Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta de apoyo a la enseñanza de la Investigación de Operaciones “Simple PL”.

Trabajo de Diploma para optar por el título de
Ingeniero en ciencias Informáticas

Autor:

Dayan Hernández Ramos

Tutor:

Ing. Yorgüy Antonio Batista Desdin

La Habana, junio 2016

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Ciencias Básicas de la Facultad 3 de la Universidad de las Ciencias Informáticas; así como a dicha institución para que hagan el uso que estimen pertinente con este trabajo, cediéndoles así los derechos patrimoniales de la misma, con carácter exclusivo.

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

Dayan Hernández Ramos

_____

Firma del Autor

Ing. Yorgüy Antonio Batista Desdin

_____

Firma del Tutor

Datos de Contacto

Síntesis del Tutor: Ingeniero en Ciencias Informáticas, UCI 2014. Recién graduado en adiestramiento. Se ha desempeñado como Programador en el proyecto *Sistema Integral de Aduanas (GINA)*. Ha impartido las asignaturas Probabilidades y Estadísticas e Investigación de Operaciones en la Universidad de las Ciencias Informáticas. Ha participado en eventos como la Jornada Científica UCI, COMPUMAT y el concurso La Clase Universitaria.

Nombre y apellidos: Yorgüy Antonio Batista Desdin.

Correo: yabatista@uci.cu


"Programar es una de las pocas cosas del mundo en la que te sientas y puedes crear algo totalmente nuevo de la nada"
Mark Zuckerberg

Agradecimientos

A mi mamá, mi gorda, mi novia, mi amiga, mi todo. Gracias por cumplirme mi sueño, nuestro sueño. Gracias por estar siempre ahí para mí, por no dudar ni un momento de que yo sí podía, aun cuando muchas personas opinaban lo contrario. Gracias por no conocer fronteras en cuanto a mí respecta. Gracias por tus consejos, tu amor y dedicación incondicional. Gracias por tantas noches de sacrificio, el tiempo demostró que no fueron en vano. Hoy ya los dos somos licenciados e ingenieros, porque este título también es tuyo.

A mi hermanita, mi cotorra, gracias por ser como eres. Gracias por quererme y preocuparte siempre por mí. Gracias por dejarme ser tu ejemplo a seguir. Gracias por ser mi enfermera personal cada vez que me enfermo en casa. Gracias por consentirme, por preferirme.

A mi team de lujo, Eimé, Nairovis, Yudelyn e Idelmis, gracias por siempre estar ahí para mí, en las buenas y en las malas, gracias por ser mis amigas, mis confidentes, mis enfermeras con sus remedios caseros, mis cocineras. Gracias por transmitirme su experiencia y enseñarme así a enfrentarme a las situaciones que me ha impuesto la vida. No importa la distancia que nos separe en el futuro, siempre estarán en mi pensamiento.

A mis amigos Daylín y José Luis, que desde el primer momento me apoyaron y me aceptaron tal y como soy. Gracias a los dos por abrirme las puertas de su casa y tratarme como uno más de la familia. Aunque hoy no nos graduemos todos, este título vale por los tres.

A mis hermanas en la UCI, Vanessa, Verónica, Valeria, Violetta y Gloria, gracias por todas las noches de alegría que me dieron con solo dos sábanas y un buen tema. Gracias por hacerme sentir tan bien, aun cuando estaba lejos de las dos personas que más quiero en este mundo, ustedes me hacían sentir en familia.

A mis amigos Yoslenys, Addiel, Tito, Richard, Aniel, Roisbel y Alex. Gracias por su amistad, sus locuras, por su tiempo dedicado, su ayuda y sus regaños oportunos.

A Rosalina, gracias por más que profesora, ser mi amiga y confidente, ojalá todos pudieran contar con una persona tan especial como tú entre sus amigos, me siento orgulloso de tenerte.

A Jany y su familia, por permitirme entrar en su casa y tratarme como un hijo más.

A Irma, esa amiga que a pesar de tanto tiempo sin vernos, me acogió como un hijo. Gracias por ofrecerme tu casa incondicionalmente.

A Andrés Asprón, gracias por ser mi amigo. Gracias por siempre estar, ya fuera para compartir en fiestas o para brindarme un hombro donde llorar cuando me hizo falta. Tu mamá y tu hermanita deben sentirse bien orgullosas del hombre en que te has convertido.

A Daniel, gracias por ser mi amigo, a pesar de que nuestras personalidades son tan distintas. Gracias por ayudarme siempre que me hizo falta, por estar en las buenas y en las malas. Gracias por abrirme las puertas del mundo de Warcraft.

A la brigada 3502, gracias a todos y cada uno de sus integrantes por aceptarme tal y como soy. Gracias por ser mis amigos, sin importar nada. Gracias por sus bromas pesadas, sus consejos, por los buenos momentos. Espero aun en la distancia podamos seguir en contacto, ya que amigos como ustedes no todos tienen.

A todos los profesores que de una u otra forma contribuyeron en mi paso por la universidad.

Al tribunal, muchas gracias por haberme ayudado a que hoy pueda estar aquí, con sus sugerencias y consejos durante el desarrollo de este trabajo. En especial a la profesora Yalice, que a pesar de que su asignatura no fue de mis preferidas, siempre tuvo la paciencia y el deseo de explicarme, sin importar cuantas veces me di por vencido.

A mi tutor, el ingeniero Yorgüy Antonio Batista Desdin, gracias por además de tutor, ser mi amigo. Gracias por ayudarme a cumplir mi sueño de graduarme como ingeniero en la Universidad de las Ciencias Informáticas. Gracias por regañarme siempre que hizo falta y por mostrarme el camino a seguir. Sin ti no hubiera sido posible.

Dedicatoria

A mi mamá, mi motor impulsor, mi fuerza vital. Te dedico este logro, fruto de tu sacrificio y amor profesado durante toda mi vida.

A mi hermana, lo mejor que me ha pasado en la vida.

Resumen

La resolución de problemas de Programación Lineal es uno de los principales temas que se imparte en la asignatura Investigación de Operaciones en la Universidad de las Ciencias Informáticas. Las aplicaciones informáticas con las que se trabaja en dicha asignatura, no satisfacen los requerimientos particulares de la misma. Además, estos sistemas no abarcan todos los temas impartidos y su método de visualización de los datos dista del presente en la bibliografía básica, lo que va en detrimento del Proceso de Enseñanza Aprendizaje. La no interoperabilidad de sistemas de dichas herramientas se suma a las razones que dan cauce a esta investigación.

Por tales motivos se decide desarrollar una aplicación que contribuya a la integración de los temas de la asignatura, acercándolos a la forma tabular presente en la bibliografía y utilizada por los profesores de la asignatura para impartir la misma, indistintamente del sistema operativo que utilicen. La creación de esta nueva aplicación permitirá elevar la calidad del Proceso de Enseñanza Aprendizaje, así como una mejor preparación por parte de los estudiantes.

PALABRAS CLAVE: Investigación de Operaciones, programación lineal, integración.

Índice

Introducción

En la actualidad los centros educacionales y en especial los centros pertenecientes al Ministerio de Educación Superior se enfrentan al reto de encontrar nuevas formas para garantizar un Proceso de Enseñanza Aprendizaje (PEA) de mayor calidad. Haciéndose necesario el empleo de nuevos recursos educativos que apoyen el trabajo cotidiano de educadores y educandos, resulta imprescindible la incorporación de las nuevas Tecnologías de la Información y las Comunicaciones (TIC).

En el plan de estudio de la carrera de Ingeniería en Ciencias Informáticas, de la Universidad de las Ciencias Informáticas (UCI), en el segundo semestre de tercer año, se imparte la asignatura Investigación de Operaciones (IO). Esta asignatura perteneciente a la disciplina de Matemática Aplicada, reviste gran importancia pues su principal fundamento es el apoyo a la toma de decisiones a partir de modelos matemáticos. La modelación, solución e interpretación de resultados de problemas de Programación Lineal (PL) son el eje y la guía de la asignatura. Los estudiantes de tercer año disponen como bibliografía fundamental el libro de texto, Introducción a la Investigación de Operaciones de los autores Frederick S. Hillier[1] y Gerarld J. Lieberman[2], que propone tanto en sus ejercicios como en sus ejemplos una estructura para modelar y resolver estos problemas. Los problemas de PL establecen dos partes fundamentales, la función objetivo y las restricciones. Encontrar una solución matemática que maximice o minimice la función objetivo, atendiendo a las restricciones planteadas es engorroso y difícil de hacer manualmente.

