardar Resultados y Operaciones.
Puntos asignados	2
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	Una vez que la aplicación halle la solución de un modelo, estos datos se mostrarán dentro de una ventana. El usuario podrá seleccionar la opción para guardarlos en un archivo de texto plano.
Notas	El usuario podrá seleccionar la ruta donde se creará el archivo. Se debe evitar, en lo posible, crear un archivo en blanco.

CUADRO C25. Historia de usuario Seleccionar el Orden de las Tablas.

Número	25
Nombre	Seleccionar el Orden de las Tablas.
Puntos asignados	4
Prioridad del cliente	Media
Riesgo de desarrollo	Alta
Descripción	El usuario podrá elegir entre dos opciones para ordenar las columnas de las tablas simplex: orden natural y orden lexicográfico.
Notas	Sin observaciones.

CUADRO C26. Historia de usuario Consultar Definiciones y Ejemplos.

Número	26
Nombre	Consultar Definiciones y Ejemplos.
Puntos asignados	3
Prioridad del cliente	Alta
Riesgo de desarrollo	Bajo
Descripción	Permitirá consultar una serie de conceptos relacionados con la Programación Lineal, como por ejemplo: origen, áreas de aplicación, métodos, técnicas, entre otros.
Notas	Toda la documentación debe mostrarse en forma ordenada y agrupada bajo un tema en específico. El usuario contará con la opción para imprimir todo o parte del texto. Incorporar recomendaciones de libros y páginas Web que traten temas relacionados con la Programación Lineal.

CUADRO C27. Historia de usuario Consultar Información del Proyecto.

Número	27
Nombre	Consultar Información del Proyecto.
Puntos asignados	1
Prioridad del cliente	Baja
Riesgo de desarrollo	Bajo

CUADRO C27. Continuación.

Descripción	La aplicación dispondrá de una opción para mostrar la información del proyecto: nombre, grupo de desarrollo, datos de la Universidad, tecnologías y herramientas utilizadas durante el proceso de construcción, entre otros.
Notas	Sin observaciones.

CUADRO C28. Historia de usuario Consultar Manual de Usuario.

Número	28
Nombre	Consultar Manual de Usuario.
Puntos asignados	3
Prioridad del cliente	Media
Riesgo de desarrollo	Bajo
Descripción	La aplicación dispondrá de la opción para consultar el manual de ayuda. El objetivo es que el usuario pueda disipar dudas sobre alguna funcionalidad de la aplicación.
Notas	Toda la documentación debe mostrarse en forma ordenada y agrupada bajo un tema en específico. Se debe incluir un resumen sobre la sintaxis aceptada por la aplicación, para plantear modelos lineales. Explicar cada apartado en forma clara y resumida.

CUADRO C29. Historia de usuario Salir de la Aplicación.

Número	29
Nombre	Salir de la Aplicación.
Puntos asignados	1
Prioridad del cliente	Baja
Riesgo de desarrollo	Bajo
Descripción	El usuario contará con la opción para salir de la aplicación.
Notas	Al seleccionar esta opción, la aplicación mostrará un mensaje de confirmación. El objetivo es evitar que el usuario pierda algún dato que no hayan sido guardado hasta ese momento.

Apéndice D. Tareas

Algunas de las tarjetas HU recabadas requerían varias semanas para su implementación. Por tal motivo, fue necesario dividirlas en tareas más simples, donde su tiempo de ejecución no sobrepasara los tres (3) días de programación ideal. En los cuadros D1, D2 hasta la D48, se presentan cada una de las tareas realizadas durante la fase de construcción de la aplicación.

CUADRO D1. Tarea Diseñar de la Interfaz General de la Aplicación.

Número tarea	1
Nombre tarea	Diseñar de la Interfaz General de la Aplicación.
Fecha	01/06/2010
Número historia	1
Tipo tarea	Nueva
Estimación	3 días
Descripción	Se creará una interfaz que le permita al usuario realizar su trabajo en forma fácil y eficiente. Ésta debe poseer una barra de menú, una barra de herramientas, áreas para plantear los modelos y para mostrar los resultados. Se debe tomar como referencia los paquetes computacionales con que cuenta la Sala de Computación para resolver problemas lineales.

CUADRO D2. Tarea Diseñar Ventana de Carga de Componentes.

Número tarea	2
Nombre tarea	Diseñar Ventana de Carga de Componentes
Fecha	01/06/2010
Número historia	1
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Al iniciar la aplicación, se debe mostrar una ventana que indique el progreso de carga de sus componentes. Debe aparecer el nombre de la aplicación y el avance del proceso. Al concluir, se debe cerrar automáticamente y desplegarse la interfaz general de la aplicación.

CUADRO D3. Tarea Crear una Ventana para Plantear Modelo.

Número tarea	3
Nombre tarea	Crear una Ventana para Plantear Modelos
Fecha	01/06/2010
Número historia	5
Tipo tarea	Nueva
Estimación	3 días
Descripción	El usuario tendrá la opción para crear una ventana de trabajo, la cual deberá contener un área de texto para ingresar los datos del modelo. El usuario podrá minimizarla, maximizarla y cerrarla.

CUADRO D4. Tarea Diseñar ventana para Seleccionar Tipo de Modelo.

Número tarea	4
Nombre tarea	Diseñar Ventana para Seleccionar el Tipo de Modelo
Fecha	01/06/2010
Número historia	5
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Se diseñará una ventana que muestre las opciones para plantear un modelo. Las opciones son: Modelo en Forma Algebraica y Modelo en Forma Tabular. Una vez que el usuario haga su selección, esta ventana se cerrará automáticamente.

CUADRO D5. Tarea Permitir Abrir Varias Ventanas.

Número tarea	5
Nombre tarea	Permitir Abrir Varias Ventanas
Fecha	03/06/2010
Número historia	5
Tipo tarea	Mejora
Estimación	3 días.
Descripción	El usuario podrá abrir más de una ventana, desde la aplicación, para plantear modelos.

CUADRO D6. Tarea Diseñar Manejador de Ventanas.

Número tarea	6
Nombre tarea	Diseñar Manejador de Ventanas
Fecha	08/06/2010
Número historia	5
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Se debe desarrollar un mecanismo que permita controlar todas las ventanas abiertas por el usuario. El objetivo es poder manipularlas desde las opciones del menú y/o la barra de herramientas. Además, se debe construir un panel donde se muestren las ventanas.

CUADRO D7. Tarea Cerrar Ventana.

Número tarea	7
Nombre tarea	Cerrar Ventana
Fecha	10/06/2010
Número historia	5
Tipo tarea	Mejora
Estimación	1 día.
Descripción	Cuando el usuario seleccione la opción para cerrar una ventana, la aplicación debe mostrar un mensaje preguntándole si está seguro de realizar esta operación. El objetivo es minimizar la posibilidad de que el usuario pierda datos que no hayan sido guardados.

CUADRO D8. Tarea Ingresar Comentarios.

Número tarea	8
Nombre tarea	Ingresar Comentarios
Fecha	21/06/2010
Número historia	6
Tipo tarea	Mejora
Estimación	2 días.
Descripción	Se debe desarrollar un mecanismo que permita escribir uno o varios comentarios durante la definición de los modelos. Estas líneas de texto no formarán parte del modelo en sí y deben ser ignorados al momento de hallar la solución del problema. La forma de escribir un comentario debe ser semejante a que se utiliza en el lenguaje de programación C++.

CUADRO D9. Tarea Modificar Número de Filas.

Número tarea	9
Nombre tarea	Modificar Número de Filas
Fecha	21/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	3 días.

CUADRO D9. Continuación.

Descripción	Una vez que el usuario cree una tabla, podrá agregar o eliminar una o varias filas. Es importante que no se pierdan los datos que hayan sido ingresados hasta ese momento. Si se crea una nueva fila, la aplicación debe colocar los valores por defecto a cada celda. Si elimina se una fila, la aplicación debe mostrar un mensaje señalando que el proceso es irreversible.

CUADRO D10. Tarea Modificar Número de Columnas.

Número tarea	10
Nombre tarea	Modificar Número de Columnas
Fecha	21/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Una vez que el usuario cree una tabla, podrá agregar o eliminar una o varias columnas. Es importante que no se pierdan los datos que hayan sido ingresados hasta ese momento. Si crea una nueva columna, la aplicación debe colocar los valores por defecto a cada celda. Si se elimina una columna, la aplicación debe mostrar un mensaje señalando que el proceso es irreversible.

CUADRO D11. Tarea Seleccionar Toda una Fila o Columna de la Tabla.

Número tarea	11
Nombre tarea	Seleccionar Toda una Fila o Columna de la Tabla.
Fecha	24/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	3 días.
Descripción	El usuario podrá seleccionar toda una fila o toda una columna dando clic sobre una celda de la tabla. El objetivo es facilitarle la tarea de editar los valores de las celdas.

CUADRO D12. Tarea Restaurar Valores.

Número tarea	12
Nombre tarea	Restaurar Valores.
Fecha	25/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	2 días.
Descripción	El usuario podrá seleccionar la opción para restaurar los valores por defecto de una tabla. La aplicación debe mostrar un mensaje preguntando si se está seguro de realizar dicha operación.

CUADRO D13. Tarea Etiquetar Tabla.

Número tarea	13
Nombre tarea	Etiquetar Tabla.
Fecha	29/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Tanto las filas como las columnas de una tabla deben poseer un nombre significativo. En caso de las columnas, se debe colocar los nombres de las variables (x_1 x_2,…, x_n), además de las siglas LD para señalar la columna del vector de recursos. En lo que respecta a las filas, éstas deben enumerarse de siguiente la manera: restricción 1, restricción 2,…, restricción *m*.

CUADRO D14. Tarea Restringir Valores para las Celdas de la Tabla.

Número tarea	14
Nombre tarea	Restringir Valores para las Celdas de la Tabla.
Fecha	29/06/2010
Número historia	15
Tipo tarea	Nueva
Estimación	3 días.

CUADRO D14. Continuación.

Descripción	Las celdas de la tabla deben contener el valor adecuado. Si el usuario intenta ingresar un valor que no es el correcto, se debe mostrar un mensaje indicándole el problema e invitarlo a consultar la documentación.

CUADRO D15. Tarea Diseñar Ventana para Ingresar los Datos del Modelo.

Número tarea	15
Nombre tarea	Diseñar Ventana para Ingresar los Datos del Modelo.
Fecha	12/07/2010
Número historia	18
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Una vez que el usuario seleccione la opción para plantear un modelo en forma algebraica, la aplicación mostrará una ventana para que ingrese un nombre para identificar el problema.

CUADRO D16. Tarea Ingresar Dimensión de una Tabla.

Número tarea	16
Nombre tarea	Ingresar Dimensión de una Tabla.
Fecha	12/07/2010
Número historia	18
Tipo tarea	Nueva

CUADRO D16. Continuación.

Estimación	3 días.
Descripción	Si el usuario selecciona la opción para plantear un modelo en forma tabular, la aplicación mostrará una ventana para que pueda ingresar, además del nombre, el número de variables y el número de restricciones del problema.

CUADRO D17. Tarea Codificar el Algoritmo Simplex.

Número tarea	17
Nombre tarea	Codificar el Algoritmo Simplex.
Fecha	15/07/2010
Número historia	19
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Los pasos del Algoritmo Simplex deben ser llevados a código. Es importante que el proceso de hallar la solución de los modelos sea eficiente, evitando el mal uso de los recursos de la máquina. Se recomienda la utilización de arreglos y matrices para los componentes del modelo, así como también, la descomposición LU para hallar la inversa de las matrices.

CUADRO D18. Tarea Tomar los Valores de las Celdas.

Número tarea	18
Nombre tarea	Tomar los Valores de las Celdas.
Fecha	15/07/2010
Número historia	19
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Cuando el usuario seleccione la opción para resolver un modelo escrito en forma tabular, la aplicación debe recopilar cada valor de las celdas y construir las estructura de entrada del algoritmo: matriz **A**, vector **C** y vector **b**.

CUADRO D19. Tarea Verificar Contenido de las Celdas.

Número tarea	19
Nombre tarea	Verificar Contenido de las Celdas.
Fecha	19/07/2010
Número historia	19
Tipo tarea	Corrección
Estimación	3 días.
Descripción	Antes de tomar los datos de una tabla, la aplicación debe verificar que las celdas no estén vacías, si es el caso, se debe mostrar un mensaje señalando sobre esta situación.

CUADRO D20. Tarea Eliminar Comentarios.

Número tarea	20
Nombre tarea	Eliminar Comentarios.
Fecha	20/07/2010
Número historia	20
Tipo tarea	Nueva
Estimación	3 días
Descripción	Al momento de resolver un modelo escrito en forma algebraica, la aplicación debe llamar a la función encargada de buscar y eliminar los comentarios que posea dicho modelo.

CUADRO D21. Tarea Diseñar Ventana para Ingresar Nombre.

Número tarea	21
Nombre tarea	Diseñar Ventana para Ingresar Nombre.
Fecha	28/07/2010
Número historia	21
Tipo tarea	Nueva
Estimación	1 día.
Descripción	Cuando el usuario seleccione la opción para resolver un modelo en forma guiada, la aplicación mostrará una ventana para que ingrese su nombre. El objetivo es personalizar los mensajes sólo dentro de este módulo. El usuario podrá dejar el campo vació, si lo desea.

CUADRO D22. Tarea Adaptar el Código del Algoritmo Simplex.

Número tarea	22
Nombre tarea	Adaptar el Código del Algoritmo Simplex.
Fecha	28/07/2010
Número historia	21
Tipo tarea	Corrección
Estimación	1 día.
Descripción	Se deben hacer modificaciones al código del Algoritmo Simplex. La aplicación no debe determinar la variable de entrada y de salida automáticamente, sino que el usuario debe indicarle cuál es su selección. Además, Se deben desarrollar los mecanismos necesarios para verificar si la selección realizada fue la correcta.

CUADRO D23. Tarea Diseñar Ventana para la Selección de Variables.

Número tarea	23
Nombre tarea	Diseñar Ventana para la Selección de Variables.
Fecha	28/07/2010
Número historia	21
Tipo tarea	Nueva
Estimación	3 días.

CUADRO D23. Continuación.

Descripción	El usuario podrá seleccionar la variable de entrada y de salida de la base, en cada iteración del algoritmo. La aplicación mostrará una ventana para que el usuario haga su selección. Una vez ingresados los datos, la ventana debe cerrarse automáticamente.

CUADRO D24. Tarea Diseñar Ventana para Mostrar la Ayuda

Número tarea	24
Nombre tarea	Diseñar Ventana para Mostrar la Ayuda.
Fecha	18/08/2010
Número historia	26
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Se debe diseñar una ventana para mostrar los textos de ayuda con que cuenta la aplicación. En relación a su interfaz, ésta debe ser semejante a la típica ventana de contenido que posee gran parte de las aplicaciones escritorio.

CUADRO D25. Tarea Recopilar Información Teórica.

Número tarea	25
Nombre tarea	Recopilar Información Teórica.
Fecha	18/08/2010

CUADRO D25. Continuación.

Número historia	26
Tipo tarea	Nueva
Estimación	3 días.
Descripción	El módulo de bases teóricas abarca diversos temas relacionadas con la Programación Lineal: origen, definición, ejemplos, aplicaciones, algoritmos, entre otros. Toda la documentación debe estar respaldada por un autor, además, debe mostrarse de manera ordenada. El objetivo es ayudar al usuario a despejar dudas, cuando esté utilizando la aplicación.

CUADRO D26. Tarea Diseñar Barra de Menú.

Número tarea	26
Nombre tarea	Diseñar Barra de Menú.
Fecha	24/08/2010
Número historia	2
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Se debe crear la típica barra de menú que posee gran parte de las aplicaciones de escritorio. Esta barra permitirá acceder a cada una de las funcionalidades con que cuenta la aplicación.

CUADRO D27. Tarea Incorporar Funcionalidades a la Barra de Menú.

Número tarea	27
Nombre tarea	Incorporar Funcionalidades a la Barra de Menú.
Fecha	24/08/2010
Número historia	2
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Una vez definido cada uno de los elementos de la barra de menú, se deben codificar los mecanismos necesarios para poder utilizarlos. Además, se debe verificar que funcionen correctamente.

CUADRO D28. Tarea Diseñar Barra de Herramientas.

Número tarea	28
Nombre tarea	Diseñar Barra de Herramientas.
Fecha	02/09/2010
Número historia	3
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Se debe crear una barra de herramientas que permita acceder a las funcionalidades utilizadas con mayor frecuencia. Se deben separar los elementos por categorías y utilizar iconos representativos para cada uno.

CUADRO D29. Tarea Incorporar Funcionalidades a la Barra de Herramientas.

Número tarea	29
Nombre tarea	Incorporar funcionalidades a la barra de herramientas.
Fecha	02/09/2010
Número historia	3
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Una vez diseñado los botones de la barra de herramientas, se debe codificar los mecanismos necesarios para poder utilizarlos. Además, se debe verificar que funcionen correctamente.

CUADRO D30. Tarea Diseñar Ventana para Buscar y Reemplazar Texto.

Número tarea	30
Nombre tarea	Diseñar Ventana para Buscar y Reemplazar Texto.
Fecha	14/09/2010
Número historia	10
Tipo tarea	Nueva
Estimación	3 días.

CUADRO D30. Continuación.

Descripción	Esta opción sólo estará disponible para los modelos escritos en forma algebraica. Cuando el usuario seleccione esta opción, la aplicación debe mostrar una ventana que permita ingresar la cadena de texto a buscar. Cuando presione el botón Buscar, la aplicación debe señalar las coincidencias que haya encontrado. Si no hay coincidencias, se debe notificar por pantalla.

CUADRO D31. Tarea Diseñar Menús Contextuales.

Número tarea	31
Nombre tarea	Diseñar Menús Contextuales
Fecha	14/09/2010
Número historia	1
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Para facilitarle el trabajo el usuario, ciertas zonas de la aplicación deben poseer un menú contextual, el cual se mostrará presionando clic derecho sobre algún elemento o panel de la aplicación.

CUADRO D32. Tarea Diseñar Pantalla de Resultados.

Número tarea	32
Nombre tarea	Diseñar Pantalla de Resultados.

CUADRO D32. Continuación.

Fecha	14/09/2010
Número historia	1
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Cuando el usuario seleccione la opción para resolver un modelo escrito en forma tabular, la aplicación debe crear una nueva ventana y mostrar cada una de las iteraciones (en forma de tablas) que fueron realizadas. Además, se debe señalar la variable de entrada, de salida y las variables básicas en cada ciclo.

CUADRO D33. Tarea Renombrar Variables.

Número tarea	33
Nombre tarea	Renombrar Variables.
Fecha	14/09/2010
Número historia	15
Tipo tarea	Mejora
Estimación	3 días.
Descripción	El usuario podrá colocarle un nombre a cada variable de un modelo escrito en forma tabular. Sólo se debe permitir el uso de caracteres alfanuméricos, cualquier otro tipo de carácter debe ser ignorado. Al mostrar la solución del problema, deben aparecer los nombres que fueron ingresados previamente.

CUADRO D34. Tarea Definir los Elementos Individuales del Lenguaje.