El libro de texto propone el método Símplex en su forma tabular y su forma algebraica. Este procedimiento es abordado en el primer tema de la asignatura. La resolución de estos problemas utilizando este método requiere gran cantidad de tiempo por el elevado número de cálculos a realizar. Para el apoyo de la asignatura se utilizan varias herramientas, entre ellas el software “WinQSB”. El mismo está implementado para un ambiente empresarial y no para un ambiente educativo. Este software es básicamente una herramienta de cálculo donde se introducen los elementos del modelo desarrollado y es devuelta la evaluación final de todas las variables como salida. Los problemas de la asignatura tienen variantes de

[1] Dr. Frederick S. Hillier: Destacado profesor de la Universidad de Stanford, nacido en Aberdeen, Washington, autor de otros textos como: *Introduction to Mathematical Programming* e *Introduction to Stochastic Models in Operations Research*, *The Evaluation of Risky Interrelated Investments*, *Queueing Tables and Graphs*, e *Introduction to Management Science A Modeling and Case Studies Approach with Spreadsheets*.

[2] Dr. Gerald J. Lieberman (1925-1999): Destacado profesor de la Universidad de Stanford, autor de otros textos como: *Introduction to Mathematical Programming* e *Introduction to Stochastic Models in Operations Research, Handbook of Industrial Statics, Tables of the Non-Central t-Distribution, Tables of the Hypergeometric Probability Distribution* e *Introduction to Management Science: A Modeling and Case Studies Approach with Spreadsheets*.

resolución de acuerdo con sus características y este software resuelve el problema sin mostrar qué variante utilizó. Como está enfocado al ambiente empresarial no cuenta con explicaciones, o ejemplos prototipos para que los estudiantes realicen comparaciones entre sus cálculos manuales y los efectuados por el software. El método de trabajo dista del método tabular enseñado en clases y presente en el libro de texto de la asignatura. Esto provoca que los profesores de la asignatura necesiten destinar tiempo para mostrar similitudes y diferencias entre el método manual tabular dado en clases y el método aplicado por el software. Estas diferencias crean problemas en la asimilación del conocimiento, dificultando el PEA. (Tomado del informe del colectivo de la asignatura (Miyashiro, y otros, 2015))

Como parte del contenido de la asignatura está el trabajo con problemas de redes, transporte y asignación. Estos presentan características especiales y necesitan una interfaz que le permita a los estudiantes modelarlos y a partir de este modelo obtener una representación gráfica. También a partir de la representación gráfica debe ser posible modelar el problema. Esta funcionalidad no está presente en el software, de modo que, para suplirla en los problemas de redes de flujo máximo, es sustituido por una pequeña herramienta denominada “Trayectorias de Aumento”, desarrollada por uno de los profesores del departamento.

Las herramientas mencionadas tienen las siguientes desventajas:

- **“WinQSB”:**
  - Solo funciona en la plataforma de Windows en los sistemas operativos Windows 98, 2000 y XP.
  - Posee una licencia privativa.
  - En los sistemas Linux solo es posible ejecutarlo utilizando el emulador Wine.
  - Desarrollado en Visual Basic, lenguaje cuya última versión fue la número 6, liberada en 1998, para la que Microsoft brindó soporte solamente hasta marzo de 2008.
- **Trayectorias de Aumento:**
  - Centra su trabajo en un método de solución de un tema de la asignatura solamente.
  - De los cuatro tipos de problemas de redes tratados en la asignatura, sólo da soporte a los problemas de flujo máximo.

La siguiente tabla muestra como estas herramientas cubren cada uno de los temas de la asignatura:

Tabla 1 *Relación de las herramientas de apoyo a la docencia y los temas de la asignatura.*

	Modelación	Métodos de solución	Problemas de redes	Apoyo a la toma de decisiones
"WinQSB"	No	Sí	No	No
Trayectorias de Aumento	No	No	Parcialmente	No

Como se puede observar, los temas "Modelación" y "Apoyo a la toma de decisiones", no están cubiertos por ninguna de las dos herramientas en cuestión. El "WinQSB" por su parte cubre el tema de "Métodos de solución". La herramienta "Trayectorias de Aumento" cubre parcialmente el tema "Problemas de redes", ya que de los métodos empleados para la resolución de dichos problemas solamente cubre flujo máximo, sin tener en cuenta el método de cortadura mínima, transporte y asignación. Todo esto evidencia la necesidad de crear una herramienta que contribuya al PEA de IO abarcando los cuatro temas de la asignatura.

Después de un análisis de las necesidades existentes y con el fin de solucionar la situación anterior, se plantea el siguiente **Problema a resolver:** Las herramientas utilizadas en la asignatura de Investigación de Operaciones no se ajustan a sus contenidos y los métodos propuestos en el libro de texto, dificultando el aprendizaje.

Definiéndose como **Objetivo general:** Desarrollar una herramienta informática multiplataforma de apoyo a la asignatura de Investigación de Operaciones que contribuya al tratamiento y visualización de sus contenidos y métodos.

Lo que determina como **Objeto de estudio:** El desarrollo de software, precisándose como **Campo de acción:** El desarrollo de software para la educación.

Para dar cumplimiento al objetivo propuesto, se han definido los siguientes **Objetivos específicos:**

1. Formalizar el marco teórico referencial del proceso de desarrollo de software para la educación.
2. Obtener el modelo de diseño de la herramienta.
3. Implementar la herramienta a partir del modelo de diseño.
4. Verificar la efectividad de la solución propuesta mediante pruebas de calidad de software y pruebas de aceptación a profesores de la asignatura.

A lo largo de esta investigación se utilizarán los siguientes Métodos científicos:

Métodos teóricos:

Análisis-síntesis: Para el estudio de los conceptos empleados relacionados con la IO.

Histórico–lógico: Para analizar, con mayor profundidad los antecedentes y las tendencias actuales referidas a las herramientas que sirven como soporte al PEA de la asignatura IO.

Métodos empíricos:

Experimental: Se utilizó para efectuar la ejecución de pruebas, creando las condiciones que permiten validar el correcto funcionamiento de la herramienta desarrollado en la presente investigación.

Observación: Se utilizó para apreciar el comportamiento de diferentes herramientas de trabajo como "WinQSB", "Trayectorias de Aumento", "OR Courseware" y "LINGO 13", para poder enfocar mejor el diseño y evitar errores.

La investigación está estructurada en 3 capítulos, como se muestra a continuación:

Capítulo 1: Fundamentación Teórica

En este capítulo se realiza un estudio del estado del arte de la problemática en cuestión. Se abordan los conceptos principales relacionados con la investigación para el desarrollo de la solución al problema planteado. Se realiza un análisis de las herramientas existentes y sus principales deficiencias. Además, se explican la metodología de desarrollo de software y herramientas utilizadas para dar solución a la problemática planteada.

Capítulo 2: Análisis y diseño de la aplicación

En este capítulo se identifican todos los requerimientos funcionales y no funcionales de la herramienta a implementar. Se muestran los principales procesos a través de las Historias de Usuario. Se presenta la estimación del esfuerzo necesario para el desarrollo de la solución, así como el plan de iteraciones y el de entregas. Se elaboran las tarjetas CRC, para un mejor entendimiento del código a generar. Además, se establece el diseño arquitectónico de la solución propuesta.

Capítulo 3: Implementación y validación de la aplicación

Se tratan los aspectos relacionados con la implementación de la solución propuesta. Además, se valida la misma a través de las pruebas que se le realizan al sistema.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En el presente capítulo se realiza un análisis sobre el proceso de desarrollo de software educativo, así como las metodologías existentes para llevar a cabo dicho proceso. Se desglosan los temas de la asignatura en la universidad y las necesidades prácticas de los mismos, así como las herramientas existentes para el trabajo de la misma, sus características y deficiencias. Se realiza además un análisis de las tecnologías y herramientas a utilizar en el desarrollo de la solución propuesta.

1.2 Suite UNIMATH

El departamento de Ciencias Básicas de la facultad 3 de la universidad define la suite llamada UNIMATH, compuesta por herramientas de apoyo a la docencia. Para el desarrollo de dicha suite, el departamento define como línea tecnológica el empleo de Pyhton 3.5 como lenguaje de programación, Django 1.8 como marco de trabajo y Bootstrap 3.0 como marco de trabajo web (ver Anexo 10). Esta línea tecnológica permite además de incluir todas las facilidades de la web al desarrollo, la ejecución de la herramienta en distintas estaciones de trabajo o en un servidor al cual tengan acceso los usuarios. UNIMATH contendrá todas las soluciones creadas por y para el departamento, facilitando el proceso de enseñanza aprendizaje y los trabajos metodológicos en los colectivos de las asignaturas pertenecientes al departamento.

Dado el hecho de que la presente investigación está dirigida a desarrollar una herramienta de apoyo a la docencia para una asignatura perteneciente a dicho departamento, el desarrollo de dicha solución está regido por las especificaciones de la suite. Además, la utilización de una arquitectura de desarrollo única para las matemáticas en el departamento docente creará homogeneidad en las soluciones.

La herramienta resultante de la presente investigación, además de poder ser integrable a la suite UNIMATH, debe ser capaz de funcionar independientemente, ya sea en una sola estación o accediendo a través de un servidor web.

1.3 Desarrollo del software educativo

En la actualidad el auge alcanzado por las TIC ha impulsado el uso de la informatización en todas las esferas de la sociedad, lo cual no deja de lado el ámbito educacional. La utilización de dicha tecnología en la educación ha causado un gran impacto en la calidad del PEA, permitiendo una mayor interacción

estudiante-profesor-tecnología. A las herramientas utilizadas para dichos fines se les conoce como software educativo.

El autor (Cataldi, 2000) define estas herramientas como “los programas de computación realizados con la finalidad de ser utilizados como facilitadores del proceso de enseñanza y consecuentemente del aprendizaje, con algunas características particulares tales como: la facilidad de uso, la interactividad y la posibilidad de personalización de la velocidad de los aprendizajes”.

Por otra parte el autor (Marquès, 1995) plantea que la terminología software educativo es utilizada para designar genéricamente los programas para ordenador creados con la finalidad específica de ser utilizados como medio didáctico, es decir, para facilitar los procesos de enseñanza y de aprendizaje. También tienen su propia definición los autores (Ledo, y otros, 2010), los que plantean de forma genérica a los softwares educativos como aplicaciones o programas computacionales que faciliten el proceso de enseñanza aprendizaje.

De lo anteriormente planteado por estos autores, se propone como concepto de software educativo: toda aquella herramienta informática destinada al tratamiento de contenidos educativos, ya sea para el profesor o para el estudiante. Debe brindar la posibilidad de evaluar el trabajo para el cual está siendo utilizada, con interfaces y métodos en los que prime la facilidad de uso, la interactividad y el uso del computador; incentivando así el PEA.

Los softwares educativos o programas didácticos, como también se les conoce, pueden clasificarse de diversas formas atendiendo a diferentes indicadores. Varios autores proponen diferentes clasificaciones, que llegan a incluir 23 categorías de programas. Las mismas son a su vez muy sensibles al paso del tiempo debido al vertiginoso avance tecnológico y en muchos casos las comparaciones que se hacen son inadecuadas.

Debido a este gran número de categorías y para un mejor entendimiento de la presente investigación se utiliza la clasificación propuesta por (Marquès, 1995), el mismo las divide en 4 grandes grupos como se muestra a continuación:

Tutoriales: son programas que dirigen el aprendizaje de los alumnos mediante una teoría subyacente conductista de la enseñanza, guían los aprendizajes y comparan los resultados de los alumnos contra

patrones, generando muchas veces nuevas ejercitaciones de refuerzo, si en la evaluación no se superaron los objetivos de aprendizaje.

Simuladores: ejercitan los aprendizajes inductivo y deductivo de los alumnos mediante la toma de decisiones y adquisición de experiencia en situaciones imposibles de lograr desde la realidad, facilitando el aprendizaje por descubrimiento.

Entornos de programación: permiten construir el conocimiento, paso a paso, facilitando al alumno la adquisición de nuevos conocimientos y el aprendizaje a partir de sus errores; y también conducen a los alumnos a la programación.

Herramientas de autor: también llamadas "lenguajes de autor" que permiten a los profesores construir programas del tipo tutoriales, especialmente a profesores que no disponen de grandes conocimientos de programación e informática, ya que usando muy pocas instrucciones, se pueden crear muy buenas aplicaciones hipermediales.

El propio autor hace una propuesta de las funciones que pueden cumplir las herramientas, determinadas por el uso que les dé cada profesor a las herramientas. En la siguiente tabla se muestran las mismas de forma sintética (ver Tabla 2):

Tabla 2 *Funciones del software educativo según el uso que le dé el profesor (Marquès, 1995).*

Función	Descripción
Informativa	Presentan contenidos que proporcionan una información estructuradora de la realidad. Representan la realidad y la ordenan.
Instructiva	Promueven actuaciones de los estudiantes encaminada a facilitar el logro de los objetivos educativos.
Motivadora	Suelen incluir elementos para captar en interés de los alumnos y enfocarlo hacia los aspectos más importantes de las actividades.

Evaluadora	Al evaluar explícita o implícitamente el trabajo de los alumnos.
Investigadora	Los más comunes son: las bases de datos, los simuladores y los entornos de programación.
Expresiva	Por la precisión de los lenguajes de programación, ya que el entorno informático, no permite ambigüedad expresiva
Metalingüística	Al aprender lenguajes propios de la informática.
Lúdica	A veces, algunos programas refuerzan su uso, mediante la inclusión de elementos lúdicos.
Innovadora	Cuando utilizan la tecnología más reciente

Acorde a lo anteriormente planteado la suite UNIMATH se clasifica como un entorno de programación. Al mismo tiempo se persigue que la misma cumpla con las funciones anteriormente definidas. Dado que la propuesta de solución de la presente investigación debe ser integrable a la suite UNIMATH, la herramienta también se clasifica como un entorno de programación. En cuanto a las funciones, la herramienta propuesta cumplirá cuatro de las nueve funciones.

Informativa: la herramienta le ofrecerá información real a estudiantes y profesores, en cuanto a los temas tratados en la asignatura. La información además estará estructurada de acuerdo a los temas tratados en la asignatura. Dicha información en su mayoría provendrá de la bibliografía básica de la asignatura.

Instructiva: la herramienta promoverá actuaciones por parte de los estudiantes, encaminadas a facilitar el logro de los objetivos educativos. Con la presentación de ejemplos prototipos y los métodos de resolución de los mismos, los estudiantes deberán ser capaces de comprender los contenidos y enfrentarse a otros problemas de la asignatura.

Expresiva: la herramienta tratará los términos de la asignatura, dotando así al estudiante del lenguaje técnico utilizado en la asignatura, evitando ambigüedades en el uso de los mismos.

Investigadora: le permite al estudiante la comparación de los resultados obtenidos manualmente con los resultados brindados por la herramienta. Además, le permitirá comparar cómo se comportan los problemas al cambiar variables y valores.

Acorde a lo anteriormente planteado y teniendo en cuenta las características de la solución propuesta, la misma puede ser clasificada como un Entorno de programación, ya que permitirá que el alumno adquiera nuevos conocimientos y aprenda de sus errores al utilizarla. Entre sus principales funciones figuran: informativa, instructiva, expresiva e investigativa.

1.4 Descripción de la asignatura en la universidad

La IO, Investigación Operativa o Investigación Operacional (conocida también como teoría de la toma de decisiones o programación matemática) es una rama de las matemáticas que consiste en el uso de modelos matemáticos, estadística y algoritmos, con el objetivo de realizar un proceso de toma de decisiones. Frecuentemente estudia sistemas reales complejos, con la finalidad de mejorar (u optimizar) su funcionamiento. La IO permite el análisis para la toma de decisiones teniendo en cuenta la escasez de recursos, para determinar cómo se puede optimizar un objetivo definido, como la maximización de los beneficios o la minimización de costos (S. Hillier, y otros, 1997).

La asignatura es impartida en el segundo semestre del tercer año académico de la carrera. La misma está dividida en cuatro temas:

1. **Modelación** (se basa en la modelación de los problemas de PL, ya sea entera, mixta o bivalente, utilizando el modelo matemático compuesto por función objetivo y restricciones. Consta de dos conferencias, cinco clases, para un total de 16 horas/clase).
2. **Métodos de solución** (se basa en la utilización de los métodos gráficos y el método “Símplex” para la resolución de los modelos matemáticos aprendidos en el tema uno de la asignatura, así como el análisis posóptimo, dígase precios sombra, sensibilidad de los coeficientes de la función objetivo y los términos independientes. Consta de tres conferencias, seis clases prácticas, para un total de 20 horas/clase)

3. **Problemas de redes** (se basa en el estudio de la resolución de problemas lineales de redes, ya sea de transporte o de asignación, su modelación y resolución a través de los problemas de flujo máximo y flujo de costo mínimo. Consta de dos conferencias, cinco clases prácticas, para un total de 16 horas/clase).
4. **Apoyo a la toma de decisiones** (se basa en el estudio del análisis de decisiones, sin experimentación utilizando la probabilidad a posteriori. Consta de dos conferencias y cuatro clases prácticas, para un total de doce horas/clase).

Cada uno de estos temas es la continuidad del anterior. De conjunto dotan al estudiante de un amplio grupo de conocimientos que le permitirán tomar decisiones una vez se enfrenten a problemas de optimización, tanto en la vida real como en el ámbito académico.

La PL utiliza un modelo matemático para describir el problema. El adjetivo lineal significa que todas las funciones matemáticas del modelo deben ser funciones lineales. En este caso, la palabra programación no se refiere aquí a términos computacionales; en esencia es sinónimo de planeación. Por lo tanto, la programación lineal involucra la planeación de actividades para obtener un resultado óptimo; esto es, el resultado que mejor alcance la meta especificada —de acuerdo con el modelo matemático— entre todas las alternativas factibles. (S. Hillier, y otros, 1997)

Los modelos de PL son frecuentemente usados para abordar una gran variedad de problemas de naturaleza real en ingeniería y ciencias sociales, lo que ha permitido a empresas y organizaciones importantes beneficios y ahorros asociados a su utilización. De aquí la importancia del estudio de dicha asignatura en la carrera.