Número tarea	34
Nombre tarea	Definir los Elementos Individuales del Lenguaje.
Fecha	23/09/2010
Número historia	23
Tipo tarea	Mejora
Estimación	3 días.
Descripción	Se debe definir los elementos que formarán parte de la gramática del modelo. Estos elementos son: las palabras reservadas, los operadores y los identificadores. Se recomienda que la forma de escribir los modelos dentro de la aplicación sea flexible, es decir, que sea posible escribir una misma sentencia de varias formas.

CUADRO D35. Tarea Realizar Análisis Léxico.

Número tarea	35
Nombre tarea	Realizar Análisis Léxico.
Fecha	23/09/2010
Número historia	23
Tipo tarea	Nueva
Estimación	3 días.

CUADRO D35. Continuación.

Descripción	Este mecanismo tomará, carácter por carácter, cada componente del modelo, para posteriormente definir los elementos de entrada para el análisis sintáctico. Se deben eliminar los espacios en blanco, tabuladores y retorno de carro. Además, se irá contabilizando el número de líneas leídas, para así, si se encuentra algún error, poder notificar donde está ubicado.

CUADRO D36. Tarea Realizar Análisis Sintáctico.

Número tarea	36
Nombre tarea	Realizar Análisis Sintáctico.
Fecha	23/09/2010
Número historia	23
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Este análisis toma los elementos creados por el analizador léxico y los verifica valiéndose de ciertas reglas establecidas previamente. Permitirá determinar si los elementos que componen el modelo son los correctos.

CUADRO D37. Tarea Realizar Análisis Semántico.

Número tarea	37
Nombre tarea	Realizar Análisis Semántico.
Fecha	23/09/2010
Número historia	23
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Este mecanismo detecta la validez semántica de las sentencias aceptadas por el analizador sintáctico. Sus rutinas deben realizar la evaluación de los atributos de la gramática.

CUADRO D38. Tarea Diseñar Ventana para Mostrar las Operaciones.

Número tarea	38
Nombre tarea	Diseñar Ventana para Mostrar las Operaciones.
Fecha	13/10/2010
Número historia	24
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Se debe crear un área para mostrar las operaciones realizadas durante el proceso de hallar la solución de un modelo. Además, debe poseer una opción para guardar, en un archivo digital, dichas operaciones.

CUADRO D39. Tarea Colocar las Columnas en Orden Natural.

Número tarea	39
Nombre tarea	Colocar las Columnas en Orden Natural.
Fecha	13/10/2010
Número historia	24
Tipo tarea	Nueva
Estimación	2 días.
Descripción	El usuario podrá seleccionar la opción para ordenar las columnas de una tabla en orden natural.

CUADRO D40. Tarea Colocar las Columnas en orden Lexicográfico.

Número tarea	40
Nombre tarea	Colocar las Columnas en Orden Lexicográfico.
Fecha	13/10/2010
Número historia	24
Tipo tarea	Nueva
Estimación	2 días.
Descripción	El usuario podrá seleccionar la opción para ordenar las columnas de una tabla en orden lexicográfico.

CUADRO D41. Tarea Crear Manual de Usuario.

Número tarea	41
Nombre tarea	Crear Manual de Usuario.
Fecha	18/10/2010
Número historia	28
Tipo tarea	Nueva
Estimación	3 días.
Descripción	Este documento tiene como objetivo mostrar las diferentes funcionalidades de la aplicación. Su redacción debe ser sencilla. Debe incluir un apartado que explique detalladamente la manera de plantear modelos lineales haciendo uso del software. Además, debe señalar los requisitos necesarios para ejecutar la aplicación. Básicamente, estará enfocado a dar asistencia al usuario.

CUADRO D42. Tarea Diseñar Barra de Zoom.

Número tarea	42
Nombre tarea	Diseñar Barra de Zoom.
Fecha	18/10/2010
Número historia	4
Tipo tarea	Nueva
Estimación	2 días.

CUADRO D42. Continuación.

Descripción	Este mecanismo debe permitir modificar el tamaño de la fuente de texto de una ventana. Su funcionamiento debe ser sencillo y eficiente. Se debe tomar como referencia la barra Zoom que posee el software *Microsoft Word.*

CUADRO D43. Tarea Distribuir Ventanas en Mosaico Horizontal.

Número tarea	43
Nombre tarea	Distribuir Ventanas en Mosaico Horizontal.
Fecha	25/10/2010
Número historia	17
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Al seleccionar esta opción, todas las ventanas que haya creado el usuario aparecerán en pantalla una debajo de otra.

CUADRO D44. Tarea Distribuir Ventanas en Mosaico Vertical.

Número tarea	44
Nombre tarea	Distribuir Ventanas en Mosaico Vertical.
Fecha	25/10/2010
Número historia	17
Tipo tarea	Nueva
Estimación	2 días.

CUADRO D44. Continuación.

Descripción	Al seleccionar esta opción, todas las ventanas que haya creado el usuario aparecerán una junto a la otra.

CUADRO D45. Tarea Distribuir Ventanas en Cascada.

Número tarea	45
Nombre tarea	Distribuir Ventanas en Cascada.
Fecha	25/10/2010
Número historia	17
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Al seleccionar esta opción, todas las ventanas que haya creado el usuario aparecerán superpuestas, es decir, una encima de otra.

CUADRO D46. Tarea Diseñar Ventana para Mostrar la Información del Proyecto.

Número tarea	46
Nombre tarea	Diseñar Ventana para Mostrar la Información del Proyecto.
Fecha	10/11/2010
Número historia	27
Tipo tarea	Nueva
Estimación	2 días.

CUADRO D46. Continuación.

Descripción	Cuando el usuario seleccione esta opción, la aplicación debe mostrar una ventana con información concerniente al proyecto.

CUADRO D47. Tarea Diseñar Ventana para Mostrar un Mensaje de Salida.

Número tarea	47
Nombre tarea	Diseñar Ventana para Mostrar un Mensaje de Salida.
Fecha	10/11/2010
Número historia	29
Tipo tarea	Nueva
Estimación	2 días.
Descripción	Al seleccionar la opción para salir de la aplicación, se debe mostrar una ventana notificando sobre esta operación. El objetivo es evitar que el usuario pierda alguna información que no haya sido guardada.

CUADRO D48. Tarea Calcular el Tiempo Computacional necesario para Resolver un Modelo.

Número tarea	48
Nombre tarea	Calcular el Tiempo Computacional necesario para Resolver un Modelo.
Fecha	18/11/2010
Número historia	19, 20

CUADRO D48. Continuación.

Tipo tarea	Mejora
Estimación	3 días.
Descripción	La aplicación debe calcular el tiempo que se tomó en resolver un modelo lineal. Además, debe estar expresado en segundos.

Apéndice E. Descripción de los casos de uso de la aplicación

Caso de uso: 1

Nombre: plantear modelo en forma algebraica

Actor: Modelador

Propósito: gestionar todas las funcionalidades disponibles para que el usuario pueda plantear, dentro de un ambiente gráfico, modelos lineales en forma algebraica.

Pre-condición: el modelador debe conocer los elementos que forman parte de la definición de los modelos lineales, además de estar familiarizado con la gramática aceptada por la aplicación.

Post-condición: modelo lineal en forma algebraica planteado dentro de una ventana de la aplicación.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para plantear un modelo.
- La aplicación muestra las opciones para que el usuario seleccione la forma del modelo a plantear.
- El usuario selecciona la opción Modelo Algebraico.
- La aplicación solicita el nombre del modelo.
- El usuario ingresa el nombre del modelo. Si el campo queda vacío, la aplicación asignará un nombre por defecto.
- La aplicación presenta por pantalla una ventana que contiene un área de texto vacía, la cual debe ser utilizada para plantear el modelo.
- El usuario ingresa los datos del modelo lineal.

Caso de uso: 2

Nombre: plantear modelo en forma tabular.

Actor: Modelador

Propósito: gestionar todas las funcionalidades disponibles para que el usuario pueda plantear, dentro de un ambiente gráfico, modelos lineales mediante el uso de Tablas Simplex.

Pre-condición: el modelador debe conocer los elementos que forman parte de la definición de los modelos lineales, además de estar familiarizado con la gramática aceptada por la aplicación.

Post-condición: modelo lineal en forma tabular planteado dentro de una ventana de la aplicación.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para plantear un modelo.
- La aplicación muestra las opciones para que el usuario seleccione la forma del modelo a plantear.
- El usuario selecciona la opción Modelo Tabular.
- La aplicación solicita el nombre del modelo, número de filas (restricciones del modelo) y número de columnas (variables del modelo).
- El usuario ingresa los datos solicitados. Si algún campo queda vacío, la aplicación le asignará el valor por defecto.
- La aplicación presenta por pantalla una ventana que contiene una estructura en forma de tabla, la cual debe ser utilizada para plantear el modelo.
- El usuario ingresa los datos del modelo dentro de cada celda de la tabla.

Caso de uso: 3

Nombre: guardar modelo.

Actor: Modelador.

Propósito: gestionar todas las funcionalidades disponibles para que el usuario de la aplicación pueda guardar, en formato *.txt, los modelos lineales que plantee (ya sean modelos en forma algebraica o en forma tabular), para trabajar con ellos en un futuro.

Pre-condición: el modelador debió haber planteado un modelo lineal.

Post-condición: modelo guardado en formato .txt con éxito.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para guardar un modelo.
- La aplicación verifica si existen datos dentro de la ventana, si no es así, se notificará por pantalla.
- La aplicación solicita el nombre del documento a crear y el directorio donde se almacenará el mismo.
- El usuario ingresa los datos y hace su selección.
- La aplicación verifica si existe un documento con el mismo nombre en ese directorio seleccionado, si es así, se muestra un mensaje preguntándole al usuario si desea reemplazarlo.
- El usuario hace su selección.
- La aplicación recopila los datos del modelo y crea el documento.
- La aplicación muestra un mensaje notificando sobre el éxito o fracaso de la operación.

Caso de uso: 4

Nombre: exportar modelo

Actor: Modelador

Propósito: gestionar todas las funcionalidades disponibles para que el usuario de la aplicación pueda exportar, a formato .pdf, los modelos lineales que plantee (o la solución de los mismos), ya sean modelos en forma algebraica o en forma tabular.

Pre-condición: el modelador debió haber planteado un modelo lineal.

Post-condición: modelo lineal exportado a formato .pdf con éxito.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para exportar un modelo a formato .pdf.
- La aplicación verifica si existen datos dentro la ventana, si no es así, la aplicación muestra un mensaje de notificación indicando esta situación.
- La aplicación solicita el nombre del documento a crear y el directorio donde se almacenará el mismo.
- El usuario llena los campos y hace su selección.
- La aplicación verifica si existe un documento con el mismo nombre en ese directorio, si es así, se muestra un mensaje preguntándole al usuario si desea reemplazarlo.
- El usuario hace su selección.
- La aplicación recopila los datos del modelo y crea el documento.
- La aplicación muestra un mensaje notificando el éxito o fracaso de la operación.

Caso de uso: 5

Nombre: imprimir modelo.

Actor: Modelador.

Propósito: gestionar todas las funcionalidades disponibles para que el usuario de la aplicación pueda imprimir los modelos lineales que plantee (o la solución de los mismos), ya sean modelos en forma algebraica o en forma tabular.

Pre-condición: el modelador debió haber planteado un modelo lineal.

Post-condición: modelo lineal impreso.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para imprimir.
- La aplicación verifica si existen datos dentro de la ventana, si no es así, se muestra un mensaje de notificación indicando sobre la situación.
- La aplicación procesa la información.
- La aplicación envía una señal al dispositivo de salida para imprimir el modelo.
- La aplicación muestra un mensaje notificando sobre el éxito o fracaso de la operación.

Caso de uso: 6

Nombre: revisar la sintaxis del modelo.

Actor: Modelador

Propósito: validar si el modelo lineal planteado cumple con la gramática aceptada por la aplicación.

Pre-condición: el usuario debió haber planteado un modelo lineal.

Post-condición: notificación por pantalla de los errores de sintaxis en el modelo.

Descripción:

- El caso de uso inicia cuando el modelador selecciona la opción para revisar la gramática del modelo.
- La aplicación verifica si existen datos dentro de la ventana, si no es así, se muestra un mensaje de notificación indicando esta situación.
- La aplicación verifica si el modelo planteado cuenta con los elementos básicos (tipo de problema, función objetivo, restricciones), si no es así, la aplicación muestra un mensaje de notificación indicando sobre la situación.
- La aplicación realiza el análisis léxico.
- La aplicación realiza el análisis sintáctico.
- La aplicación realiza el análisis semántico.
- La aplicación le notifica si se encontraron errores durante los procesos de análisis, si es así, muestra el error y el número de la línea donde se localizó el mismo. Si no hubo algún error, se muestra un mensaje notificando que el modelo planteado cumple con la sintaxis.

Caso de uso: 7

Nombre: visualizar operaciones del Algoritmo Simplex

Actor: Modelador

Propósito: permitir que el modelador pueda observar las operaciones realizadas por la aplicación al momento de hallar la solución del modelo.

Pre-condición: el modelador tiene la necesidad de visualizar las operaciones del Algoritmo Simplex.

Post-condición: las operaciones realizadas por la aplicación son mostradas por pantalla.

Descripción:

- El caso de uso inicia cuando el modelador selecciona la opción para ver las operaciones del Algoritmo Simplex.
- La aplicación realiza los cálculos necesarios y recopila esta información.
- La aplicación muestra por pantalla todas las operaciones llevadas a cabo.

Caso de uso: 8

Nombre: cargar modelo desde archivo.

Actor: Modelador

Propósito: permitir que el usuario pueda recuperar cualquier modelo lineal planteado previamente.

Pre-condición: el modelador tiene la necesidad de visualizar los datos de un modelo lineal almacenado en memoria secundaria.

Post-condición: modelo lineal mostrado por pantalla.

Descripción:

- El caso de uso inicia cuando el modelador selecciona la opción para abrir un archivo.
- La aplicación muestra el directorio para que el usuario pueda buscar el archivo.
- El modelador selecciona el archivo a visualizar.
- La aplicación verifica si el archivo posee la extensión correcta, si no es así, se muestra un mensaje de notificación indicando que no es posible realizar la operación.
- La aplicación verifica si el archivo posee datos que correspondan a un modelo lineal, si no es así, se muestra un mensaje de notificación indicando que no es posible realizar la operación.
- La aplicación carga el nombre y la ruta del archivo.
- La aplicación crea una nueva ventana dentro de la aplicación y muestra los datos que contiene el archivo.

Caso de uso: 9

Nombre: obtener solución del modelo

Actor: Modelador

Propósito: poder resolver los modelos lineales que plantee el usuario dentro de la aplicación, ya sean modelos en forma algebraica o en forma tabular.

Pre-condición: necesidad de hallar la solución a modelos lineales.

Post-condición: los datos de la solución del modelo son mostrados por pantalla.

Descripción:

- El caso de uso inicia cuando el modelador accede a la opción para resolver modelos.
- La aplicación verifica si existen datos dentro de la ventana, si no es así, se muestra un mensaje de notificación indicando esta situación.
- La aplicación muestra las opciones disponibles.
- El modelador realiza su selección.
- La aplicación realiza los cálculos necesarios y recopila los datos de la solución.
- La aplicación muestra los datos de la solución del modelo por pantalla.

Caso de uso: 10

Nombre: consultar documentación.

Actor: Modelador.

Propósito: poder consultar ejemplos, ejercicios propuestos y definiciones propias de la Programación Lineal. Además, acceder al manual de usuario de la aplicación.

Pre-condición: el modelador tiene la necesidad de realizar una consulta específica.

Post-condición: consulta realizada con éxito.

Descripción:

- El caso de uso inicia cuando el usuario selecciona la opción para consultar la documentación.
- La aplicación muestra los diferentes ítems que corresponden a toda la documentación del software.
- El usuario selecciona el ítem a consultar.
- La aplicación muestra el resultado de la selección.

Apéndice F. Descripción de los métodos de la aplicación

En los cuadros F1, F2 hasta la F21, se muestra la descripción de los métodos de la aplicación, identificados durante la etapa de diseño de la aplicación.

CUADRO F1. Descripción de los métodos de la clase Frame.

Método	**Descripción**
iniciar	Método que realiza el proceso de carga de los componentes de la aplicación.
minimizar	Método que permite definir las operaciones que realiza la aplicación cuando se minimiza la ventana principal.
maximizar	Método que permite definir las operaciones que realiza la aplicación cuando se maximiza la ventana principal.
cerrar	Método que permite definir las operaciones que realiza la aplicación cuando se cierra la ventana principal.
definirFuente	Método utilizado para seleccionar el tipo de fuente de los componentes de la aplicación.
definirDimension	Método que permite establecer el tamaño de la ventana principal, según la resolución del monitor.

CUADRO F2. Descripción de los métodos de la clase Barra.

Método	**Descripción**
mostrarBarra	Método que realiza los procedimientos necesarios para desplegar un componente de la aplicación.

CUADRO F2. Continuación.

Método	Descripción
ocultarBarra	Método que realiza los procedimientos necesarios para ocultar un componente de la aplicación.
agregarComponente	Permite agregar un componente específico a la barra.
cambiarEstado	Método que permite habilitar o deshabilitar un componente de la barra.
etiquetarComponente	Método que permite colocar una descripción emergente sobre un componente de la barra.

CUADRO F3. Descripción de los métodos de la clase BarraDeMenu.

Método	Descripción
construirMenuArchivo	Método donde se declaran y definen los elementos que conforman el menú Archivo.
construirMenuEditar	Método donde se declaran y definen los elementos que conforman el menú Editar.
construirMenuVer	Método donde se declaran y definen los elementos que conforman el menú Ver.
construirMenuResolver	Método donde se declaran y definen los elementos que conforman el menú Resolver.
construirMenuTablas	Método donde se declaran y definen los elementos que conforman el menú Tabla.
construirMenuConfiguracion	Método donde se declaran y definen los elementos que conforman el menú Configuración.

CUADRO F3. Continuación.

Método	Descripción
construirMenuAyuda	Método donde se declaran y definen los elementos que conforman el menú Ayuda.
asignarComandoRapido	Permite asignar una combinación de teclas específica para ejecutar un comando de la barra de menú.
agregarOyente	Método que permite incorporar un oyente a cada componente de la barra de menú, para la captura de eventos.
actualizar	Permite actualizar el estado de los componentes de la barra de menú.

CUADRO F4. Descripción de los métodos de la clase BarraDeHerramientas.

Método	Descripción
insertarSeparador	Permite insertar un espacio entre los componentes de la barra de herramientas.
colocarImagen	Define el ícono que muestra los componentes de la barra de herramientas
agregarOyente	Método que permite incorporar un oyente a cada componente de la barra de herramientas, para la captura de eventos.
establecerOrientacion	Método que permite establecer la dirección (vertical u horizontal) de los componentes de la barra de herramientas.
actualizar	Actualiza los componentes de la barra de herramientas.

CUADRO F5. Descripción de los métodos de la clase BarraDeEstado.

Método	**Descripción**
actualizar	Método que permite actualizar la información que muestra la barra de estado.
definirDistribucion	Permite establecer la posición dentro del panel de los componentes de la barra de estado.

CUADRO F6. Descripción de los métodos de la clase BarraDeProgreso.