Para llevar las situaciones prácticas a la PL en aras de obtener beneficios se transita en dos fases fundamentales: modelación y solución del modelo planteado. El objetivo fundamental de la PL aplicado a la IO es la distribución de recursos entre actividades competitivas, ya sea para disminuir costos o aumentar ganancias. Persiguiendo el cumplimiento de ese objetivo en la fase de modelación se definen las variables presentes en la situación planteada, así como su influencia numérica sobre el resultado final. Posteriormente, se define la función objetivo, que será la contabilizadora del aporte que realicen las variables de decisión a la solución. Dado el hecho que los problemas son basados en situaciones reales, los recursos con los que cuenta la empresa no son ilimitados y para ello se definen las restricciones que

serán los topes para estos recursos. Las mismas pueden ser de dos tipos: cota superior y cota inferior. El modelo anteriormente planteado es la vía utilizada para solucionar el problema, para ello la fase de solución persigue como objetivo encontrar la mejor combinación de valores finales para la variable, cumpliendo el objetivo de la función objetivo y sin violar las restricciones.

Entre los principales métodos de solución que plantea la bibliografía básica de la asignatura se encuentran: el Símplex, el gráfico y el ramificación y acotación (branch and bounds).

Método Símplex: es un proceso iterativo que en cada iteración permite ir mejorando la solución. Hace uso de la propiedad de que la solución óptima de un problema de PL se encuentra en un vértice o frontera del dominio de puntos factibles, por lo cual, la búsqueda secuencial del algoritmo se basa en la evaluación progresiva de estos vértices hasta encontrar el óptimo. Haciéndolo idóneo para el apoyo en la toma de decisiones en entornos empresariales principalmente (Programación Lineal, 2014). El proceso concluye cuando no es posible seguir mejorando más dicha solución. (S. Hillier, y otros, 1997)

Método gráfico: es un procedimiento de solución de problemas de programación lineal muy limitado en cuanto al número de variables (sólo dos), pero muy rico en materia de interpretación de resultados e incluso análisis de sensibilidad. Este consiste en representar cada una de las restricciones y encontrar en la medida de lo posible el polígono factible, comúnmente llamado el conjunto solución o región factible, en el cual por razones trigonométricas en uno de sus vértices se encuentra la mejor respuesta (solución óptima). (S. Hillier, y otros, 1997) Este método es la base de la comprensión del método Símplex, por lo que solamente se utiliza para darle introducción al mismo, no como método de evaluación de la asignatura.

Método ramificación y acotación: aborda la resolución de modelos de programación entera a través de la resolución de una secuencia de modelos de programación lineal que constituirán los nodos o subproblemas del problema entero. Recibe su nombre precisamente por las dos técnicas en las que basa su desarrollo, que son la ramificación y la acotación. La ramificación consiste en dividir cada problema en dos nuevos subproblemas, obtenidos mediante la imposición de restricciones excluyentes que dividen el conjunto de oportunidades del problema original en dos partes, pero eliminando en ambas partes la solución no entera del problema original. La acotación se basa en el hecho de que dado que los conjuntos de oportunidades del subproblema 1.1 y del subproblema 1.2 son a su vez subconjuntos del conjunto de

oportunidades del problema 1 la solución óptima de los dos subproblemas siempre será inferior que la solución óptima del problema 1 por ser los conjuntos de elección menores. (S. Hillier, y otros, 1997) Este método utiliza como base el método Símplex, por lo que no constituye objetivo su evaluación en la asignatura.

1.4.1 Método Símplex

Para un mejor entendimiento del principal método de resolución de los problemas de PL, seguidamente se describe el proceso del mismo. Los pasos de ejecución son los siguientes:

1. **Modelar el problema de PL:** a partir de los datos presentes en el problema de PL, se deben primeramente definir la(s) variable(s) de decisión, la función objetivo y las restricciones.
2. **Obtención del modelo ampliado:** a partir de la modelación del problema de PL, se deben introducir variables que transformen las restricciones funcionales de desigualdad en restricciones de igualdad equivalentes. Las restricciones de no negatividad se dejan como desigualdades porque se manejan por separado. En el caso de restricciones del tipo menor o igual que (<=), se introduce una variable de holgura sumando en el miembro izquierdo de la inecuación, la misma no afecta la función objetivo pues su coeficiente en la misma es cero. En las restricciones del tipo igual a (=), se introduce una variable artificial sumando en el miembro izquierdo de la inecuación, además de modificar las restricciones, modifica también la función objetivo, pues en la misma tiene coeficiente M, el cual representa un número muy grande. En el caso de las restricciones del tipo mayor o igual que (>=), se introducen dos variables, una denominada superávit restando en el miembro izquierdo de la inecuación sin afectar la función objetivo, y una variable artificial del mismo modo que en las restricciones del tipo igual a (=). Finalmente, la función objetivo debe ser despejada, igualándose a cero.
3. **Preparación del modelo ampliado para aplicar el Símplex**: la introducción de variables con coeficiente M en la función objetivo representa la obtención de ganancias infinitas o el consumo de recursos infinitos. Por esta razón se hace necesario eliminar el valor M de las variables artificiales de la función objetivo. Para ello se toma la función objetivo y las restricciones que generaron los coeficientes M, se definen conjuntos ecuaciones y se resuelven de manera tal que se cumpla con el objetivo de eliminar los coeficientes M de la función objetivo. Una vez eliminados los mismos se puede pasar al siguiente paso.

4. **Iterar:** primeramente, se debe en dependencia del criterio de optimización de la función objetivo, buscar el mayor (en caso de minimización) o el menor (en caso de maximización) valor de la función objetivo. Este valor corresponde a la variable que pasará a formar parte de la solución base factible del problema en cuestión. La columna correspondiente a dicha variable se denomina columna pivote. Se calculan los coeficientes entre el lado derecho de la matriz de coeficientes y la columna de la variable entrante, de estos coeficientes resultantes, el menor positivo diferente de cero, indica la variable que debe salir de la solución base factible. La fila correspondiente a dicha variable se denomina fila pivote. El valor de la celda de intersección de la fila pivote y la columna pivote es denominado pivote. Para calcular los valores de la nueva iteración:
   - Columna pivote: la intersección de la fila de la variable entrante en la posición de solución base factible con la columna correspondiente a dicha variable toma valor uno, el resto de los valores de esta columna toman valor cero.
   - Fila pivote: todos y cada uno de los coeficientes correspondientes a la fila pivote se calculan dividiendo los mismos entre el valor pivote.
   - Demás coeficientes de la matriz: para calcular el resto de los valores se utilizan la fila pivote, la columna pivote y el pivote. Al valor actual de la tabla se le resta la división de la multiplicación del valor correspondiente en la fila pivote, con el valor correspondiente en la columna pivote entre el pivote.

   El proceso de iteración puede ser detenido si al calcular los coeficientes para hallar la variable que sale de la solución base factible, los mismos son todos negativos o igual a cero, en dicho caso no se puede determinar que variable debe salir de la solución base factible y por lo tanto se está en presencia de una solución no acotada o infinita. De no darse esta situación, el proceso de iteración se repite hasta alcanzar la solución óptima, que a su vez puede ser clasificada como:

   - Solución óptima múltiple: siempre que un problema tiene más de una solución básica factible óptima, al menos una variable no básica tiene coeficiente cero en la ecuación final correspondiente a la función objetivo, de manera que, si se aumenta su valor, el valor de la función objetivo no cambia. Por lo tanto, estas otras soluciones básicas factibles óptimas se pueden identificar (si se desea) realizando iteraciones adicionales del método Símplex, en las que cada vez se elige una variable no básica con coeficiente cero como variable básica entrante.

- Solución no factible o imposible: se reconoce que hay una solución no factible o imposible, cuando todos los coeficientes relativos indican que la solución es óptima, pero por lo menos, una variable artificial permanece en la solución base factible con valor mayor que cero.
- Solución óptima degenerada: Se reconoce que hay una solución óptima degenerada cuando el número de variables básicas con valor mayor que cero es menor que el número de restricciones funcionales en el modelo, otra forma de reconocerlo, es cuando en la última iteración del Símplex hay una variable básica con valor igual a cero en la solución óptima.

5. **Clasificar la solución:** de acuerdo a los criterios de solución anteriormente expuestos, una vez alcanzada la solución óptima se clasifica la misma.

A continuación, se realiza un análisis de las herramientas utilizadas en la asignatura. Es válido aclarar que los softwares utilizados, no clasifican la solución obtenida.

1.5 Herramientas utilizadas para el apoyo a la docencia

Como parte de la introducción de las TIC en las aulas de los centros educacionales, varias herramientas que antes eran utilizadas solamente en entornos empresariales, han sido acogidas como Herramientas de Apoyo a la Docencia (HAD). Las mismas permiten la realización de ejercicios de gran complejidad ahorrando tiempo y recursos. Aun teniendo esto en cuenta, no pueden ser utilizadas en todos los temas de las asignaturas, puesto que no están destinadas a su uso docente, además, se limitan en su mayoría a dar el resultado sin mostrar la vía de solución y los cálculos necesarios para alcanzar dicha solución.

La asignatura IO no está exenta a esta regla. En la resolución de los problemas de PL, se utiliza como principal método de resolución el Símplex. Dado que el mismo puede llegar a ser tan complejo como restricciones y variables de decisión tenga, se impone el uso de una HAD por parte de educadores y educandos de la materia, específicamente las herramientas “WinQSB” y Trayectorias de aumento. Estas herramientas son las propuestas por el Departamento Docente Central de la asignatura en la universidad.

“WINQSB”

“WINQSB” es un paquete de herramientas muy versátil que permite el análisis y resolución de modelos matemáticos, problemas administrativos, de producción, proyectos, inventarios y transporte. Ofrece una interfaz básica pero amigable, y es la aplicación por excelencia utilizada por profesionales de Ingeniería

Industrial y áreas administrativas para la resolución de sus modelos de PL, continua o entera (López, 2012).

En la UCI es utilizado actualmente como HAD, específicamente en el segundo semestre del tercer año en la asignatura IO, ya que entre sus prestaciones se encuentra la resolución de problemas de programación lineal con la utilización del método Símplex. A pesar del hecho de que su uso minimiza el tiempo de resolución de los ejercicios, el software presenta varias limitaciones para su uso por parte de los estudiantes. Entre estas complicaciones figuran:

- El sistema presenta la solución con un formato que no corresponde con la forma tabular del "Símplex" establecida en el libro de texto, lo cual dificulta el desarrollo del PEA, ya que el profesor tiene que dedicar parte del turno de clases a explicarles a los estudiantes como interpretar el resultado mostrado por el software.
- El sistema no permite la visualización del modelo ampliado.
- El sistema omite la variante utilizada para resolver el problema en cuestión, limitándose solo a proponer la solución final.
- El sistema no contempla un módulo que permita la modelación de los problemas de redes, Transporte y Asignación, para su posterior graficación y solución.
- Dado que está hecho para utilizarse en entornos empresariales, no presenta explicaciones ni ejemplos prototipos mediante los cuales los estudiantes puedan realizar comparaciones entre sus cálculos y los efectuados por el sistema.
- Solo funciona en la plataforma de Windows en los Sistemas Operativos de Windows 98, 2000 y XP.
- En los sistemas Linux solo es posible ejecutarlo con el emulador Wine.
- Posee una licencia privativa, lo cual atenta directamente en contra de las políticas de Software Libre del centro de estudios y el país.
- Está desarrollado en una versión muy antigua de Visual Basic, lo que dificulta su actualización.

Trayectorias de aumento:

Desarrollado por profesores pertenecientes al departamento de Ciencias Básicas de la Facultad 3 de la UCI como objeto de aprendizaje para mejorar la comprensión y asimilación del Algoritmo de Trayectorias de Aumento y del teorema de Flujo Máximo – Cortadura Mínima como métodos de solución del problema de Flujo Máximo. Está desarrollado en java, como aplicación Web y como applet[3].

Entre las principales desventajas que tiene figuran:

- Centra su trabajo en un método de solución de un tema de la asignatura solamente.
- No es posible extenderla a otros temas, ya que no se posee el código original de la aplicación.

Basado en las deficiencias de las herramientas que se utilizan actualmente en la asignatura, se decide realizar un análisis de otras herramientas afines. En este caso las herramientas OR Courseware y LINGO 13.

OR Courseware:

Courseware es un término que resulta de combinar las palabras en inglés “course” y “software”. Es un software diseñado para fines educativos que sirve como ayuda o refuerzo del tópico estudiado. Por su parte OR Courseware proviene del inglés Operational Research Courseware (entiéndase como: Software para cursos de IO). Dicha herramienta brinda soporte a los temas dos y tres de la asignatura impartida de la universidad, pero no se cuenta con su código fuente ni presenta servicios que permitan extenderla o integrarla, lo que la elimina como candidata para integrar la suite del departamento.

LINGO 13:

Es un software de la empresa Lindo Systems Inc. que permite resolver modelos matemáticos lineales y no lineales. Trabaja los problemas de PL de manera similar al “WinQSB”, por lo que no incluye el trabajo con los temas uno, tres y cuatro de la asignatura. Además, posee una sintaxis específica, del lenguaje LINDO, para introducir los datos de los problemas a resolver, lo que dificulta en cierto modo su utilización.

[3] **Applet:** Un applet es un programa escrito en Java y que forma parte de los componentes de una página de Internet. Los applets han sido usados para proporcionar funcionalidad a páginas de Internet que no puede ser satisfecha usando únicamente HTML.

El análisis de las herramientas de conjunto, evidencia la necesidad de desarrollar una nueva herramienta que cumpla con los requerimientos del cliente. A pesar de que las mismas no se ajustan a dichos requerimientos, se desea incluir algunas de sus funcionalidades para la concepción y desarrollo de la solución de la presente investigación como la visualización de todas las iteraciones del "WinQSB" y la graficación de "Trayectorias de Aumento". Para llevar a cabo el desarrollo de dicha solución se hace necesario seguir un conjunto de fases y pasos, definidos por las metodologías de desarrollo. En el siguiente epígrafe se hace alusión a las mismas.

1.6 Metodologías para el desarrollo de software educativo

Según (Pressman, 2010), "una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo." Las metodologías se basan en una combinación de los modelos de proceso genéricos. Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto, es trascendental para el éxito del producto. Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.

Los autores (Cataldi, 2000) y (Orjuela Duarte, y otros, 2008) en su estudio sobre el desarrollo de software educativos coinciden que dadas las características de los mismos, es recomendable su desarrollo utilizando metodologías de desarrollo ágil.

A partir de lo planteado hasta el momento, teniendo en cuenta que el equipo de trabajo del presente está formado por tres miembros (incluye los dos clientes como miembros del equipo de trabajo), con probabilidades de requisitos cambiantes, se decide emplear una metodología ágil.

Existen varias metodologías ágiles para regir el proceso de desarrollo de softwares, el autor (Orjuela Duarte, y otros, 2008) propone como principales las siguientes:

- **eXtreme Programming (XP):** XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los programadores, 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 (Beck, 1999).
- **Scrum:** es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales (Albaladejo, 2010).
- **Crystal:** se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir los esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (Fowler, y otros, 1999).

1.6.1 Selección de la metodología a utilizar

Después de vistas las metodologías recomendadas para el desarrollo ágil de software educativo, se selecciona la metodología XP. La misma se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y audacia para enfrentar los cambios. XP es principalmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Corrige los errores antes de

añadir una nueva funcionalidad, realizando entregas frecuentes del software para analizar su funcionamiento.

Dentro de las condiciones de ambiente para el desarrollo de este trabajo se tiene poco personal de desarrollo, pues solo se dispone de tres integrantes (los clientes forman parte del equipo de desarrollo), los requerimientos cambian a lo largo de todo el ciclo de vida de la aplicación. El equipo de trabajo tiene experiencia con el uso de ésta metodología y el cliente se siente cómodo con la generación de las Historias de Usuario (HU) como método para definir los requisitos del sistema. La existencia de estos factores principales determinó la selección de la metodología que, como especifican sus características, constituye una de las metodologías destacada dentro de los procesos ágiles para el desarrollo de un software, así como la experiencia del equipo de desarrollo con el uso de esta metodología y la disponibilidad del cliente.

Para un mejor proceso de desarrollo, XP define varias fases. Varios autores difieren en cuanto a las fases que deben componer el proceso de desarrollo. En la presente investigación se opta por la variante de fases propuesta por el creador de la metodología, Kent Beck, la que propone 4 fases relacionadas a continuación.

Tabla 3 *Fases que propone la metodología XP.*

Fases	Descripción
Planificación	Es la primera actividad en el proceso de desarrollo. Comienza creando una serie de HU que describen la funcionalidad del software que se va a construir. El cliente les asigna una prioridad y el equipo de desarrollo evalúa cada una y le asigna un periodo de desarrollo. Si la historia supera más de tres semanas de desarrollo se divide la historia en historias menores. Una vez establecido el acuerdo detallando la fecha de entrega, el equipo de desarrollo ordena las historias para implementar antes las que tengan mayor prioridad. Conforme avanza el trabajo de desarrollo, el cliente puede agregar nuevas HU con nuevas funcionalidades.
Diseño	Sigue el principio de hacerlo todo simple. El diseño se va modificando a lo largo de todo el proceso de desarrollo. Como principal artefacto en esta fase se obtienen las tarjetas CRC (Clase-Responsabilidad-Colaboradoras). Estas tarjetas se dividen en tres

	secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. En la práctica conviene tener pequeñas tarjetas de cartón, que se llenarán y que son mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas.
Codificación	Después de diseñar las historias el equipo no debe comenzar la codificación, sino que debe desarrollar una serie de pruebas de unidad que les ayuden a centrarse en lo que debe implementarse para pasar esa prueba. Durante esta fase se genera el código, el cual es el principal artefacto generado durante todo el proceso de desarrollo del software en cuestión. Para un mejor aprovechamiento de los recursos en la generación de dicho código, XP propone buenas prácticas como son: el juego de la planificación, entregas pequeñas, diseño simple, pruebas, cliente in-situ, estándares de programación, entre otras.
Pruebas	Las pruebas de unidad creadas deben ser automatizadas para que puedan ejecutarse de manera fácil y rápida. De esta forma se puede modificar el código y asegurar que funciona pese a los cambios producidos.