Método	**Descripción**
definirValor	Método que permite establecer el valor por defecto de la barra de progreso.
establecerOrientacion	Método que permite establecer la dirección de la barra de progreso.
definirDimension	Método que hace posible definir el tamaño del panel que contiene la barra de progreso.
mostrarMensaje	Método que permite mostrar el estado de un proceso.
definirRango	Método utilizado para establecer el valor mínimo y máximo de la barra de progreso.
definirApariencia	Método que permite establecer el aspecto de los elementos de la barra de progreso.

CUADRO F7. Descripción de los métodos de la clase BarraDeZoom.

Método	**Descripción**
definirApariencia	Establecer el aspecto de los elementos de la barra de zoom.
obtenerValor	Método que retorna el valor del indicador de la barra de zoom.
agregarOyente	Método que permite incorporar un oyente para la captura de eventos.
moverIndicador	Permite mover el indicador de la barra zoom, dentro del rango previamente definido.
definirRango	Método utilizado para establecer el valor mínimo y máximo de la barra de zoom.
establecerOrientacion	Método que permite establecer la dirección (vertical u horizontal) de la barra de zoom.

CUADRO F8. Descripción de los métodos de la clase Directorio.

Método	**Descripción**
crear	Recorre el directorio de la máquina donde se ejecuta la aplicación y crea un árbol de dicho recorrido.
mostrar	Método que permite visualizar el directorio de la máquina donde se ejecuta la aplicación.

CUADRO F8. Continuación.

Método	Descripción
ocultar	Método que permite ocultar el directorio de la máquina donde se ejecuta la aplicación.
agregarOyente	Método que permite incorporar al directorio un oyente para la captura de eventos.
actualizar	Permite realizar la reconstrucción de la estructura del árbol, debido a cambios en sus elementos.
abrirRuta	Método que hace posible abrir un archivo desde el directorio, para trabajar con él.
filtarExtension	Método que permite seleccionar qué archivos mostrar en la estructura del árbol.
verificarExtension	Método que permite chequear el tipo de archivo seleccionado.
definirApariencia	Método que permite establecer el aspecto de los elementos que componen el directorio.

CUADRO F9. Descripción de los métodos de la clase PanelSolución.

Método	Descripción
mostrar	Permite desplegar el panel donde se muestran los datos de la solución de los modelos, planteados por el usuario de la aplicación.
ocultar	Método que permite ocultar el panel Solución.
definirDimension	Método que hace posible establecer el tamaño del panel.

CUADRO F9. Continuación.

Método	Descripción
agregarComponente	Permite agregar un componente específico al panel.
guardarContenido	Método que hace posible guardar, en formato .rtf, los datos que se muestren dentro del panel Solución.
borrarContenido	Método que restaura el estado por defecto del panel.
mostrarSolucion	Recopila y muestra los datos de la solución de un modelo.
mostrarOperaciones	Recopila y muestra las operaciones realizadas por el algoritmo, durante el proceso de hallar la solución de un modelo.
verificarDatos	Método que hace posible organizar y chequear los datos de la solución de un modelo.

CUADRO F10. Descripción de los métodos de la clase Ventana.

Método	Descripción
crear	Define los elementos que componen una ventana.
nombrarVentana	Método que permite establecer el nombre de la ventana activa.
agregarOyente	Método que permite incorporar un oyente a cada ventana, para la captura de eventos.
cambiarTipoDeFuente	Permite seleccionar el tipo de fuente a utilizar dentro de la ventana activa.
cambiarColorDeFuente	Permite seleccionar el color de fuente a utilizar dentro de la ventana activa.

CUADRO F10. Continuación.

Método	Descripción
cambiarColorDeFondo	Permite seleccionar el color de fondo a utilizar dentro de la ventana activa.
distribuir	Método que permite ordenar todas las ventanas en mosaico horizontal, vertical o en cascada.
minimizar	Define las operaciones que realiza la aplicación cuando se minimiza una ventana.
maximizar	Define las operaciones que realiza la aplicación cuando se maximiza una ventana.
cerrar	Define las operaciones que realiza la aplicación cuando se cierra una ventana.
incorporarDatos	Método que recopila y muestra en una ventana, los datos que posee un archivo.
verificarDatos	Encargado de procesar la entrada de datos por teclado.
verificarContenido	Devuelve verdadero si la ventana posee datos.

CUADRO F11. Descripción de los métodos de la clase ManejadorDeVentanas.

Método	Descripción
mostrar	Método que permite mostrar el panel Manejador de ventanas
ocultar	Método que permite ocultar el panel Explorador de ventanas
crearEnlace	Vincula cada elemento del panel con una ventana.
agregarOyente	Incorpora un oyente para la captura de eventos.

CUADRO F11. Continuación.

Método	Descripción
definirDimension	Método que permite establecer el tamaño del panel.
definirApariencia	Método que permite establecer el aspecto de los elementos que componen el manejador de ventanas.
obtenerVentanaActiva	Devuelve el número de la ventana activa y falso en caso contrario.

CUADRO F12. Descripción de los métodos de la clase Modelo.

Método	Descripción
abrir	Dispone de opciones para abrir un archivo ubicado en una carpeta o unidad.
guardar	Dispone de opciones para guardar un modelo en formato .txt,
imprimir	Imprime, en físico, un modelo planteado.
exportar	Método que permite exportar un modelo a formato .pdf
validar	Método que permite verificar si un modelo cumple con la sintaxis aceptada por la aplicación.
resolver	Método que hace posible hallar la solución de modelos planteados dentro de la aplicación.
copiar	Permite almacenar el contenido de la ventana activa (modelo) en el portapapeles del sistema operativo, para poder ser "pegado" o reproducido en otro lado.

CUADRO F12. Continuación.

Método	Descripción
cortar	Método que permite eliminar el contenido de la ventana activa (modelo) para ser "pegado" o reproducido en otro lado.
pegar	Método que permite reproducir el contenido almacenado en el portapapeles del sistema operativo, luego de haber sido copiado.
definirDecimales	Permite definir el número de cifras significativas para todos los valores de las variables del modelo
cargarDatos	Método que permite recopilar los datos de un modelo.
buscarTipos	Método que devuelve los tipos de modelos que el usuario puede plantear dentro de la aplicación.
cargarCampos	Método que define los campos necesarios para plantear un modelo, según el tipo seleccionado por el usuario.

CUADRO F13. Descripción de los métodos de la clase Variables.

Método	Descripción
definir	Establece la forma válida de escribir las variables de un modelo.
identificar	Verifica si un elemento del modelo lineal, planteado dentro de la aplicación, es una variable.

CUADRO F14. Descripción de los métodos de la clase PalabrasReservadas.

Método	**Descripción**
definir	Método que permite definir la forma válida de escribir las palabras reservadas de un modelo.
identificar	Método que verifica si un elemento del modelo es una palabra reservada.

CUADRO F15. Descripción de los métodos de la clase Comentarios.

Método	**Descripción**
agregarComentario	Método que permite insertar un comentario dentro de la definición del modelo.
verificar	Método que verifica si un elemento del modelo es un comentario.
eliminar	Método que suprime los comentarios que posee un modelo lineal, al momento de recopilar sus datos.

CUADRO F16. Descripción de los métodos de la clase ModeloAlgebraico.

Método	**Descripción**
buscarTexto	Método que realiza una búsqueda de un elemento dentro de la definición del modelo.
separarModelo	Método que separa el modelo algebraico en líneas para su posterior tratamiento.

CUADRO F16. Continuación.

Método	Descripción
agruparElementos	Define los elementos válidos, dada la sintaxis del modelo.
transformarCoeficiente	Transforma un número a su equivalente en real.
reemplazarTexto	Reemplaza todas las coincidencias, dada una palabra o frase.

CUADRO F17. Descripción de los métodos de la clase Analizador.

Método	Descripción
analisisLexico	Método que lleva a cabo el análisis léxico al modelo lineal planteado por el usuario.
analisisSintactico	Método que lleva a cabo el análisis sintáctico al modelo lineal planteado por el usuario.
analisisSemantico	Método que lleva a cabo el análisis semántico al modelo lineal planteado por el usuario.
recopilarErrores	Recopila y muestra los errores encontrados durante el proceso de análisis de la sintaxis del modelo.

CUADRO F18. Descripción de los métodos de la clase Dialogo.

Método	Descripción
mostrar	Método encargado de mostrar un diálogo dentro de la aplicación.

CUADRO F18. Continuación.

Método	Descripción
cerrar	Método que hace posible de ocultar un diálogo dentro de la aplicación.
crearDialogoAbrir	Diálogo que muestra las carpetas y documentos para que el usuario pueda abrir un archivo.
crearDialogoGuardar	Diálogo para guardar un documento.
crearDialogoCerrar	Diálogo de salida de la aplicación.
crearDialogoModelos	Diálogo que muestra las diferentes formas disponibles para plantear un modelo.
crearDialogoSolucion	Muestra los datos de la solución de un modelo.
verificarPropiedades	Método que permite chequear las propiedades de los archivos abiertos desde la aplicación.

CUADRO F19. Descripción de los métodos de la clase ModeloTabular.

Método	Descripción
definirParametros	Método encargado de establecer la dimensión de una tabla, además del nombre de la misma.
crearTabla	Método utilizado para crear una tabla según el número de filas y columnas, previamente ingresado por el usuario.
agregarFila	Método que permite añadir una nueva fila a una tabla.
agregarColumna	Método que permite añadir una nueva columna a la tabla.

CUADRO F19. Continuación.

Método	Descripción
eliminarFila	Método que permite eliminar una fila previamente seleccionada.
eliminarColumna	Método que permite eliminar una columna previamente seleccionada.
nombrarFilas	Método que hace posible colocarle un nombre representativo a cada fila de una tabla.
tomarValores	Método que localiza y extrae cada valor de las celdas de una tabla.
nombrarColumnas	Método creado para colocar un nombre representativo a cada columna de una tabla.
definirApariencia	Método que permite establecer la apariencia de una tabla.
restablecerTabla	Permite restaurar el valor por defecto para cada celda de una tabla.

CUADRO F20. Descripción de los métodos de la clase AlgoritmoSimplex.

Método	Descripción
definirMatrizA	Método que permite establecer el valor de cada elemento de la matriz **A**.
definirMatrizB	Método que permite establecer el valor de cada elemento de la matriz **B**.
definirVectorC	Método donde se define el vector de costo.

CUADRO F20. Continuación.

Método	**Descripción**
calcularBInversa	Método que calcula la inversa de la matriz **B** (por descomposición LU).
calcularVectorb	Método donde se define el vector del lado derecho.
calcularValorZ	Método utilizado para calcular el valor objetivo.
calcularValorW	Método que calcula el valor de la variable **w**.
calcularZj-Cj	Método que calcula el valor de los z_j - c_j.
definirVariablesBasicas	Método que establece el conjunto de variables básicas para cada iteración del algoritmo.
definirVariablesNoBasicas	Método que establece el conjunto de variables no básicas para cada iteración del algoritmo.
calcularVariableEntrada	Método que calcula la variable que entra a la base para cada iteración.
calcularVariableSalida	Método que calcula la variable que sale de la base para cada iteración.
verificarFactibilidad	Método que verifica si existen candidatos para entrar a la base.
verificarOptimilidad	Método que verifica si existen candidatos para salir de la base.
solucionesMultiples	Método que devuelve verdadero en caso de que el modelo tenga soluciones múltiples y falso en caso contrario.

CUADRO F21. Descripción de los métodos de la clase Ayuda.

Método	**Descripción**
cargarDocumento	Método que incorpora un documento de ayuda, para ser visualizado dentro de la aplicación.
buscarTexto	Método que permite realizar una búsqueda de algún tema o término, solicitado por el usuario.
posicionarCursor	Método utilizado para mover el cursor a través del texto de ayuda.
mostrarTexto	Permite mostrar la ayuda de algún tema solicitado por el usuario de la aplicación.
copiarTexto	Incorpora los procedimientos necesarios para copiar texto previamente seleccionado.
seleccionarTexto	Permite seleccionar todo o parte del texto de ayuda.
imprimirAyuda	Método creado para imprimir parte del texto de ayuda, que dispone la aplicación.

Apéndice G. Diagramas de secuencia de la aplicación

En UML, los diagramas de secuencia permiten mostrar la representación del comportamiento de un caso de uso. Constan de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar unos objetos a otros. En las figuras G1, G2 hasta la G5, se muestran los diagramas de secuencia construidos durante la fase de diseño de la aplicación.

Figura G1. Diagrama de secuencia para el caso de uso Guardar Modelo.

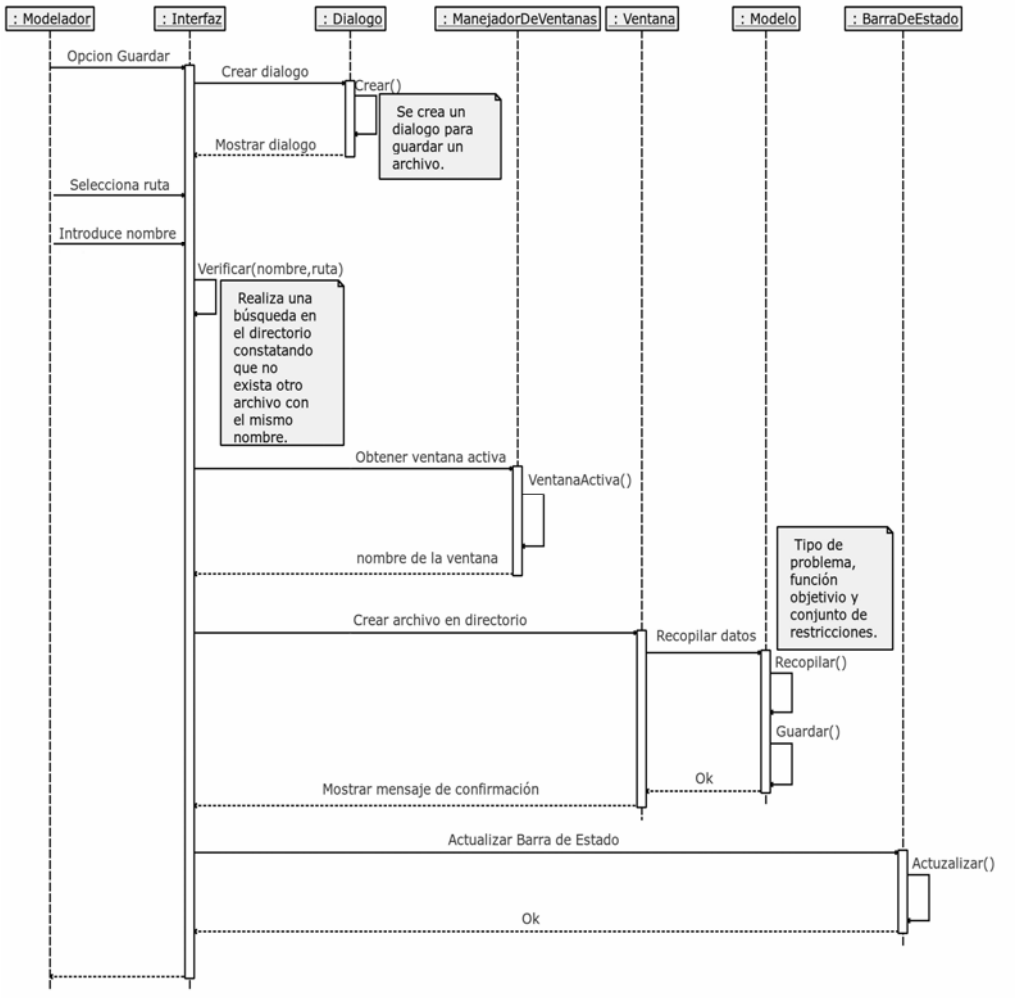


Figura G2. Diagrama de secuencia para el caso de uso Cargar Modelo desde Archivo.

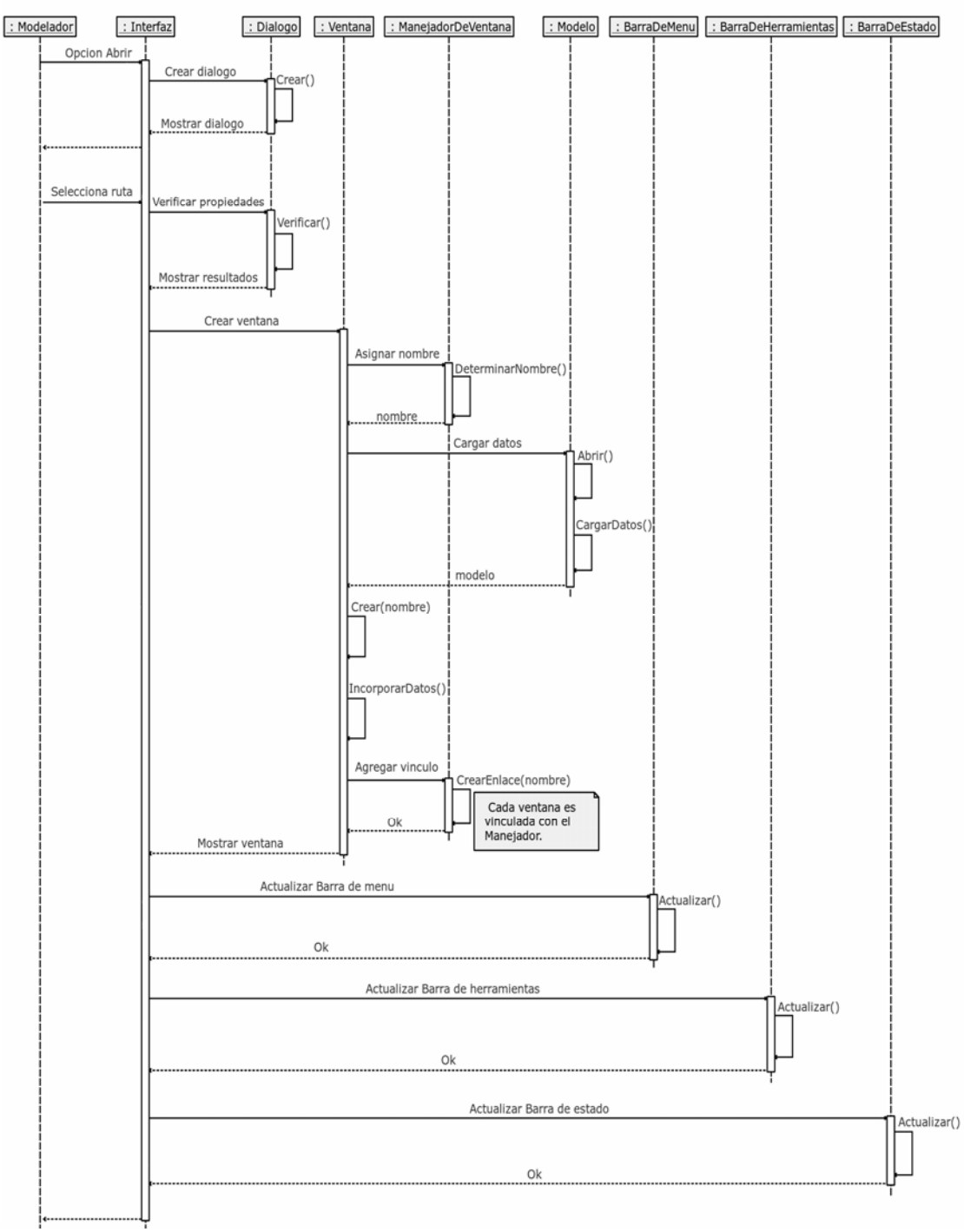


Figura G3. Diagrama de secuencia para el caso de uso Plantear Modelo.