Para una mejor comprensión de las buenas prácticas definidas por la metodología en la frase de codificación, se explica en qué consisten a continuación:

El juego de la planificación: hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las HU y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

Entregas pequeñas: producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con todas las funcionalidades del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de tres meses.

Diseño simple: se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

Refactorización: es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

Integración continua: cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

Cliente in-situ: el cliente tiene que estar presente y disponible todo el tiempo para el equipo. Este es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportaría un mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.

Estándares de programación: XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos esquemas de programación para mantener el código legible. Además, posibilita un mejor entendimiento del código para futuros cambios que éste pueda sufrir.

1.7 Herramientas y Tecnologías

Una vez analizada la metodología, se hace necesario realizar el análisis de las herramientas que se utilizarán en el desarrollo de la solución de la presente investigación. La selección de estas herramientas está sujeta a las propuestas por la suite a la cual se integrará la solución. En este epígrafe se evaluarán y seleccionarán las tecnologías, técnicas y herramientas idóneas para el desarrollo de la solución.

1.7.1 Lenguaje de programación

Python 3.5

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos (Rossum, 2009). Por su parte el sitio oficial de Python (Foundation, 1990-2016) lo define como un lenguaje de programación orientado a objetos, interpretado e interactivo. Incorpora módulos, excepciones, tipado dinámico y clases. Python combina potencia con una sintaxis muy clara. Tiene interfaces para muchas llamadas y bibliotecas de sistema, así como a varios sistemas de ventanas. También es utilizable como un

lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Por último, Python es portátil: se ejecuta en muchas variantes de Unix, Mac y la plataforma Windows.

Creado por el matemático y científico de la computación holandés Guido van Rossum, este lenguaje es idóneo para el trabajo con aplicaciones matemáticas. Ya que para agilizar el desarrollo de las mismas, permite el uso de bibliotecas o módulos, entre los que destacan SciPy (provee al lenguaje de herramientas y algoritmos matemáticos para el trabajo con optimización, álgebra lineal, integración, interpolación, funciones especiales, procesamiento de señales y de imagen, entre otras tareas de la ciencia y la ingeniería (developers, 2016)) y NumPy (provee al lenguaje de funciones matemáticas de alto nivel para aplicar en vectores o matrices (developers, 2016)). Ambas bibliotecas inciden directamente en el desarrollo de la solución en cuestión. Además, provee la clase Fraction, la cual permite el uso de fracciones para el cálculo, mejorando así la visualización de los resultados una vez se vayan a mostrar en pantalla y la precisión de los cálculos realizados, ya que no redondea los valores.

Por otra parte, el intérprete estándar de Python incluye un modo interactivo, en el cual se escriben las instrucciones en una especie de intérprete de comandos. Las expresiones pueden ser introducidas una a una, mostrándose el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados. Lo anteriormente planteado hace de Python el lenguaje ideal para el desarrollo de la herramienta que satisfaga las necesidades de la presente investigación.

1.7.2 Marco de trabajo

“Un marco de trabajo web, está diseñado para apoyar el desarrollo de sitios web dinámicos. Ofrece un conjunto de componentes para acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones” (Gutiérrez, 2011).

Teniendo en cuenta que la herramienta resultante de la presente investigación se desea utilizar como parte del desarrollo de la suite del departamento, se escoge Django. Deduciéndose de lo anterior la necesidad de su uso en la presente solución, permitiendo una estructura conceptual y tecnológica de soporte para el desarrollo de sus funcionalidades.

Django 1.7.9

Django es un marco de trabajo web de Python de alto nivel que fomenta el rápido desarrollo, diseño limpio y pragmático. Construido por los desarrolladores con experiencia, que se encarga en gran parte de la molestia de desarrollo web, para que el programador pueda centrarse en la escritura de su aplicación sin necesidad de reinventar la rueda. Es de código abierto y libre (Foundation, 2005-2016).

Pone énfasis en la reutilización, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio DRY (del inglés Don't Repeat Yourself) consistente en hacer las cosas una sola vez y rehusarlas siempre que sea posible. Python es usado en todas las partes del marco de trabajo, incluso en configuraciones, archivos y en los modelos de datos. Mantiene de forma rigurosa un diseño limpio en su propio código y facilita que el programador siga las mejores prácticas de desarrollo web en las aplicaciones que crea. Fomenta el bajo acoplamiento.

El uso de este marco de trabajo web permitirá que la solución una vez terminada pueda ser montada en un servidor, al cual los profesores o estudiantes podrán acceder para su uso o en caso de que así lo prefieran, también pueda ser ejecutada en las computadoras personales. Lo planteado anteriormente hacen de Django el marco de trabajo web idóneo para el desarrollo de la solución en cuestión.

1.7.3 Otras herramientas y lenguajes

Bootstrap 3

Para la interacción de la vista con las clases controladoras, se utiliza el motor de plantillas de Django, el cual utiliza archivos HTML. En aras de ganar en estética y presentación, se utiliza el marco de trabajo Bootstrap. El mismo “permite crear interfaces web con CSS y Javascript. Las mismas adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, es decir, automáticamente se adapta al tamaño de un ordenador o de una *Tablet.* Todo esto ocurre sin que el usuario tenga que hacer nada, esto se denomina diseño adaptativo o *Responsive Design* “ (Sánchez, 2013). Entre las principales características que hicieron posible su selección, se destacan:

- Soporte HTML5 y CSS3.
- Soporte multi-navegador, incluido Internet Explorer.
- Permite definir diferentes diseños.

- integración de la biblioteca jQuery para diferentes efectos.
- Amplio conjunto de componentes para el desarrollo.
- Posibilidad de compilar el marco de trabajo con diferentes valores.
- Desarrollo de aplicaciones *Responsive Design*.

HTML5

Lenguaje de Marcado de Hipertexto o HTML. Lenguaje compuesto por cientos de etiquetas o marcas que permite definir el contenido y la apariencia de una página web en gran medida, además puede incluir script como JavaScript el cual afecta el comportamiento de los navegadores y otros procesadores de HTML, define una estructura de documento jerárquica, con elementos y componentes interconectados. (Mora, 2002)

CSS3

CSS[4] desarrollada por la W3C[5], es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML que separa los contenidos y su presentación por lo que es imprescindible para crear páginas web complejas (Eguíluz, 2009). Permite definir la apariencia de cada elemento de la aplicación web: colores, fondos, márgenes, bordes, tipos de letra, modificando la apariencia y posición de cada elemento dentro de la página. Lo que permite controlar el estilo y formato de sus documentos.

Ofrece, además:

- Fácil creación de las plantillas al mantener la misma imagen en todas las páginas del sitio.
- Mayor accesibilidad y limpieza del código fuente.
- Código HTML más legible.
- Logra que los documentos se vean igual en todos los navegadores.
- Optimización de los tiempos de carga y el tráfico del servidor.

[4] CSS: hojas de estilo en cascada, es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML)

[5] W3C: son las siglas de *World Wide Web Consortium*, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución y aseguren su interoperabilidad. (Gaitan, 2016)

JavaScript 1.8.5

Los autores (Mora, 2002) y (Zakas, 2009) constatan que es un lenguaje interpretado que no necesita ser compilado, solo es necesario un navegador para su interpretación por lo que se ejecuta del lado del cliente. Creado con el objetivo de hacer páginas web dinámicas, es orientado a objeto y guiado por eventos que produce el mismo usuario, haciendo amena la interactividad con las páginas mediante efectos tales como: cambio de color de algunos elementos de la página, crear páginas interactivas con programas como calculadoras, agendas, tablas y calendarios.

Es un lenguaje multiplataforma siendo interpretado por la mayoría de los navegadores modernos. Sencillo, rápido y fácil de aprender por personas de poca experiencia. Permite reducir la carga en el servidor, ya que los datos incorrectos se filtran en el cliente y no se envían al servidor.