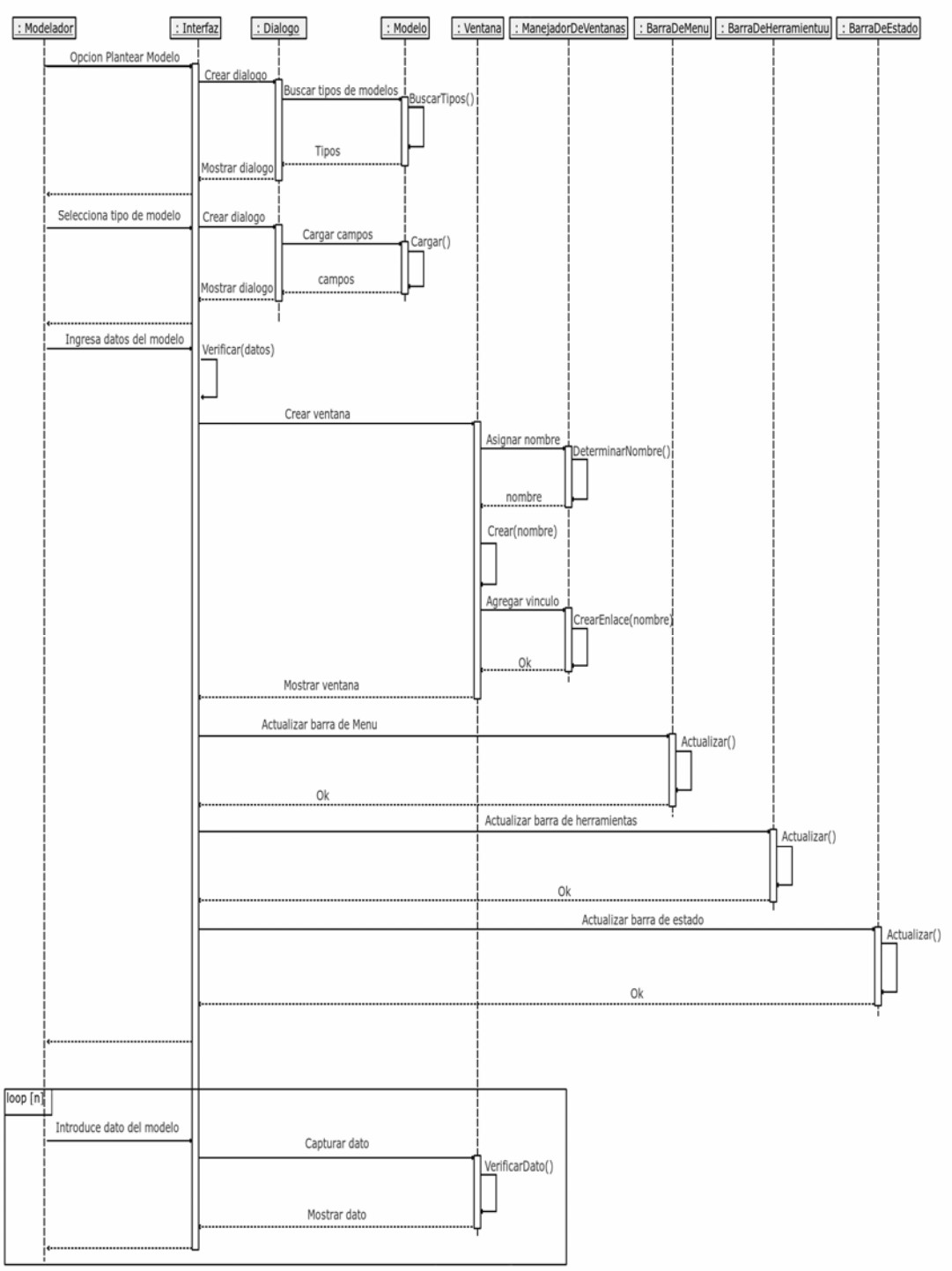


Figura G4. Diagrama de secuencia para el caso de uso Revisar Sintaxis del Modelo.

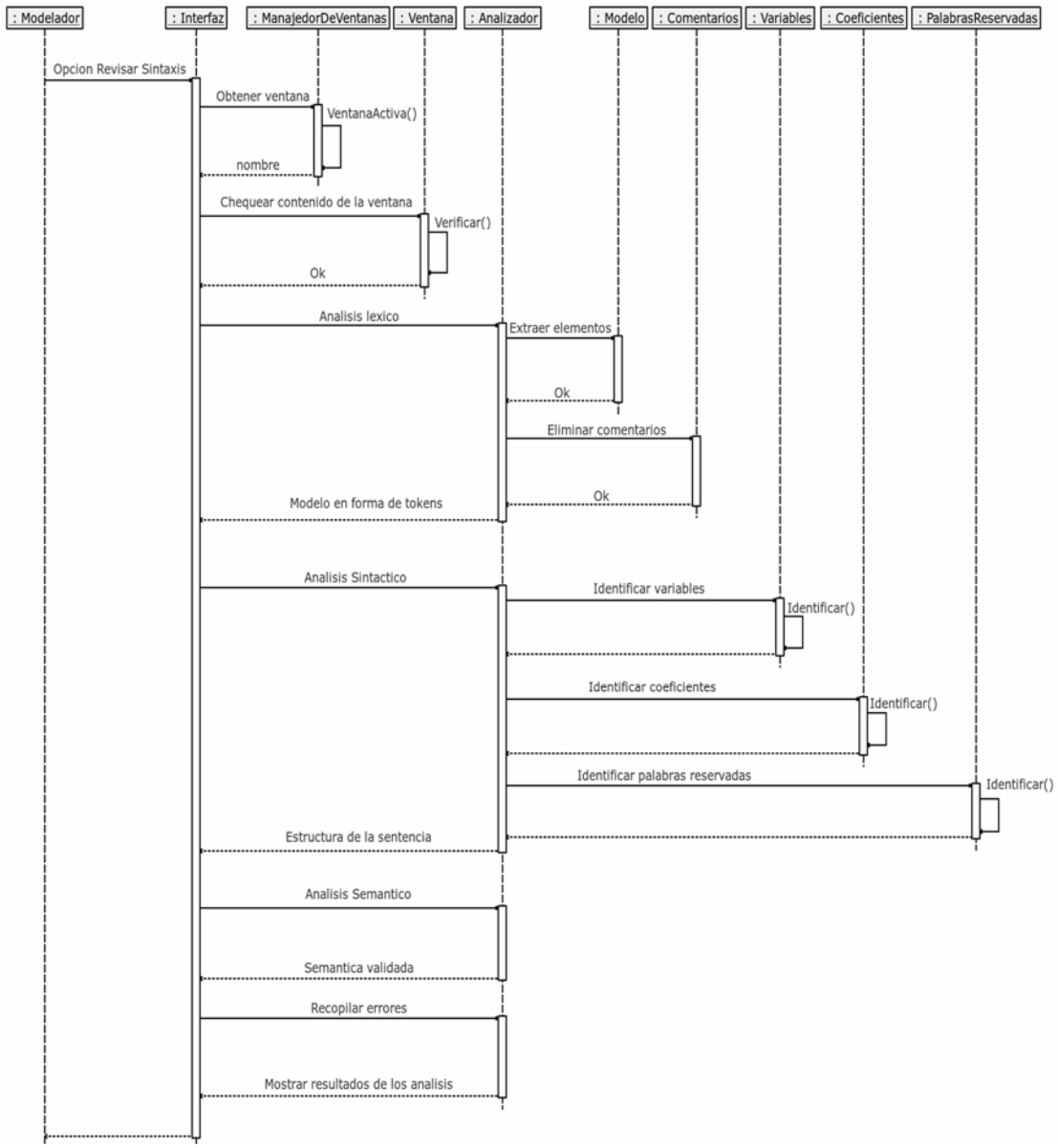
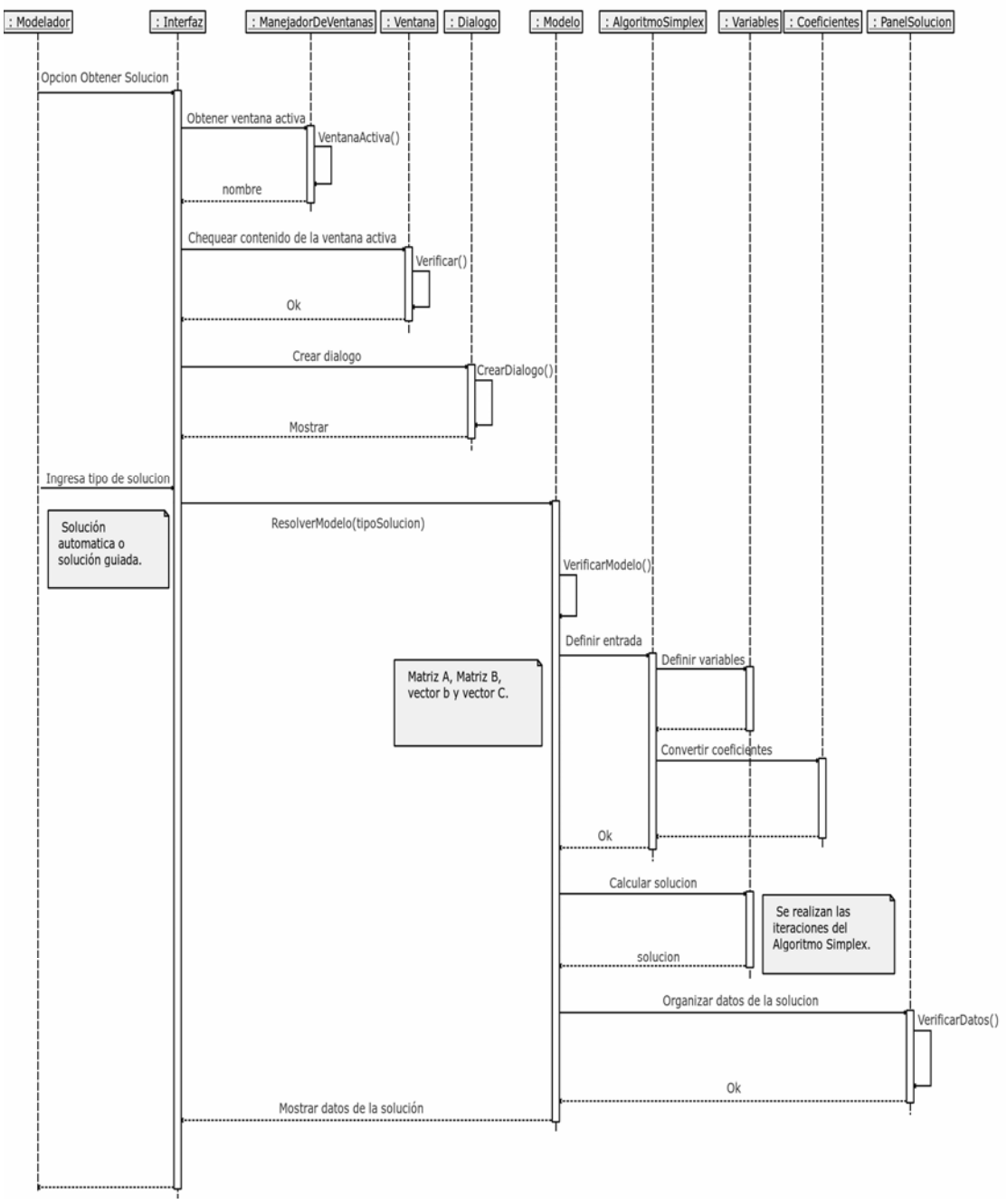


Figura G5. Diagrama de secuencia para el caso de uso Obtener Solución del Modelo.



Apéndice H. Pruebas de aceptación.

En los cuadros H1, H2 hasta la H4, se muestra la descripción de las pruebas de aceptación, las cuales fueron redactadas por el cliente durante la etapa de construcción de la aplicación.

CUADRO H1. Prueba de aceptación Ingresar Parámetros del Modelo.

Número	1
Nombre	Ingresar parámetros del modelo (Historia de usuario #18)
Entrada	Una cadena de texto que representa el nombre del modelo y dos valores enteros positivos para el número de variables y el número de restricciones.
Pruebas	Comprobar que cada carácter de la cadena de texto sea una letra [A-Z][a-z]. Comprobar que no existan espacios en blanco en la cadena. Validar que el tamaño de la cadena no sobrepase los 20 caracteres. Establecer el nombre por defecto para el modelo. El número de variables y de restricciones debe ser un valor numérico de tipo entero positivo. Chequear que los campos de textos no queden vacíos.
Salida	Si todos los valores son permitidos, la aplicación muestra una ventana donde el usuario puede plantear el modelo lineal.

CUADRO H2. Prueba de aceptación Plantear Modelo en Forma Algebraica.

Número	2
Nombre	Plantear modelo en forma algebraica. (Historia de usuario #6)
Entrada	Un modelo de programación lineal escrito en forma algebraica
Pruebas	El modelo debe empezar con el tipo de problema (minimización o maximización). Posibles valores: Min, Minimizar o Minimice (para el primer caso). Max, Maximizar o maximice (para el segundo caso). Debe ser indiferente el uso de mayúsculas o minúsculas. La función objetivo debe estar compuesta por una o más variables, separadas por el operador mas (+) u operación menos (-). Las variables deben empezar con una letra, seguida de una o más letras o números. Sin espacios en blanco ni caracteres especiales. Las variables pueden tener un coeficiente delante. Si carece de él, se asume que su valor es uno (1). Los coeficientes sólo pueden ser valores numéricos. Ejemplo 2, -10, 1.19. Si se ingresa una fracción, se debe colocar entre paréntesis. Ejemplo: (1/7). Una vez escrita la función objetivo, se debe señalar que se escribirán las restricciones del modelo. Para hacer esto la aplicación debe aceptar las cadenas “s.a:”, “s.t:”, “sa:” o “st:”, que significan “sujeto a”. Al terminar de escribir una restricción, se debe señalar su tipo. El único valor aceptado debe ser “<=” (menor o igual), ya que el modelo debe ser de caso fácil.

CUADRO H2. Continuación

Pruebas	Los valores para el vector **b** deben ser estrictamente valores numéricos. Ejemplo: 24, -6, 1.87, (1/88). Se debe ingresar al final de cada sentencia el caracter punto y coma (;). Se debe permitir la escritura de comentarios. Éstos deben comenzar con doble barra (//) seguido de uno o más caracteres de cualquier tipo. Además, no deben forman parte del modelo y deben ser ignorados al momento de hallar su solución.
Salida	Si el usuario no plantea el modelo correctamente, la aplicación debe notificar el posible error y la fila donde se originó el mismo. De lo contrario, se debe mostrar un mensaje indicando que el modelo cumple con la sintaxis.

CUADRO H3.. Prueba de aceptación Plantear Modelo en Forma Tabular.

Número	3
Nombre	Plantear modelo en forma tabular. (Historia de usuario #15).
Entrada	Un modelo de programación lineal escrito en forma tabular.
Pruebas	Las celdas correspondientes a los coeficientes de la matriz **A,** vector **C** y vector **b,** sólo deben aceptar valores numéricos. Ejemplo: 12, -1, 0.23, (1/9). Cualquier otro carácter debe ser ignorado. Si el usuario ingresa un valor incorrecto, la aplicación debe notificar el error por pantalla.

CUADRO H3. Continuación.

Pruebas	Si el usuario ingresa una restricción del tipo mayor o igual (≥) o igual (=), la aplicación debe tratar de convertir el modelo a caso fácil (multiplicar por -1 la restricción). Si el lado derecho queda negativo, la aplicación debe notificar que no posee los mecanismos para resolverlo, si no es así, debe hacer los procedimientos necesarios para replantear el modelo y resolverlo. El usuario podrá renombrar las variables del modelo. El nuevo nombre debe empezar con una letra, seguida de una o más letras o números, sin espacios en blanco ni caracteres especiales.
Salida	Si el usuario no plantea el modelo correctamente, la aplicación debe notificar el posible error. De lo contrario, se debe mostrar un mensaje indicando que el modelo cumple con la sintaxis.

CUADRO H4. Prueba de aceptación Buscar y Reemplazar Texto.

Número	4
Nombre	Buscar y reemplazar texto. (Historia de usuario #10).
Entrada	Una cadena de texto.
Pruebas	Si el usuario deja el campo vacío, la aplicación debe notificarlo por pantalla. La cadena a buscar o reemplazar puede contener cualquier tipo de caracter. La aplicación debe notificar si la búsqueda fue exitosa o no. Si lo fue, debe resaltar la palabra encontrada. Si no, debe mostrar un mensaje informado que la búsqueda no tuvo resultados.

CUADRO H4. Continuación.

Pruebas	Si el usuario desea reemplazar una cadena por otra, la aplicación debe mostrar un mensaje de confirmación.
Salida	Notificación de que la cadena de texto suministrada fue encontrada o no.

Apéndice I. Manual de usuario

UNIVERSIDAD DE ORIENTE
NÚCLEO DE SUCRE
ESCUELA DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICAS
PROGRAMA DE LA LICENCIATURA EN INFORMÁTICA

MANUAL DE USUARIO PARA EL SOFTWARE JSIMPLEX

Br. Renan Salazar
Desarrollador

Prof. José Lockiby
Asesor del proyecto

CUMANÁ, 2011

INTRODUCCIÓN

El software desarrollado, denominado JSimplex, es una aplicación de escritorio dirigida especialmente a los estudiantes y profesores del área de Optimización de la Universidad de Oriente. El mismo representa una alternativa viable al momento de resolver problemas de Programación Lineal utilizando el Algoritmo Símplex (caso fácil), ya sean modelos escritos en forma algébrica o en forma tabular. La aplicación incorpora un mecanismo encargado de validar la sintaxis de los modelos lineales planteados por el usuario. Además, ofrece un módulo para la ejercitación y práctica, enfocado especialmente a los estudiantes que se inician en el área de la Investigación de Operaciones. También cuenta con un apartado teórico para que el modelador pueda consultar ejemplos y definiciones de temas relacionados con la Programación Lineal. Todo esto construido aprovechando las ventajas que ofrece la Programación Orientada a Objetos, en el diseño de interfaces usables e intuitiva, que le permitirán al usuario de la aplicación llevar a cabo su trabajo de forma rápida y eficiente.

A continuación se presenta el manual de usuario del software JSimplex, el cual está enfocado a servir de guía para la instalación y manejo de los componentes de la aplicación, procurando así un mejor aprovechamiento de cada unas de las funcionalidades que ésta ofrece.

MANUAL DE USUARIO

Acerca de java

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia, tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía *Sun Microsystems* (adquirida por ORACLE) con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

Una de las principales características por las que *Java* se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir, que si hacemos un programa en *Java* podrá funcionar en cualquier computadora del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, entre otros. Esto lo consigue porque se ha creado la Máquina Virtual de *Java* para cada sistema, que hace de puente entre el sistema operativo y el programa de *Java* y posibilita que este último se entienda perfectamente. Pava ver más características sobre este fabuloso lenguaje de programación visite la siguiente dirección en Internet: http://java.com /en/java_in_action/

Requisitos mínimos para ejecutar la aplicación

Debido a que *Java* es un lenguaje multiplaforma, los requerimientos mínimos para ejecutar aplicaciones desarrolladas en código *Java* dependen de la plataforma de trabajo del usuario. En los cuadros I1, I2 y I3 se muestran los requerimientos de sistemas oficiales, obtenidas desde la página de la compañía desarrolladora.