1.7.4 Entorno Integrado de Desarrollo.

Pycharm 4.5

PyCharm es un Entorno de Desarrollo Integrado (por sus siglas en inglés, IDE). Desarrollado por la compañía JetBrains[6], está basado en IntelliJ IDEA[7], el IDE de la misma compañía, pero enfocado hacia Java y la base de Android Studio. Pycharm tiene cientos de funciones que lo puede hacer ver como una herramienta muy pesada, pero que valen la pena ya que ayuda con el desarrollo del día a día. (Graterol, 2014)

El IDE en cuestión cuenta con varias ventajas, entre las que destacan:

- Autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización.
- Integración con marcos de trabajo web como: Django, Flask, Pyramid, Web2Py.
- Integración con marcos de trabajo *javascripts como*: jQuery, AngularJS.
- *Debugger* avanzado de Python y Javascript.

---

[6] Jetbrains: fundada en la República Checa en el año 2000, es una empresa de desarrollo de software cuyas herramientas están dirigidas a los desarrolladores de software y administradores de proyectos. La compañía ofrece una extensa familia de entornos de desarrollo integrado para los lenguajes de programación Java, Ruby, Python, PHP, SQL, Objective-C, C ++, y JavaScript. Actualmente su director ejecutivo es Maxim Shafirov. (JetBrains, 2000-2016)

[7] *IntelliJ IDEA*: es un entorno de desarrollo integrado para el desarrollo de programas informáticos. Es desarrollado por *JetBrains*

- Integración con lenguajes de plantillas: Mako, Jinja2, Motor de Plantillas de Django.
- Soporta entornos virtuales e intérpretes de Python 2.x, 3.x, PyPy, Iron Python y Jython.
- Sistemas de control de versiones: Git, CVS, Mercurial.

Además, Pycharm cuenta con un servidor que posibilita la ejecución del código al mismo tiempo que es generado, permitiendo así medir la completitud y satisfacción del cliente con lo concebido hasta el momento.

1.8 Conclusiones parciales del capítulo

Durante el desarrollo del capítulo se pudo identificar que las herramientas existentes para el apoyo a la docencia de la asignatura IO no pueden ser debidamente utilizadas debido a los requerimientos y necesidades del Departamento de Ciencias Básicas de la Facultad 3 de la UCI. La imposibilidad de la aplicación o adaptación de alguna de estas herramientas en los temas de dicha asignatura, evidenció la necesidad de la creación de una solución que permita la integración de los temas de la misma, así como la interoperabilidad de dicha solución.

Las herramientas y tecnologías a utilizar en el desarrollo de la herramienta se encuentran subordinadas a una investigación más amplia, de la cual el presente trabajo forma parte. El lenguaje de programación a emplear es Python y el marco de trabajo a utilizar es Django. Para el trabajo con las vistas se empleará HTML5, CSS3 y el marco de trabajo web Bootstrap, y para el manejo de eventos JavaScript. La metodología de desarrollo a emplear es XP. El estudio de estas herramientas y tecnologías le permitió al equipo de desarrollo familiarizarse con el marco de trabajo y el lenguaje de programación.

Capítulo 2 Análisis y diseño de la solución

2.1 Introducción

En el presente capítulo se realiza la descripción de la propuesta de solución y se detallan los principales requerimientos de la aplicación. Además, se describen las historias de usuarios para cada requisito funcional identificado. También, se explica el uso de los patrones de diseño empleados para la implementación del sistema y el patrón arquitectónico a aplicar en el mismo.

2.2 Descripción de la propuesta de solución

Como propuesta de solución, se requiere una herramienta que cumpla las necesidades del cliente. Para ello la herramienta debe constar de un componente informativo el cual se encargue de darle soporte al primer tema de la asignatura de modelación de los problemas de PL. En este componente se deberá explicar en qué consisten los problemas de PL, ya sea entera, bivalente o mixta, además de los problemas de redes de transporte, asignación, flujo máximo y flujo de costo mínimo. Debe también explicar cómo se declaran las variables de decisión, como se formulan la función objetivo y las restricciones. Además, se requiere que proponga varios ejemplos prototipos, donde se muestre la resolución completa de los mismos, así como la interpretación de la solución una vez alcanzada la misma.

La herramienta también debe dar soporte al tema dos de la asignatura, permitiendo crear y resolver problemas de PL. Debe mostrar el proceso y resultado utilizando la forma tabular del método Símplex propuesta por la bibliografía básica de la asignatura. Debe, además, mostrar el modelo ampliado donde se introducen las variables de holgura, superávit y artificiales, de manera clara, para que los estudiantes puedan realizar un posterior análisis de las mismas. Para un mejor entendimiento de la solución del problema, la herramienta debe permitir la visualización de todas las iteraciones del método Símplex para el problema en cuestión y la solución del mismo orientada al análisis pos óptimo, permitiendo la interpretación por parte del usuario.

Además, debe brindar la posibilidad de crear y modelar, (tanto matemática, como gráficamente), problemas de transporte y asignación. En el caso de la solución gráfica los nodos origen deben mostrarse de color verde, los destinos de color azul y en el caso de los problemas que así lo requieran los nodos ficticios serán de color rojo, permitiéndole una mejor identificación de esta tipología de nodos al usuario. La aparición de los nodos ficticios está condicionada por el hecho de que el problema no esté balanceado,

el método de solución propone la creación de estos nodos para balancear estos problemas. La herramienta deberá balancear los problemas basándose en los criterios propuestos por la bibliografía básica de la asignatura y al mismo tiempo dar la opción para asignar los valores asociados a los mismos.

Finalmente, para cubrir a totalidad los temas a tratar en la asignatura, la herramienta deberá brindar información referente a la toma de decisiones, presente en el tema cuatro de la misma. Así como permitir la creación y resolución de problemas de esta índole. Para la resolución, se hará a través de los métodos de la toma de decisiones sin experimentación: criterio del pago máximo, criterio de la máxima posibilidad y la regla de decisión de Bayes[8].

2.3 Requisitos del sistema

"Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información"(Brito, 2012). La metodología seleccionada en cuanto al levantamiento de requisitos funcionales del sistema, propone en lugar de los mismos, declarar HU, las que posteriormente se desglosarán en Requisitos funcionales (RF) del sistema, para lograr un mejor entendimiento de las mismas.

2.3.1 Historias de usuarios

Una HU es una representación de uno o varios requisitos de software escritos, de una o dos frases, utilizando el lenguaje común del usuario. Las HU son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos de forma general por parte del cliente. Además, se consideran una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir mucho tiempo para administrarlos, permitiendo responder rápidamente a los requisitos cambiantes. (Cohn, 2004)

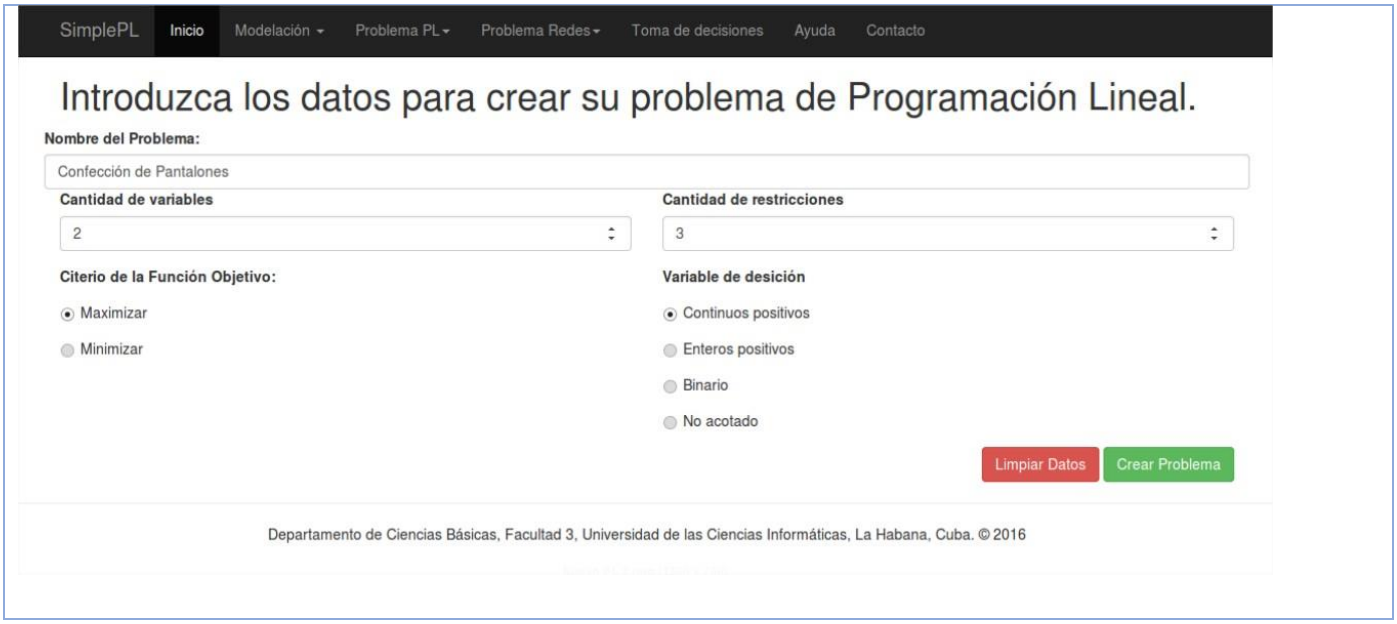
Durante el diseño de la propuesta de solución se identificaron seis historias de usuario que responden a las diferentes funcionalidades solicitadas por el cliente y presentan una descripción para que el