CUADRO I1. Requisitos del sistema en Solaris

Plataformas	Espacio necesario en disco	Recomendaciones
• Solaris 7 • Solaris 8 • Solaris 9	Solaris-sparc: 60 MB Solaris-i586: 49 MB Solaris-sparcv9: 26,5 MB	Antes de instalar el JRE (*Java SE Runtime Environment)* debe asegurarse de que ha instalado la totalidad de las modificaciones necesarias para esta versión.

CUADRO I2. Requisitos del sistema en Linux

Plataformas	Espacio necesario en disco	Recomendaciones
• Red Hat 7.3 • Red Hat 8.0 • Red Hat Enterprise Linux WS 2.1 • Red Hat Enterprise Linux ES 2.1 • Red Hat Enterprise Linux AS 2.1 • SuSE 8.0 • TurboLinux 7.0 • SLEC 8	Mínimo de 75 MB de espacio libre en el disco	Necesitará también un procesador Pentium 166 MHz o superior y un mínimo de 32 MB de RAM.

CUADRO I3. Requisitos del sistema en Windows

Plataformas	Espacio necesario en disco	Recomendaciones
• Windows 98 (1.ª y 2.ª ediciones) • Windows ME • Windows NT • Windows 2000 • Windows XP Home • Windows XP Professional (Service Pack 1) • Windows 2003 Server Editions	Mínimo de 75 MB de espacio libre en el disco	Necesitará también un procesador Pentium 166 MHz o superior y un mínimo de 32 MB de RAM..

Para mayor información puede visitar la siguiente dirección en Internet:http://www.java.com/es/download/help/sysreq.xml

Rendimiento

El rendimiento de una aplicación escrita en código *Java* está determinado por multitud de factores, por lo que no es fácil hacer una comparación que resulte totalmente objetiva. En tiempo de ejecución, el rendimiento depende más de la eficiencia del compilador o de la Máquina Virtual de *Java*, que de las propiedades intrínsecas del lenguaje. El *bytecode* de *Java* puede ser interpretado en tiempo de ejecución por la Máquina Virtual o bien compilado al cargarse el programa o durante la propia ejecución, para generar código nativo que se ejecuta directamente sobre el hardware. Si es interpretado, será más lento que usando el código máquina intrínseco de

la plataforma destino. Si es compilado, durante la carga inicial o la ejecución, la penalización está en el tiempo necesario para llevar a cabo la compilación.

Algunas características del propio lenguaje conllevan una penalización en tiempo, aunque no son únicas de *Java*. Algunas de ellas son el chequeo de los límites de *arrays* y chequeo en tiempo de ejecución de tipos.

Instrucciones para la instalación de java

Para ejecutar una aplicación codificada en *Java*, sólo se necesita el paquete *Java SE Runtime Environment,* el cual está disponible gratuitamente desde el sitio oficial de descargas de la compañía de desarrollo de software ORACLE http://www.oracle.com/ technetwork/java/javase/downloads/index.html, o bien, se pude conseguir una copia redistribuida por terceros en distintos sitios Web, según los términos de la licencia de la plataforma *Java SE Runtime Environment*.

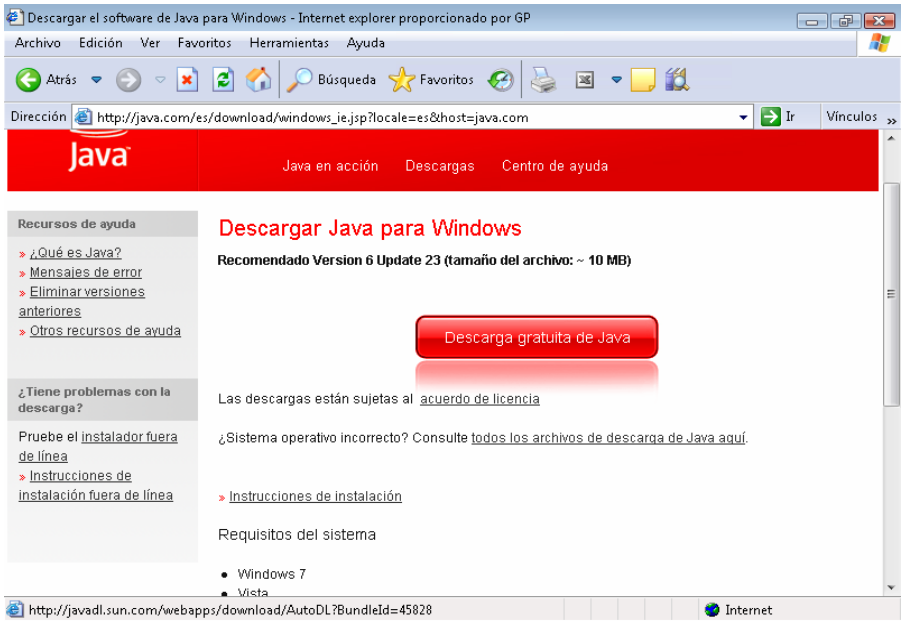
Es importante mencionar, que la aplicación desarrollada puede ejecutarse bajo cualquier sistema operativo que posea una Máquina Virtual de *Java*, por lo tanto, y como el proceso de instalación y manejo, puede variar un poco dependiendo del sistema operativo que usted esté utilizando, las imágenes que se mostrarán de ahora en adelante corresponderán a las plataformas más utilizadas según las encuestas recabadas en las primeras etapas del proceso de desarrollo: Linux y Windows, a fin de facilitar el proceso de comprensión de cada una de las funcionalidades que ofrece el software. Para mayor información visite la siguiente dirección en Internet: http://www.oracle.com/ technetwork/java/javase/re readme-182762.html.

Instalación en la plataforma Windows

Para descargar los paquetes necesarios para ejecutar la aplicación, diríjase a la siguiente dirección en Internet: http://java.com/es/download/windows_ie.jsp?locale =es&host=java.com. Las siguientes figuras le guiarán a través de todo el proceso a llevar a cabo para la instalación de *Java*.

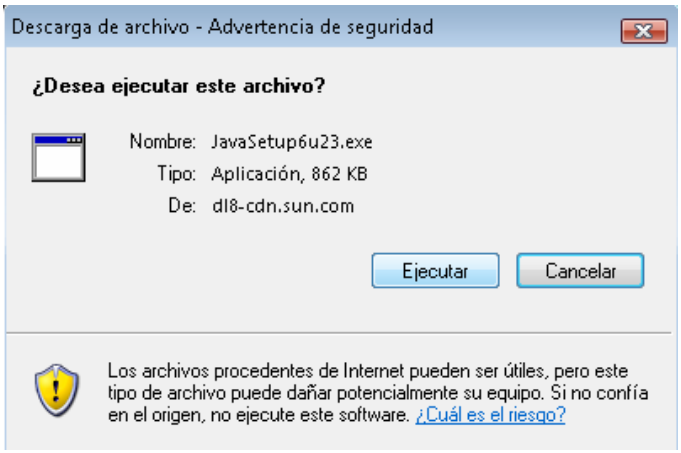
Paso 1. Ya dentro de la página, de un clic en el botón Descarga gratuita de *Java*. En la Figura I1 se muestra una captura de pantalla del sitio Web de descargas de *Java*.

Figura I1. Proceso de descarga de *Java* para Windows (paso 1)



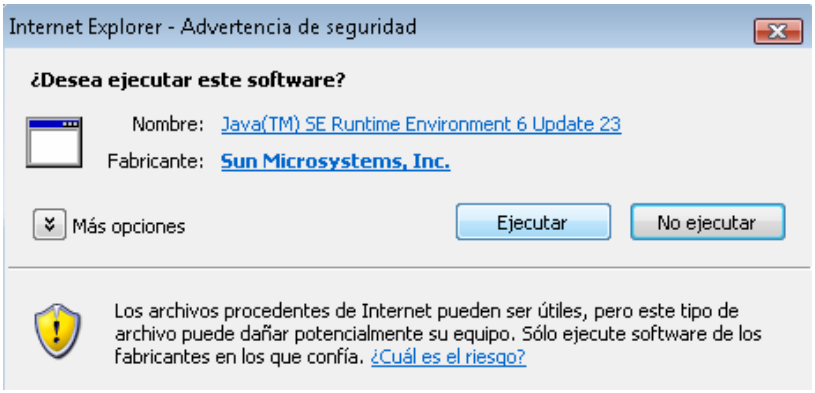
Paso 2. Se visualizará a continuación una ventana emergente, como se muestra en la Figura I2, preguntándole si desea ejecutar el archivo. De un clic en el botón Ejecutar. Seguidamente, se iniciará el proceso de descarga.

Figura I2. Proceso de descarga de *Java* para Windows (paso 2)



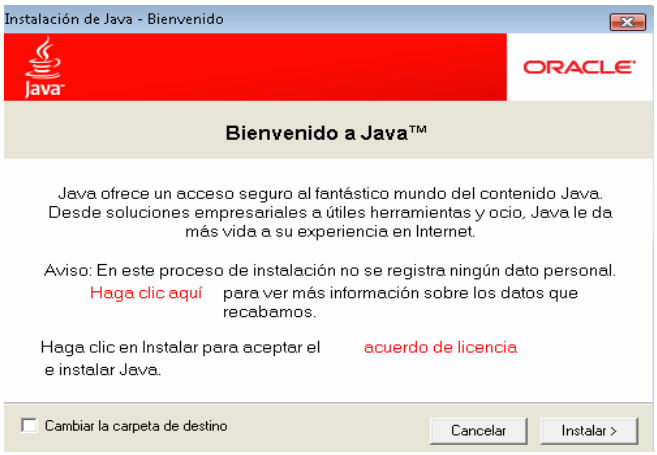
Paso 3. Una vez concluido el proceso, aparecerá ante usted una segunda ventana emergente, que se muestra en la Figura I3. Nuevamente de un clic en el botón ejecutar.

Figura I3. Proceso de descarga de *Java* para Windows (paso 3)



Paso 4. Continuando con el proceso, se mostrará una ventana con un mensaje de bienvenida a *Java* (Figura I4). De un clic en el botón Instalar para iniciar el proceso de instalación.

Figura I4. Proceso de descarga de *Java* para Windows (paso 4)



Paso 5. Concluido todo el proceso de instalación, aparecerá un mensaje indicándole que *Java* ya está listo para ejecutarse en su equipo (Figura I5). De un clic en el botón Cerrar.

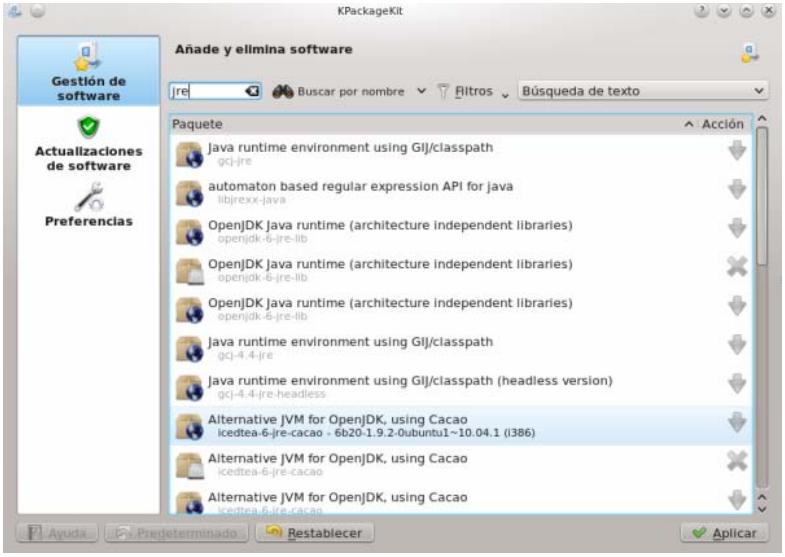
Figura I5. Proceso de descarga de *Java* para Windows (paso 5)

Para mayor información visite la siguiente dirección en Internet: http://www.oracle.com/ technetwork/ java/javase/install-windows-141940.html

Instalación en la plataforma Linux

El proceso de descarga de *Java* para plataformas Linux es el mismo que se describe en el apartado anterior. Adicionalmente, es posible instalar *Java* desde el gestor de paquetes de la distribución de Linux que usted esté utilizando. A manera de ejemplo, en la Figura I6, se muestra cómo instalar *Java* desde el gestor de paquetes KPackageKit, perteneciente a la distribución libre *Kubuntu.*

Figura I6. Interfaz del gestor de paquetes KPackageKit



Para mayor información visite la siguiente dirección en Internet: http://www.oracle.com /technetwork/java/javase/install-linux-rpm-136142.html.

Manejo de la aplicación

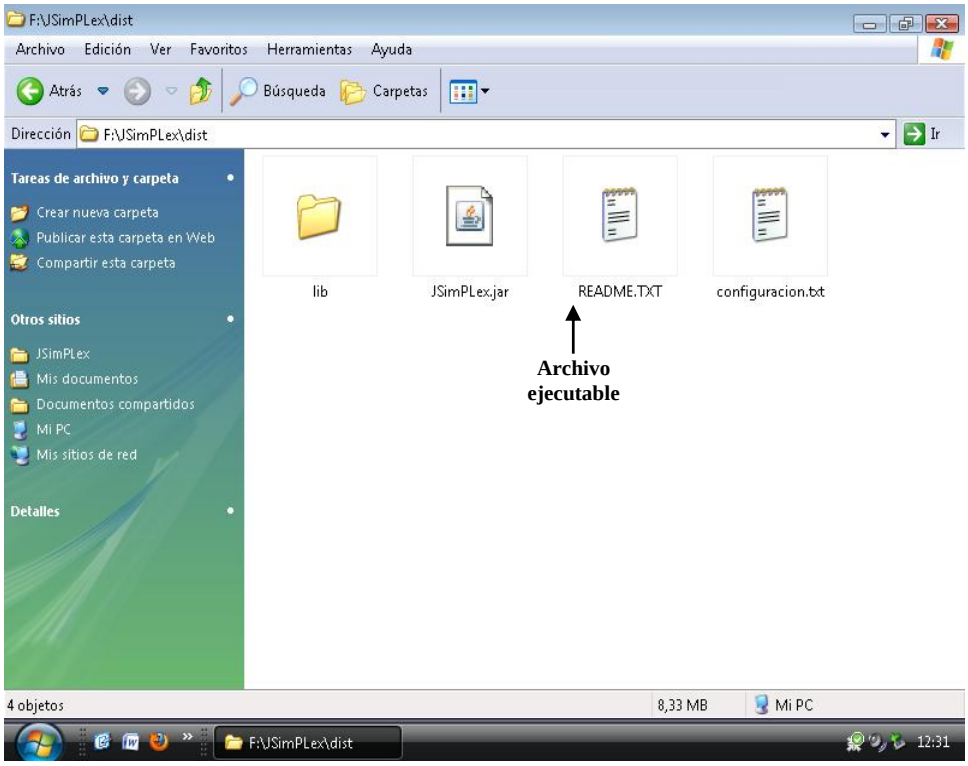
Una vez instalado satisfactoriamente los paquetes de *Java*, el usuario podrá hacer uso de las funcionalidades del software siguiendo el conjunto de instrucciones descritas

a continuación.

Iniciando la aplicación

Sitúe el cursor sobre del archivo JSimPlex.jar, como se muestra en la Figura I7, y de doble clic sobre éste; es decir, basta con localizar el icono de JSimplex y ejecutarlo para dar inicio a la aplicación.

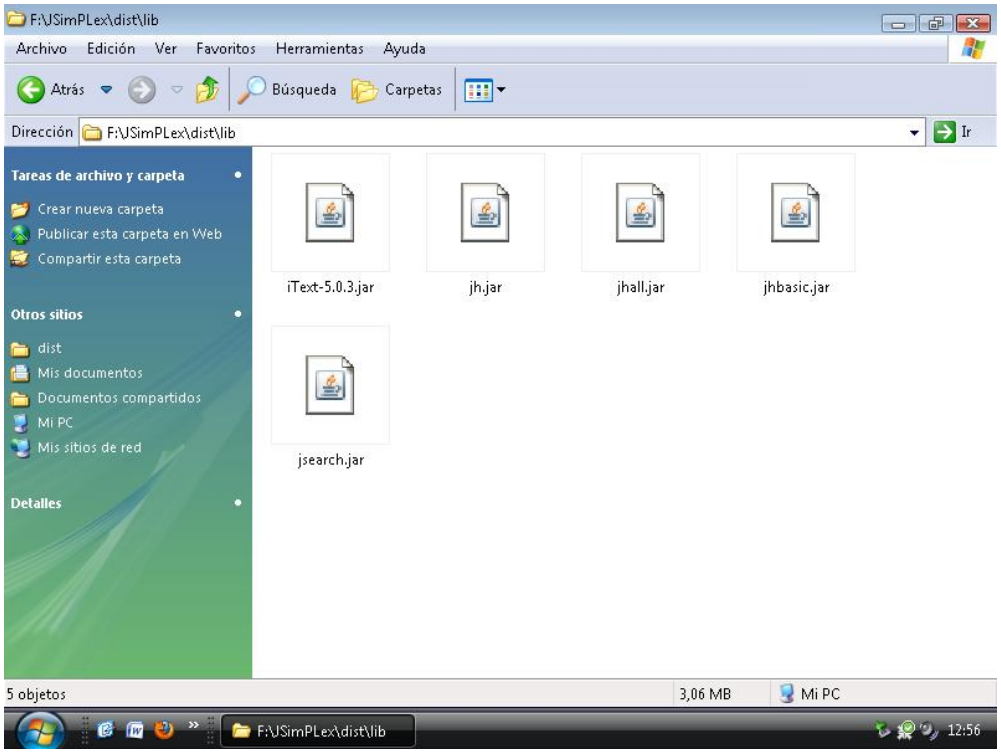
Figura I7. Iniciar la aplicación



En el mismo directorio donde se encuentre el archivo JSimPLex.jar debe localizarse, como se observa en la Figura I8, una carpeta de nombre lib, la cual contiene parte de las librerías utilizadas por la aplicación; un archivo de texto de nombre

configuración.txt, el cual guarda las preferencias del usuario (tipo y color de fuente, número de cifras significativas, entre otras); y por último, un archivo de nombre readme.txt, el cual es generado automáticamente por *Java* al compilar el código fuente de la aplicación. Es importante mencionar, que la modificación o eliminación de algunos de estos elementos puede acarrear que el software no funcione correctamente.

Figura I8. Contenido de la carpeta de nombre lib



Independientemente de la plataforma donde se esté ejecutando la aplicación, se mostrará a continuación una ventana de carga de componentes (Figura I9), la cual indica el avance de dicho proceso junto a una imagen de presentación. Una vez concluida la carga, se desplegará la interfaz general del software.

Figura I9. Ventana de carga de los componentes de la aplicación

Descripción de la interfaz general del software

A continuación se hace una descripción de la interfaz general del software, la cual se muestra en la Figura I10:

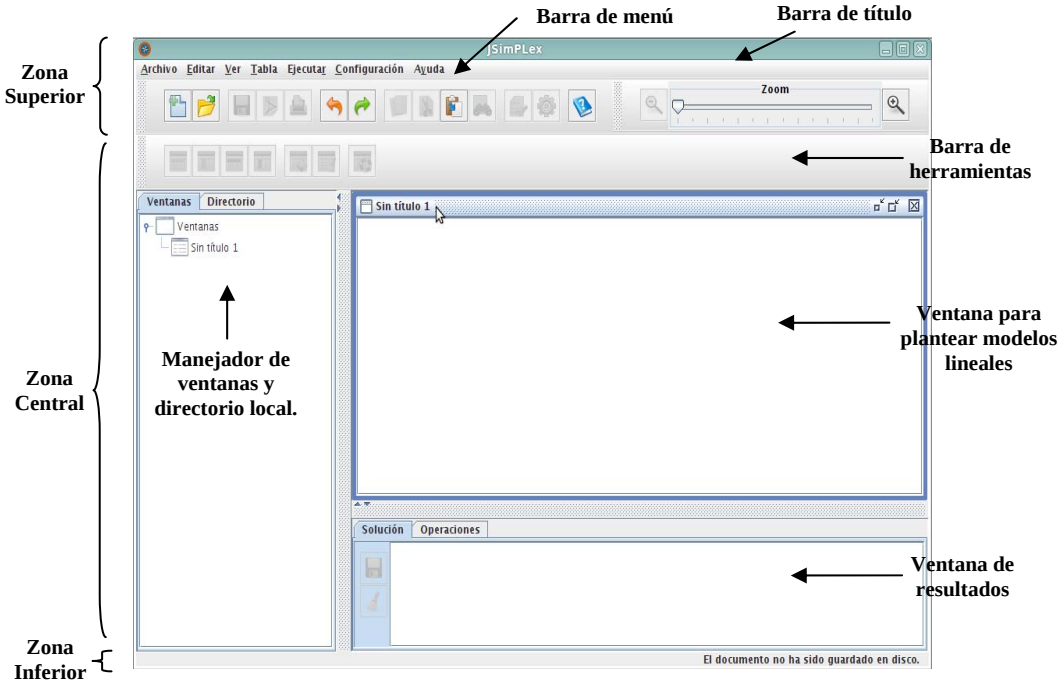
- La zona superior se encuentra dividida en dos áreas. En la primera se ubica la barra de título, donde se localiza el nombre de la aplicación junto con los botones de minimizar, maximizar y cerrar la ventana principal. En la segunda área, se encuentran las barras de menú y de herramientas, éstas reúnen las operaciones disponibles para que el usuario pueda interactuar con la aplicación.

- La zona central está compuesta por tres (3) áreas distribuidas proporcionalmente. El área izquierda está divida a su vez en dos paneles: el primero contempla un área destinada a la visualización rápida de cada una de las ventanas de trabajo creadas por el usuario y el segundo muestra el directorio local de la máquina donde se está ejecutando la aplicación. El área derecha corresponde a la ventana creada para plantear modelos lineales, esta incorpora las operaciones más utilizadas para manejar texto plano: copiar, cortar, pegar, seleccionar texto, entre

otras. Debajo de esta última, se encuentra el área de resultados, la cual está dividida en dos paneles: el primero está destinado a mostrar la solución de los modelos lineales planteados por el usuario y en el segundo se listan las operaciones realizadas por el software.

- En la zona inferior de la aplicación se muestra la barra de estado, la cual ofrece información sobre las ventanas creadas y el estado del documento de trabajo activo.

Figura I10. Interfaz general del software



Barra de menú

Agrupa las diferentes opciones, procedimientos y aplicaciones que se pueden ejecutar desde el software. Está dividido en siete menús desplegables: Archivo, Editar, Ver, Tabla, Ejecutar, Configuración y Ayuda. Todos incorporan un carácter

mnemónico. Además, los componentes más utilizados poseen combinaciones de teclas de acceso rápido. A continuación se explican en detalle cada uno de los componentes de la barra de menú.

Menú Archivo

Contiene todas las opciones para manipular los documentos creados por el usuario. En el Cuadro I4, se expone cada elemento de este menú.

CUADRO I4. Comandos del menú Archivo.

Imagen	Comando	Descripción
	Nuevo	Crea una nueva ventana para plantear un modelo (el usuario elige posteriormente entre: modelo en forma algebraica o modelo en forma tabular).
	Abrir	Al seleccionar esta opción, aparece una ventana de exploración para ubicar un archivo en el directorio, abrirlo y trabajar con él.
	Cerrar	Sirve para cerrar la ventana de un documento.
	Guardar	Permite guardar los cambios realizados en un documento que haya sido guardado previamente.
	Guardar como	Al hacer clic en esta opción, se muestra un cuadro de dialogo para guardar un documento. El usuario ingresa el nombre del documento por teclado.
	Exportar a .pdf	Guarda un documento en formato *.pdf
	Imprimir	Se utiliza para imprimir un documento.
	Salir	Cierra la ventana principal de la aplicación.

Menú Editar

Dentro de este menú se encuentran todos los comandos referentes a la edición de los modelos planteados por el usuario, así como también las opciones para buscar y reemplazar texto.

Cabe destacar, que cada una de las ventanas de trabajo creadas por el usuario poseen un mecanismo para llevar el control de todas sus operaciones, así por ejemplo, se podrá seleccionar y copiar texto de una ventana (o de cualquier otro documento abierto) y reproducirlo en otro lugar. En el Cuadro I5, se expone cada uno de los comandos que componen el menú Editar.

CUADRO I5. Comandos del menú Editar.

Imagen	Comando	Descripción
	Deshacer	Deshace la última operación (sólo disponible para los modelos escritos en forma algebraica).
	Rehacer	Rehace la última operación deshecha con el comando deshacer (sólo disponible para los modelos escritos en forma algebraica).
	Cortar	Acción que toma parte o todo el texto seleccionado de una ventana, eliminándolo a la vista, para ser reproducido posteriormente.
	Copiar	Acción que toma parte o todo el texto seleccionado de una ventana, para ser reproducido posteriormente.
	Pegar	Reproduce los datos previamente cortados o copiados, desde cualquier aplicación.

CUADRO I5. Contnuación.

Imagen	Comando	Descripción
	Seleccionar todo	Selecciona todo el texto de la ventana activa.
	Borrar todo	Borra todo el texto de la ventana activa.
	Buscar y reemplazar	Permite buscar una palabra dentro de un documento. De forma similar, se comporta la función reemplazar, salvo que se debe especificar qué la palabra va a reemplazar la palabra buscada.

Menú Ver

Contiene los comandos para manipular la visibilidad y/o la apariencia de varios elementos de la interfaz gráfica de la aplicación. En el Cuadro I6, se exponen cada uno de los comandos que agrupa este menú.

CUADRO I6. Comandos del menú Editar.

Imagen	Comando	Descripción
	Zoom	Incorpora opciones para aumentar o disminuir el tamaño de la fuente de la ventana activa.
	Panel	Permite mostrar u ocultar cada uno de los paneles que conforman la aplicación.
	Distribuir ventanas	Muestra todas las ventanas creadas por el usuario; ya sea en mosaico vertical, horizontal o en cascada.

Menú Tabla

Como se puede observar en el Cuadro I7, este menú agrupa todas las funcionalidades disponibles para plantear modelos en forma tabular. Además, posee opciones para manipular la dimensión de las tablas creadas por el usuario.

CUADRO I7. Comandos del menú Tabla.

Imagen	Comando	Descripción
Sin imagen	Insertar	Dentro de este submenú se encuentran las opciones de insertar una nueva fila o una nueva columna.
Sin imagen	Eliminar	Ofrece las opciones para eliminar una fila o una columna, previamente seleccionada por el usuario.
	Próxima tabla	Permite iniciar la próxima iteración del Algoritmo Simplex. Corresponde exclusivamente al módulo de ejercitación y práctica.
	Limpiar celdas	Borra el contenido de las celdas de la tabla activa, restaurándolas a su valor por defecto.
	Tips	Al seleccionar esta opción, la aplicación muestra un diálogo donde se señala una serie de consejos para facilitar la tarea de plantear modelos en forma tabular.

Menú Ejecutar

Este menú está conformado por los elementos que se describen en el Cuadro I8.

CUADRO I8. Comandos del menú Ejecutar.

Imagen	Comando	Descripción
	Compilar modelo	Comprueba si la gramática del modelo planteado se ajusta a la definida por la aplicación.
	Resolver	Al seleccionar esta opción, se muestra un diálogo con las opciones para hallar la solución de los modelos planteados a través de la aplicación.

Menú Configuración

Agrupa los comandos para configurar ciertas características de la aplicación. En el Cuadro I9, se presenta una descripción de los elementos que conforman este menú.

CUADRO I9. Comandos del menú Configuración.

Imagen	Comando	Descripción
	Estilo de fuente	Despliega una ventana emergente para seleccionar el tipo de fuente.
	Color de fuente	Permite seleccionar el color de la fuente.
	Color de fondo	Permite seleccionar el color de fondo de la ventana.
	Color de texto seleccionado	Muestra una paleta de colores para elegir el color del texto seleccionado.
Sin imagen	Cifras significativas	A través de esta opción, el usuario podrá cambiar el número de cifras significativas que se mostrarán en la solución de los modelos.

Menú Ayuda

Proporciona ayuda documental acerca de las operaciones que el usuario puede realizar con la aplicación. Sus comandos se muestran en el Cuadro I10.

CUADRO I10. Comandos del menú Ayuda.

Imagen	Comando	Descripción
	Ayuda de JSimPLex	Esta opción permite acceder al módulo de bases teóricas y al manual de usuario de la aplicación.
	Acerca de Java	Al seleccionar esta opción, se despliega información sobre el lenguaje de programación utilizado para desarrollar la aplicación.
Sin imagen	Acerca de JSimPLex	Muestra información sobre las características del proyecto.

Barra de herramientas

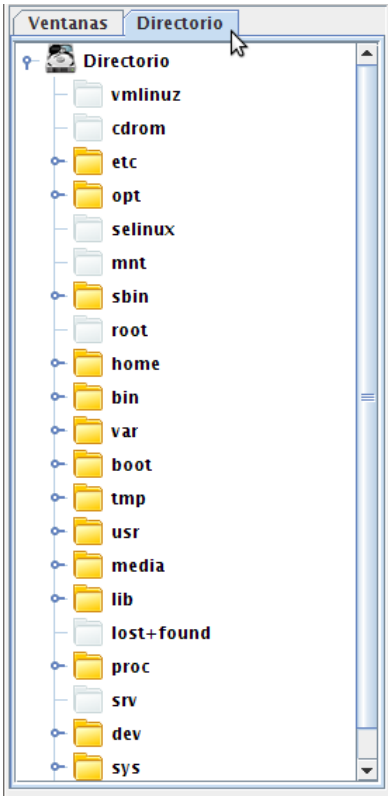
Cada botón de la barra de herramientas corresponde a algún comando de la barra de menú. No todos los comandos tienen un botón en la barra de herramientas, sólo aquellos usados con mayor frecuencia por el usuario. Es importante mencionar que dentro de la barra de herramientas se incorporó una barra de desplazamiento para ajustar, de forma rápida, el tamaño de la fuente de la ventana activa. Este mecanismo realiza un seguimiento continuo para cambiar su estado, dependiendo de la ventana donde esté trabajando el usuario. En la Figura I11, se puede apreciar la barra de herramientas de la aplicación.

Figura I11. Barra de herramientas

Explorador del directorio

Sección del entorno que permite buscar a través de un árbol de carpetas del sistema operativo, los distintos documentos creados por el usuario de la aplicación. En la Figura I12, se muestra este mecanismo.

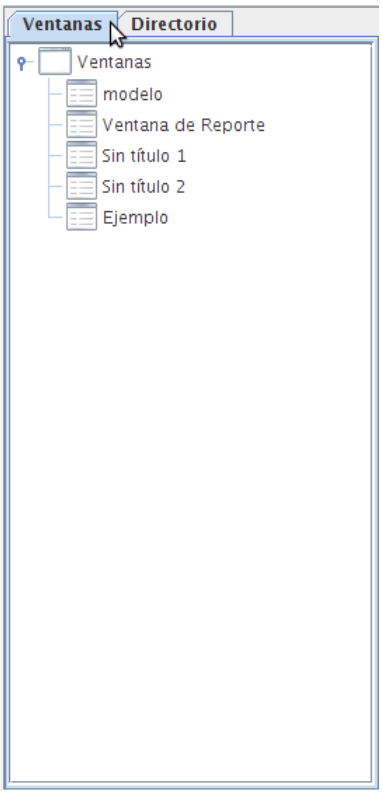
Figura I12. Explorador del directorio



Manejador de ventanas

Esta funcionalidad fue desarrollada con la objetivo de permitirle, al usuario de la aplicación, acceder a cada una de las ventanas de trabajo abiertas. Además, cuenta con un menú contextual con opciones para guardar y renombrar los documentos creados. En la Figura I13, se muestra el manejador de ventanas con que cuenta la aplicación.

Figura I13. Manejador de ventanas

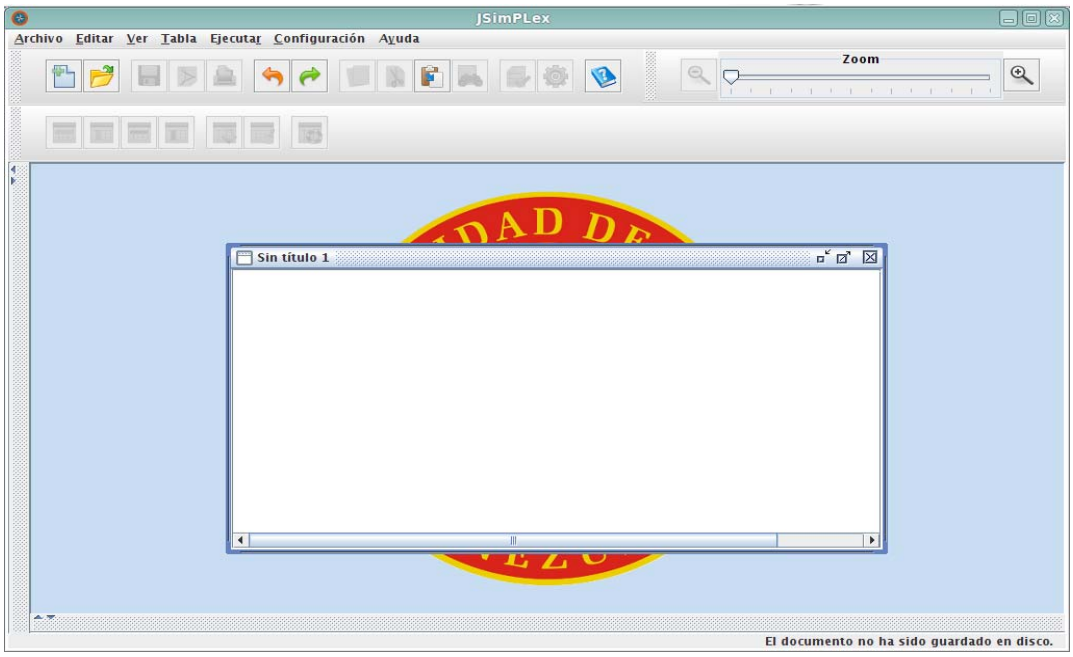


Plantear modelo en forma algebraica.

Al iniciar la aplicación, se mostrará una ventana que contiene un área de texto vacía. Esta área será utilizada para plantear un modelo lineal en forma algebraica. Si el

usuario desea crear una nueva ventana, puede seleccionar el comando Archivo → Nuevo → Plantear modelo en forma algebraica, o dar un clic el botón  ubicado en la barra de herramientas y posteriormente seleccionar la opción "Modelo en forma algebraica", (también puede utilizar Ctrl + L). En la Figura I14, se muestra la nueva área de trabajo creada.

Figura I14. Área de trabajo para plantear modelos en forma algebraica



Ahora se podrá plantear el modelo dentro de la ventana etiquetada con el nombre de <<Sin título 1>>, pero antes es importante conocer la gramática aceptada por el software para plantear modelos lineales. A continuación, se presenta un ejemplo sencillo para tal fin.

En principio se debe escribir el tipo de problema, para este ejemplo el modelo será del tipo maximización. Por lo tanto, escribiremos la palabra "Maximizar" dejamos un

espacio e ingresamos nuestra función objetivo seguido de un punto y coma (carácter de final de sentencia), quedando todo de la siguiente manera:

Maximizar 40x + 60y;

Aquí, *x* e *y* serán las variables de decisión del modelo. Estas variables representan el nivel de alguna actividad y pueden tomar cualquier valor, incluyendo valores no enteros que satisfagan las restricciones funcionales y de no negatividad. Mientras que 40 y 60representan el incremento del valor objetivo que resulta al aumentar una unidad en el nivel de la actividad asociada. También se pueden escribir fracciones y decimales en lugar de valores enteros.

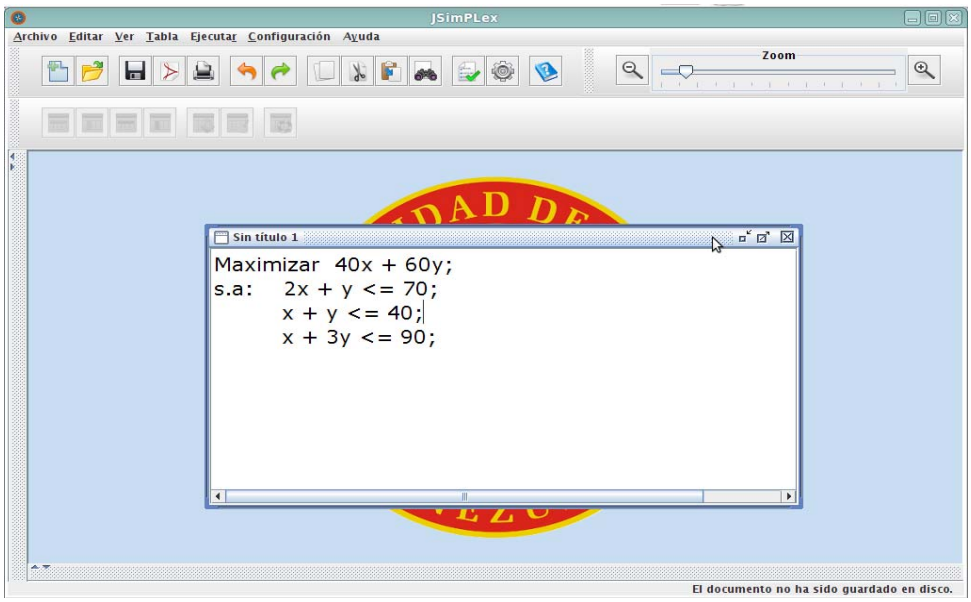
Posteriormente, se deben ingresar las restricciones del problema, pero antes, se le debe indicar al mecanismo de validación de la sintaxis, que éstas van a ser escritas. Simplemente, se colocan las siglas "s.a:", que significan "sujeto a", dejamos un espacio e introducimos el conjunto de restricciones. Éstas representan las condiciones a satisfacer y deben expresarse en forma de inecuaciones lineales. Su estructura es semejante a la de la función objetivo, sólo que al final se debe señalar su tipo, que puede ser: menor o igual (≤), mayor o igual (≥) o igual (=). Al final, se ingresa el valor del vector de recursos (también llamado vector **b**). Para este ejemplo el conjunto de restricciones serán:

2x + y<= 70;
x + y<= 40;
x + 3y<= 90;

Si se desea, se pueden escribir comentarios al final de cada sentencia y facilitar así la compresión del modelo por parte de terceros. Para ello, simplemente se colocan dos barras inclinadas (//) seguido del comentario.

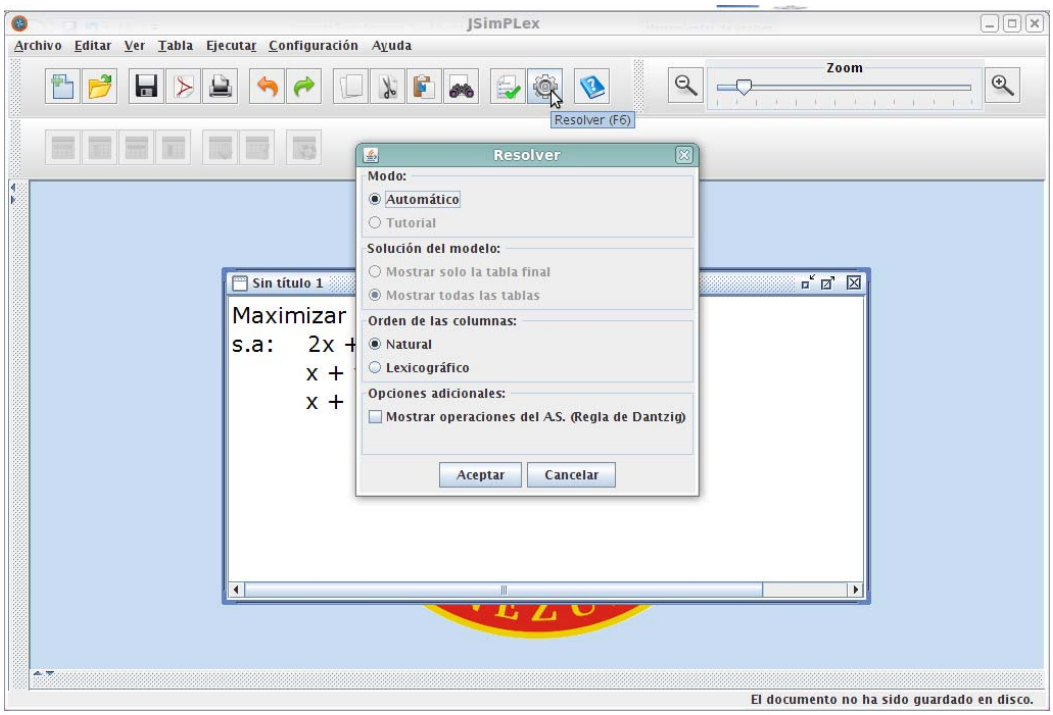
Después de introducir todos los dados necesarios, el modelo lineal debe verse como se muestra en la Figura I15.

Figura I15. Ejemplo de un modelo planteado en forma algebraica



El modelo ya ha sido planteado y está listo para ser resuelto. A continuación, seleccione el comando “Resolver”, desde la barra de menú o presione el botón  ubicado en la barra de herramientas (también se dispara el evento al presionar la tecla F4). Una ventana emergente aparecerá y se mostrarán las opciones para resolver el modelo. En la Figura I16 es posible apreciar dicha ventana.

Figura I16. Ventana de opciones



En el Cuadro I11, se expone cada una de las opciones con que cuenta la aplicación para resolver modelos lineales.

CUADRO I11. Opciones disponibles para resolver modelos lineales.

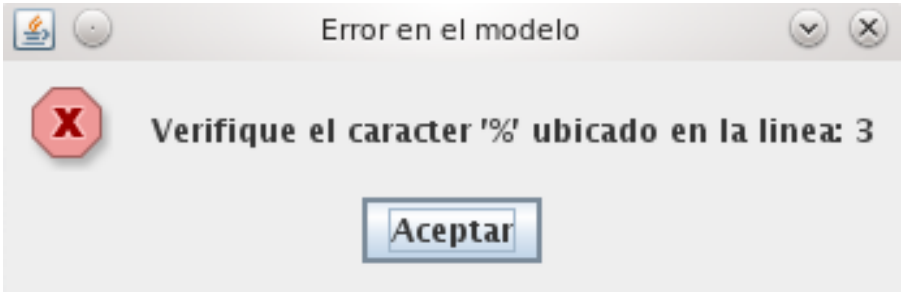
Opción	**Descripción**
Automático	El software resuelve el problema en forma automática.
Tutorial	Permite que el usuario seleccione las operaciones necesarias para conseguir la solución del modelo (disponible únicamente para modelos escrito en forma tabular).

CUADRO I11. Continuación.

Opción	Descripción
Mostrar sólo la tabla final	Como su nombre lo indica, muestra sólo la última tabla de la solución del modelo (disponible únicamente para modelos escrito en forma tabular).
Mostar todas las tablas	Muestra todas las tablas de la solución del modelo (disponible sólo para modelos escrito en forma tabular).
Natural	Muestra todas las tablas en orden natural.
Lexicográfico	Muestra todas las tablas en orden lexicográfico
Mostrar operaciones del A.S	Si está seleccionada, se mostrarán las operaciones realizadas por el software para hallar la solución de un modelo lineal.

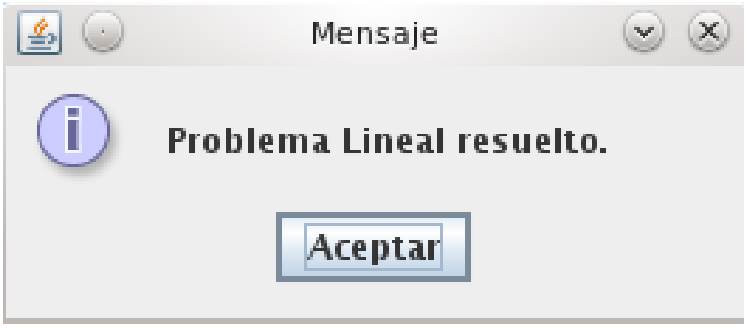
Después de hacer la selección, el software realizará las operaciones necesarias para determinar si el modelo cumple con la sintaxis exigida por la aplicación. Si es así, entonces se verifica cada uno de los elementos del modelo. Si algún elemento no pertenezca a la gramática, aparecerá un mensaje indicando la línea donde se encuentra el error y el posible elemento no válido. Deberá entonces, el usuario, examinar y corregir el problema. En La Figura I17, se muestra un ejemplo de este tipo de mensaje.

Figura I17. Mensaje de carácter no válido



Si no se presentaron errores durante el proceso de validación de la gramática, se mostrará un mensaje indicando que el modelo lineal ha sido resuelto. Un ejemplo de este tipo de mensaje puede observarse en la Figura I18.

Figura I18. Mensaje indicando que el problema lineal ha sido resuelto



Ahora, únicamente resta chequear los datos de la solución del modelo. Los resultados se ubican en el panel inferior de la aplicación, específicamente en la pestaña "Salida". Aquí se podrán observar: los datos originales del problema., el tiempo en segundos que se requirió para conseguir la solución, el valor objetivo, el vector X_B, y si el problema presentó soluciones óptimas alternativas y/o tablas degeneradas. A manera de ejemplo, en la Figura I19, se puede observar la solución al modelo lineal planteado inicialmente.

Plantear modelo en forma tabular

Si el usuario desea plantear y resolver modelos en forma tabular, debe seleccionar el comando Archivo → Nuevo →Modelo en forma tabular (también puede utilizar Ctrl + T). Se desplegará una ventana para ingresar los datos del modelo: nombre del modelo, número de variables y número de restricciones. En la Figura I20 se muestra un ejemplo de esta ventana.

Figura I19. Ejemplo área de resultados (Panel Solución)

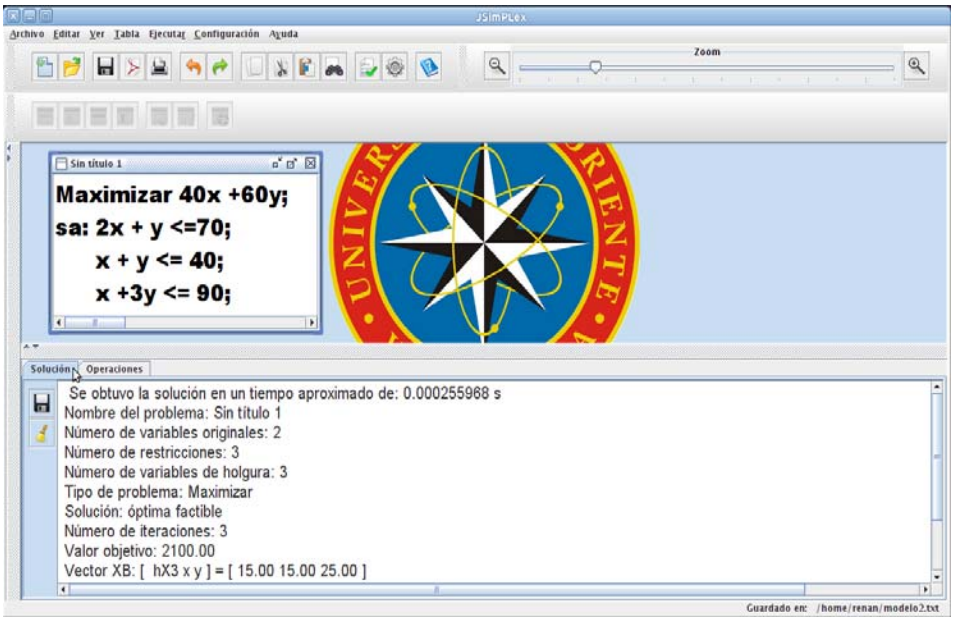
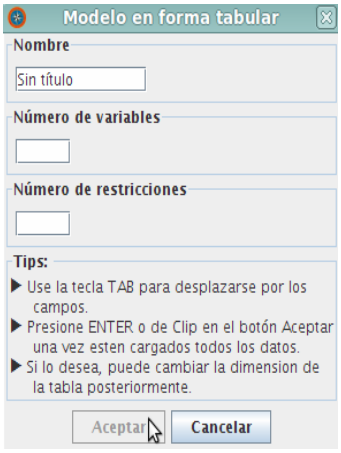


Figura I20. Ventana para ingresar los datos del modelo



Una vez ingresado todos los datos, se mostrará la ventana para plantear modelos a través de tablas simplex. Para ingresar los valores del modelo, se debe dar un clic sobre las celdas de la tabla. Inmediatamente la celda seleccionada pasará a modo edición y se

podrán introducir datos, ya sean valores enteros o decimales. Para indicar el tipo de problema y el tipo de restricción, se debe posicionar y hacer clic sobre la celda correspondiente (o seleccionarla y presionar la tecla F2). El valor de dicha celda cambiará automáticamente. En la Figura I21, se muestra un modelo lineal planteado en forma tabular.

Cabe destacar que la aplicación fue desarrollada exclusivamente para resolver problemas lineales de caso fácil, es decir, donde todas las restricciones son del tipo menor o igual (≤). Si el usuario ingresa alguna restricción del tipo mayor o igual (≥), el software está en la capacidad de intentar transformarla, si esto no es posible (el componente del lado derecho queda negativo), se desplegará un mensaje informando sobre la situación. Un ejemplo de este tipo de mensaje se muestra en la Figura I22.

Figura I21. Ejemplo modelo planteado en forma tabular

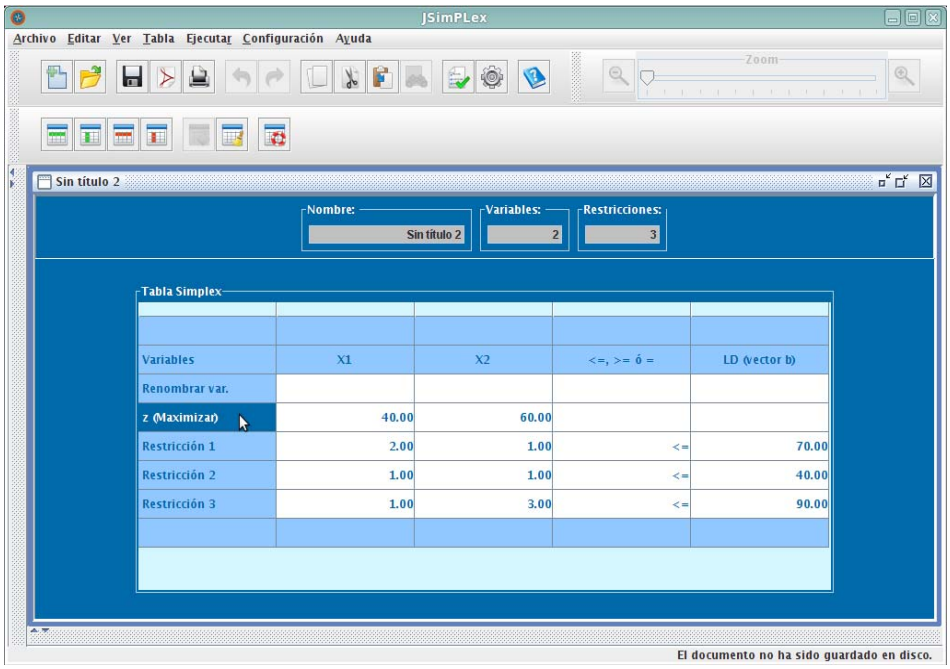
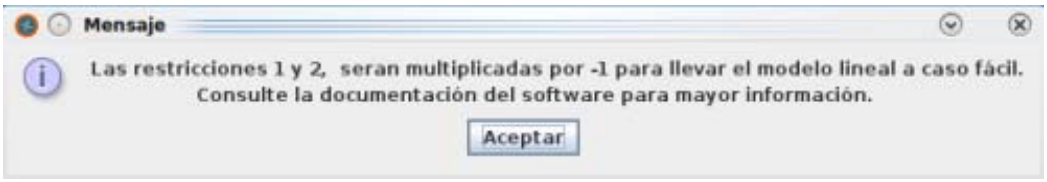
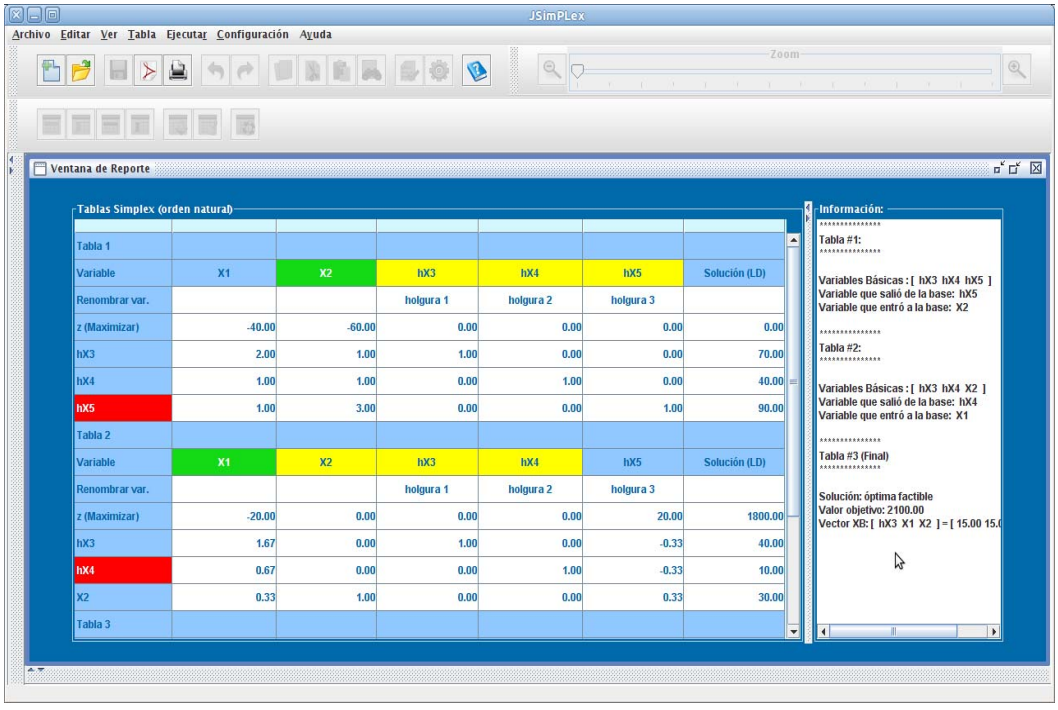


Figura I22. Mensaje de conversión de restricciones



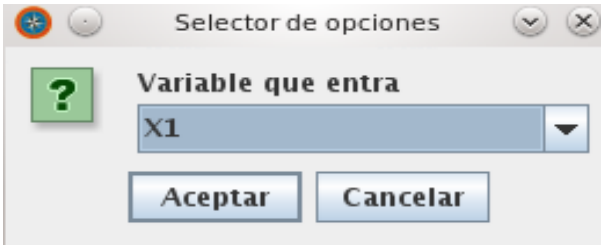
El modelo ya ha sido planteado y está listo para ser resuelto. Ahora, se debe seleccionar el comando “Resolver” desde la barra de menú o presione el botón  ubicado en la barra de herramientas (también se dispara el evento al presionar la tecla F4). Una ventana emergente aparecerá y se mostrarán las opciones con que cuenta la aplicación (para mayor información consulte el Cuadro I14 de este manual: ventana de opciones). El modelo lineal resuelto se muestra en la Figura I23.

Figura I23. Ejemplo ventana de resultados



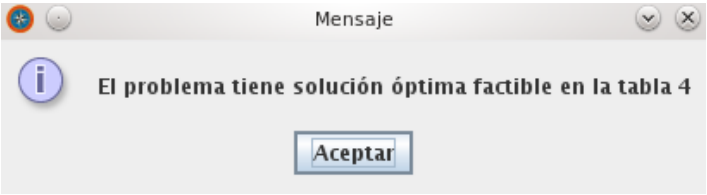
Una vez planteado un modelo en forma tabular, se puede seleccionar la opción para resolverlo de manera guiada (modo tutorial), la cual consiste en resolver paso a paso el problema, señalando las operaciones a realizar y dejando que la aplicación realice los cálculos. Para hacer esto, el usuario debe seleccionar la opción "Modo tutorial" y dar un clic en el botón Aceptar. Inmediatamente, aparecerá la primera tabla correspondiente al modelo lineal estandarizado. Para seleccionar la variable de entrada y la variable de salida, debe ubicar el comando "Próxima iteración", desde el menú Tablas, o presionar el botón  ubicado en la barra de herramientas. De esta manera, se mostrará una ventana emergente para hacer la selección. En la Figura I24 se puede observar un ejemplo de esta ventana.

Figura I24. Ventana de selección de variables



Cabe destacar que las opciones mostradas para la selección de variables se actualiza automáticamente después de cada iteración, así, se logra minimizar los posibles errores en lo que respecta a la entrada de datos. Una vez que el software ha alcanzado la solución del modelo, aparecerá una ventana emergente notificándolo al usuario, como se muestra en la Figura I25.

Figura I25. Mensaje indicando la solución del modelo.



La sintaxis del modelo

En esta sección, se detallará la sintaxis a seguir para plantear modelos lineales utilizando la aplicación.

Todo modelo lineal debe empezar con el tipo de problema, es decir, se debe señalar si el problema es del tipo maximización o minimización. Para hacer esto, podrá utilizar las palabras Max, Maximizar o Maximice, en caso de maximización o Min, Minimizar o Minimice, en caso de minimización. Es importante señalar, que el analizador de la sintaxis de la aplicación no es sensible al uso de mayúsculas o minúsculas.

Al momento de escribir la función objetivo y las restricciones del modelo, se reconocen tres tipos de operadores: el operador mas (+), el operador menos (-) y el símbolo de menor o igual (≤). Además, el software no soporta el uso de paréntesis como indicadores del orden de precedencia. Todas las operaciones son leídas de izquierda a derecha.

Al terminar de escribir la función objetivo, se debe indicar que empezará a escribir el conjunto de restricciones. Para hacer esto, se escribe la instrucción “s.a:” que significa “sujeto a:”. También puede utilizar “sa:”, “st:” o “s.t:”. Igual que en el caso anterior, el analizador de la sintaxis no es sensible al uso de mayúsculas o minúsculas.

En lo que respecta a la forma de escribir las variables, las mismas deben empezar con una letra [A-Z] [a-z], seguido de uno o varios caracteres del mismo tipo y puede o no finalizar con un número entero. En este caso, el analizador de la sintaxis es sensible al uso de mayúsculas y minúsculas, es decir, por ejemplo, la aplicación considera diferentes las variables a1 y A1. Se recomienda el uso de nombres de variables significativos que faciliten el entendimiento del modelo por parte de terceros. A manera de ejemplo, los siguientes nombres de variables con aceptadas por el analizador de la gramática: A, X1, casa.

Los comentarios pueden ser escritos al finalizar alguna de las sentencias que conforman el modelo. Un comentario comienza con dos barra inclinadas (//), seguido de uno o varios caracteres de cualquier tipo, finalizando con un salto de línea, por lo tanto, todo lo que se escriba en la misma línea después de las dos barra (//) será ignorado por el analizador de la sintaxis.

En cuanto a los coeficientes de la función objetivo, de las restricciones y los componentes del vector de recursos, éstos pueden ser representados a través de valores enteros, decimales o fracciones. Para este último caso es necesario que sean escritos entre paréntesis. El software posee los mecanismos necesarios para transformar una fracción a un número decimal de hasta nueve (9) cifras significativas Un ejemplo de una fracción aceptada por el analizador de la gramática sería (3/2).

Al finalizar cada instrucción es necesario ingresar el carácter punto y coma (;), para así indicarle al analizador de la gramática que ha concluido una sentencia y se dará inicio a la siguiente. No es necesario indicar que se ha terminado de formular el modelo lineal con alguna palabra reservada, la aplicación es capaz de identificar la última restricción del modelo y entonces asumir que el planteamiento del mismo ha llegado a su fin.

También es importante señalar, que el software considera que todas las variables a utilizar son estrictamente mayores o iguales a cero (≥ 0). Por lo tanto, no hace falta señalarlo de alguna manera.

Teclas de acceso rápido

Es posible acceder a ciertas funcionalidades del software a través del uso del *mouse* y/o combinaciones de teclas de acceso rápido. De esta manera, el usuario de la aplicación podrá realizar su trabajo de forma simple y eficiente. En el Cuadro I12, se muestra cada uno de los comandos con que cuenta la aplicación.

CUADRO I12. Combinaciones de teclas de acceso rápido.

Comando	**Acción**
Ctrl + A	Selecciona todo el texto del área de trabajo activa.
Ctrl + O	Abrir un documento.
Ctrl + G	Guardar un documento.
Ctrl + L	Crear una ventana para plantear modelo en forma algebraica.
Ctrl + T	Crear una ventana para plantear modelo en forma tabular.
Ctrl + I	Imprimir un documento.
Ctrl + S	Cerrar la aplicación.
Ctrl + Z	Deshace la última acción.
Ctrl + Y	Rehace la última acción.
Ctrl + X	Cortar texto seleccionado.
Ctrl + C	Copiar texto seleccionado.

CUADRO I12. Continuación.

Comando	Acción
Ctrl + V	Pegar texto.
Ctrl + B	Buscar y reemplazar texto.
F1	Acceder a la ayuda del software.
F3	Validar sintaxis del modelo.
F4	Resolver modelo.
Alt + A	Despliega el menú Archivo.
Alt + E	Despliega el menú Editar.
Alt + V	Despliega el menú Ver.
Alt + T	Despliega el menú Tabla.
Alt + R	Despliega el menú Resolver.
Alt + C	Despliega el menú Configuración.
Alt + Y	Despliega el menú Ayuda.

ANEXOS

Anexo 1. Valores en XP.

Comunicación

Uno de los valores más importantes en XP es la comunicación. La mala comunicación no surge por casualidad en un proyecto y pueden aparecer muchas circunstancias que hagan que ésta falle; un programador le da malas noticias al gerente y éste lo castiga, un cliente le dice al programador algo importante y éste no le presta atención. En cualquiera de los casos la comunicación es un factor importante en cualquier tipo de proyecto. En XP se trata de mantener una buena comunicación mediante un conjunto de prácticas que no se pueden realizar sin tener una buena comunicación en el equipo.

Muchas de estas prácticas hacen corto circuito si no hay buena comunicación como en el caso de las pruebas unitarias, programación por pares, el uso de estándares o la estimación de las tareas. Trabajar en espacios abiertos hace que la comunicación mejore al contrario de otras metodologías en las cuales los programadores trabajan en espacios reducidos.

La comunicación con el cliente es de vital importancia en XP y es por este motivo que el cliente es integrado al equipo. De esta forma, cualquier duda sobre los requerimientos puede ser evacuada inmediatamente. Además, se planifica con el cliente y este puede estar al tanto del avance del proyecto.

XP ha sido diseñada para minimizar el grado de documentación como forma de comunicación, haciendo énfasis en la interacción personal. De esta manera se puede

avanzar rápidamente y de forma efectiva, realizando solo la documentación necesaria.

Simplicidad

La simplicidad es el segundo valor que se utiliza en esta metodología. XP apuesta a realizar algo simple hoy y destinar un poco más de esfuerzo para realizar un cambio en el futuro, a realizar algo más complicado hoy y no utilizarlo nunca. XP propone una regla muy simple: "hacer algo que funcione de la manera más sencilla". En el caso de tener que añadir nueva funcionalidad al sistema se deben examinar todas las posibles alternativas y seleccionar la más sencilla. En otras ocasiones se hace uso de la refactorización de código que permite mantener el código en funcionamiento pero mucho más simple y organizado.

Otra regla muy importante es: "realizar sólo lo necesario". Con esto se pretende agregar nueva funcionalidad que cumpla con los objetivos actuales sin necesidad de preocuparse.

Retroalimentación temprana

Brindar un *feedback* correcto y preciso hace que se pueda mantener una buena comunicación y conocer el estado actual del proyecto. El *feedback* trabaja a diferentes escalas de tiempo. Uno es el *feedback* que se realiza minuto a minuto. Cuando un cliente escribe sus HU los programadores realizan la estimación de cada una de ellas y el cliente puede obtener inmediatamente el *feedback* sobre la calidad de dichas historias.

El otro tipo de *feedback* que se realiza es a través de pequeñas entregas del sistema. De esta manera, el cliente está al tanto del avance del proyecto. Además, el sistema es puesto en producción en menor tiempo, con lo cual los programadores saben si realizaron un buen trabajo y si sus decisiones fueron acertadas.

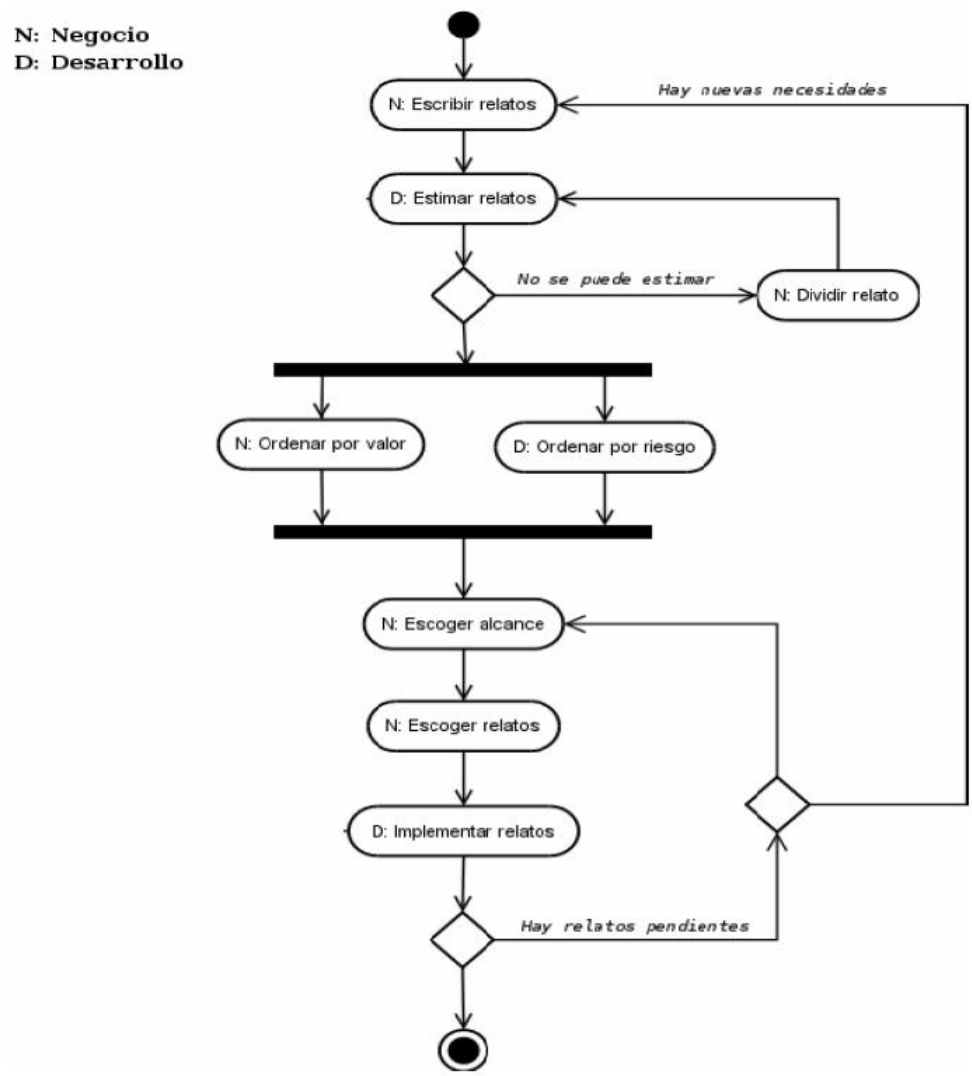
Coraje

Uno de los lemas de XP menciona: “Si no trabajas al tope de tu capacidad, alguien más lo está haciendo y si no llegas a tiempo se comerá tu almuerzo”. Esto hace, a que por ejemplo, se tenga el coraje de modificar el código en cualquier momento por cualquier miembro del equipo sabiendo que no se afectará el correcto funcionamiento del sistema. El coraje también es poder realizar cambios cuando algo no funciona del todo bien, diseñar e implementar sólo lo necesario para el presente, pedir ayuda o reducir el alcance de una entrega si el tiempo no alcanza. Si no hay coraje en un proyecto no se puede clasificar como extremo y es necesario que los otros tres valores estén presentes.

Obviamente cada uno de los valores antes mencionados tiene una gran interacción entre ellos. La comunicación, la simplicidad y el *feedback* forman el coraje, el cual se convierte en el objetivo principal de XP.

ANEXO 2

DIAGRAMA DE ACTIVIDADES DEL JUEGO DE PLANIFICACIÓN



HOJA DE METADATOS

Hoja de Metadatos para Tesis y Trabajos de Ascenso – 1/6

Título	SOFTWARE PARA LA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL APLICANDO EL ALGORITMO SIMPLEX (CASO FÁCIL)
Subtítulo	

Autor(es)

Apellidos y Nombres	Código CVLAC / e-mail	
Salazar Salazar, Renan Alberto	**CVLAC**	17957100
	e-mail	renansalazar87@gmail.com
	e-mail	
	CVLAC	
	e-mail	
	e-mail	
	CVLAC	
	e-mail	
	e-mail	
	CVLAC	
	e-mail	
	e-mail	

Palabras o frases claves:

Investigación de Operaciones
Programación Lineal
Método Simplex
Software
Java

Hoja de Metadatos para Tesis y Trabajos de Ascenso – 2/6

Líneas y sublíneas de investigación:

Área	Subárea
Ciencias	Informática

Resumen (abstract):

Se desarrolló un software, denominado JSimplex, para resolver problemas de Programación Lineal (PL) utilizando el Algoritmo Simplex (AS), para el caso fácil. El mismo se construyó siguiendo las fases y principios de la metodología Programación Extrema, bajo un enfoque ágil e iterativo-incremental. En la fase de exploración, se procedió a recabar toda la información necesaria a través de entrevistas, cuestionarios, redacción de las tarjetas Historias de Usuario (HU), revisión bibliográfica y la observación directa, obteniendo así los requisitos funcionales y las necesidades de los usuarios. Adicionalmente, se creó un plan de liberaciones, el cual permitió definir el total de iteraciones realizadas para cada entrega de código. En la fase de gestión, se estableció un ambiente de trabajo acorde a las necesidades y se seleccionó la arquitectura, tecnologías y herramientas a utilizar a lo largo de todo el proceso de desarrollo. La fase de construcción se dividió en cuatro sub-fases: planificación, diseño, codificación y pruebas. En principio, se seleccionaron las tarjetas HU a implementar dentro de cada iteración, para posteriormente llevar a cabo el proceso de modelado de la aplicación y la creación de prototipos. La codificación fue realizada utilizando herramientas de software libre, entre las cuales destacan: *Ubuntu* 10.04 como sistema operativo GNU/Linux, *Java* como lenguaje para la programación de los módulos de la aplicación y *Netbeans* 6.8 como entorno de desarrollo. Al final de cada liberación de código, se llevaron a cabo las pruebas de usuario y de aceptación, las cuales permitieron evaluar si el requerimiento solicitado había sido cumplido. Para la fase de culminación, se procedió a unir cada uno de los módulos desarrollados en las etapas anteriores y además se elaboró la documentación de la aplicación. El software desarrollado permitirá resolver problemas de PL utilizando el AS, caso fácil, ya sea mediante la escritura de modelos en forma algebraica o a través del uso de tablas simplex. Se empleó la descomposición LU (del inglés *Lower-Upper)* para hallar la inversa de la matriz **B**, para proporcionarle estabilidad numérica a la aplicación y se utilizó la regla de Bland para prevenir el fenómeno de ciclaje. Además, el software cuenta con un módulo interactivo-intruccional orientado a reforzar los conocimientos adquiridos en las aulas de clase, especialmente dirigidos a los estudiantes que se inician en el área de la Investigación de Operaciones. También ofrece las posibilidades de observar las operaciones realizadas por el algoritmo y de consultar definiciones y ejemplos sobre temas relacionados con la PL.

Hoja de Metadatos para Tesis y Trabajos de Ascenso – 3/6

Contribuidores:

Apellidos y Nombres	ROL / Código CVLAC / e-mail	
Lockiby, José	**ROL**	**CA** [ ] **AS** [X] **TU** [ ] **JU** [ ]
	CVLAC	**10.302.899**
	e-mail	
	e-mail	
Centeno, Manuel	**ROL**	**CA** [ ] **AS** [ ] **TU** [ ] **JU** [x]
	CVLAC	**4.431.407**
	e-mail	
	e-mail	
Marcano, Hugo	**ROL**	**CA** [ ] **AS** [ ] **TU** [ ] **JU** [x]
	CVLAC	**6.766.726**
	e-mail	
	e-mail	
	ROL	**CA** [ ] **AS** [ ] **TU** [ ] **JU** [ ]
	CVLAC	
	e-mail	
	e-mail	

Fecha de discusión y aprobación:

Año	Mes	Día
2011	12	01

Lenguaje: SPA

Hoja de Metadatos para Tesis y Trabajos de Ascenso – 4/6

Archivo(s):

Nombre de archivo	Tipo MIME
TESIS-SalazarR.DOCX	Aplication/word

Alcance:

Espacial: _____ (Opcional)

Temporal: _____ (Opcional)

Título o Grado asociado con el trabajo: Licenciado en Informática

Nivel Asociado con el Trabajo: Licenciado

Área de Estudio: Informática

Institución(es) que garantiza(n) el Título o grado:

UNIVERSIDAD DE ORIENTE

Hoja de Metadatos para Tesis y Trabajos de Ascenso – 5/6

UNIVERSIDAD DE ORIENTE
CONSEJO UNIVERSITARIO
RECTORADO

CU Nº 0975

Cumaná, 04 AGO 2009

Ciudadano
Prof. JESÚS MARTÍNEZ YÉPEZ
Vicerrector Académico
Universidad de Oriente
Su Despacho

Estimado Profesor Martínez:

Cumplo en notificarle que el Consejo Universitario, en Reunión Ordinaria celebrada en Centro de Convenciones de Cantaura, los días 28 y 29 de julio de 2009, conoció el punto de agenda ***"SOLICITUD DE AUTORIZACIÓN PARA PUBLICAR TODA LA PRODUCCIÓN INTELECTUAL DE LA UNIVERSIDAD DE ORIENTE EN EL REPOSITORIO INSTITUCIONAL DE LA UDO, SEGÚN VRAC Nº 696/2009".***

Leído el oficio SIBI – 139/2009 de fecha 09-07-2009, suscrita por el Dr. Abul K. Bashirullah, Director de Bibliotecas, este Cuerpo Colegiado decidió, por unanimidad, autorizar la publicación de toda la producción intelectual de la Universidad de Oriente en el Repositorio en cuestión.

Comunicación que hago a usted a los fines consiguientes.

UNIVERSIDAD DE ORIENTE
SISTEMA DE BIBLIOTECA
RECIBIDO POR
FECHA 5/8/09 HORA 5:30

Cordialmente,

JUAN A. BOLAÑOS CURVELO
Secretario

REPÚBLICA BOLIVARIANA DE VENEZUELA · UNIVERSIDAD DE ORIENTE · SECRETARÍA · CONSEJO UNIVERSITARIO

C.C: Rectora, Vicerrectora Administrativa, Decanos de los Núcleos, Coordinador General de Administración, Director de Personal, Dirección de Finanzas, Dirección de Presupuesto, Contraloría Interna, Consultoría Jurídica, Director de Bibliotecas, Dirección de Publicaciones, Dirección de Computación, Coordinación de Teleinformática, Coordinación General de Postgrado.

JABC/YGC/maruja

Hoja de Metadatos para Tesis y Trabajos de Ascenso- 6/6

Artículo 41 del REGLAMENTO DE TRABAJO DE PREGRADO (vigente a partir del II Semestre 2009, según comunicación CU-034-2009) : "los Trabajos de Grado son de la exclusiva propiedad de la Universidad de Oriente, y sólo podrán ser utilizados para otros fines con el consentimiento del Consejo de Núcleo respectivo, quien deberá participarlo previamente al Consejo Universitario para su autorización".

_____
Salazar, Renan
AUTOR 1

_____
Lockiby, José
TUTOR