[8] Bayes: El origen de este nombre es que con frecuencia se da crédito por él al Reverendo Thomas Bayes, un ministro inglés no conformista del siglo xviii reconocido como filósofo y matemático. (La misma idea básica tiene raíces más antiguas en el campo de la economía.) Esta regla de decisión también suele llamarse criterio del *valor monetario esperado* (*VME*), no obstante que es un mal nombre para los casos en que la medida de pago es otra que el valor monetario. (S. Hillier, y otros, 1997)

desarrollador conozca su posterior implementación. A continuación, se muestra un ejemplo de las HU, el resto está en los anexos 1, 2, 3, 4 y 5 del documento:

Tabla 4 *Historia de usuario - Insertar problema de Programación Lineal.*

Historia de Usuario	
Número 2	**Nombre de la Historia de Usuario:** Insertar problema de Programación Lineal.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Estándar.	**Iteración asignada:** 1
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Bajo.
Descripción: El sistema debe permitir al usuario la inserción de los datos necesarios para la modelación del problema de Programación Lineal. Para ello el actor involucrado debe de llenar el formulario que el sistema le provee. Dicho formulario cuenta con los siguientes campos: **"Nombre del Problema", "Cantidad de Variables", "Cantidad de Restricciones"** y **"Criterio de la Función Objetivo".** Donde el campo **"Nombre del Problema"** se refiere al nombre del problema que se pretende crear, este campo admite valores alfanuméricos, el campo **"Cantidad de Variables"** hace referencia a la cantidad de variables involucradas en el problema, este campo admite sólo valores numéricos (*un número <= 20*), el campo **"Cantidad de Restricciones"** hace referencia a la cantidad de restricciones por las cuales está regida el problema, este campo admite sólo valores numéricos (*un número <= 20*) y el campo **"Criterio de la Función Objetivo"** hace referencia al fin que está destinado el problema, puede tomar solamente dos valores (Maximizar, Minimizar). Una vez llenados dichos campos el estudiante o profesor debe de hacer clic en el botón **"Crear Problema"** y automáticamente el sistema redirecciona al actor a la página para introducir los valores necesarios para el procesamiento del problema. En caso de presentar algún error en el contenido introducido por el estudiante o profesor en el formulario, se muestra un mensaje de error en el campo donde se cometió el mismo y se mantiene en el formulario actual. También, si el usuario determina que los datos no están correctos y desea borrarlos para introducir nuevos, el botón **"Limpiar Datos"** le facilita esta acción.	
Prototipo:	



2.3.2 Requisitos funcionales

"Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares" (Brito, 2012). Acorde a lo planteado por la metodología de desarrollo anteriormente seleccionada, las seis HU ya definidas, buscando facilitar su implementación; se desglosan en los siguientes requisitos funcionales (RF):

Tabla 5 *Requisitos Funcionales del sistema.*

Historia de Usuario	No. RF	Nombre del RF	Descripción
Mostrar componente informativo	RF. 1	**Mostrar ejemplo prototipo de PL entera.**	El sistema debe permitir la visualización de ejemplos prototipos de modelación sobre PL entera.
	RF. 2	**Mostrar ejemplo prototipo de PL entera bivalente.**	El sistema debe permitir la visualización de ejemplos prototipos de modelación sobre PL entera bivalente.
	RF. 3	**Mostrar ejemplo prototipo de PL**	El sistema debe permitir la visualización de ejemplos prototipos de modelación

		entera mixta.	sobre PL entera mixta.
	RF. 4	**Mostrar ejemplo prototipo de problemas de transporte.**	El sistema debe permitir la visualización de ejemplos prototipos de modelación sobre problemas de transporte.
	RF. 5	**Mostrar ejemplo prototipo de problemas de asignación.**	El sistema debe permitir la visualización de ejemplos prototipos de modelación sobre problemas de redes.
Insertar problema de Programación Lineal	RF. 6	**Crear problema.**	El sistema debe permitir la introducción de los datos necesarios para la creación del problema de PL.
	RF. 7	**Introducir coeficientes del problema.**	El sistema debe permitir la introducción de los coeficientes para el posterior trabajo con el problema.
	RF. 8	**Calcular solución del problema.**	El sistema debe ser capaz a partir de los datos introducidos calcular la solución al problema en cuestión.
	RF. 9	**Mostrar solución del problema.**	El sistema debe mostrar al usuario la solución final del problema de PL.
	RF. 10	**Mostrar solución paso a paso del problema.**	El sistema debe permitir al usuario visualizar las n iteraciones necesarias para solución del problema de PL.
Insertar problema de redes	RF. 11	**Crear problema de transporte.**	El sistema debe permitir la creación de un problema de transporte a partir de la tabla de costos.
	RF. 12	**Crear problema de**	El sistema debe permitir la creación de un

		asignación.	problema de asignación a partir de la tabla de costos.
	RF. 13	**Crear modelo de problema de transporte.**	El sistema debe permitir la creación del modelo de un problema de transporte a partir de los datos procesados.
	RF. 14	**Crear modelo de problema de asignación.**	El sistema debe permitir la creación de un problema de asignación a partir de los datos procesados.
	RF. 15	**Graficar red de problema de transporte.**	El sistema debe permitir graficar la red correspondiente al problema de transporte en cuestión.
	RF. 16	**Graficar red de problema de asignación.**	El sistema debe permitir graficar la red correspondiente al problema de asignación en cuestión.
Calcular problema de redes de flujo máximo	RF. 17	**Calcular solución utilizando el método "Cortadura mínima".**	El sistema debe permitir calcular la solución del problema en cuestión, utilizando el método de resolución "Cortadura mínima".
	RF. 18	**Calcular solución utilizando el método "Trayectorias de aumento".**	El sistema debe permitir calcular la solución del problema en cuestión, utilizando el método de resolución "Trayectorias de aumento".
Calcular problema de redes de flujo de costo mínimo	RF. 19	**Modelar el problema de redes de flujo de costo mínimo.**	El sistema debe modelar el problema de redes de flujo de costo mínimo a partir de los datos introducidos por el usuario.
Toma de decisiones	RF. 20	**Mostrar ejemplos prototipos de toma**	El sistema debe permitirle al usuario visualizar ejemplos prototipos de toma de

		de decisiones.	decisiones.
Calcular problemas de toma de decisiones sin experimentación.	RF. 21	**Crear problema de toma de decisiones.**	El sistema debe permitir la introducción de los datos necesarios para la creación del problema de PL.
	RF. 22	**Introducir coeficientes del problema.**	El sistema debe permitir la introducción de los coeficientes para el posterior trabajo con el problema.
	RF. 23	**Calcular problemas de toma de decisiones sin experimentación.**	El sistema debe permitirle al usuario visualizar la solución de los problemas de toma de decisiones sin experimentación, utilizando los métodos criterio del pago máximo, criterio de la máxima posibilidad y la regla de decisión de Bayes.

2.3.3 Requisitos no funcionales

"Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluye restricciones de tiempo, sobre el proceso de desarrollo y estándares" (Brito, 2012). Para el desarrollo de la aplicación se capturaron los siguientes requisitos no funcionales (RnF):

Tabla 6 *Requisitos no Funcionales del sistema.*

No.	Tipo de RnF	Descripción
1	**Usabilidad**	- El sistema a desarrollar deberá poseer un dominio de aplicación web 2.0. - El sistema podrá ser utilizado por personas que tengan un conocimiento básico en el manejo de las computadoras, por lo cual debe presentar un acceso fácil y rápido.
2	**Software**	- La aplicación deberá de correr en sistema operativo Windows 7 o superior, o GNU Linux en cualquiera de sus distribuciones.

		- Para interactuar con el sistema se deberá utilizar un navegador web, preferiblemente Firefox en una versión superior a la 30. - La computadora donde esté ejecutándose la aplicación, deberá tener un intérprete de Python 3, el módulo de Python para Django 1.8 y las bibliotecas SciPy y NumPy.
3	**Hardware**	- Capacidad del disco duro superior a los 5 Giga bytes. - Procesador Intel Pentium Dual Core a 1.6 GHz de velocidad de procesamiento, equivalente o superior. - Se requiere un mínimo de 512 Mega bytes de RAM
4	**Interfaz de usuario**	- En el sistema las interfaces estarán sustentadas por los colores: blanco y negro. Por otra parte, contará con un menú en la parte superior el que incluye las funcionalidades específicas. - Las interfaces del sistema contendrán los datos de forma estructurada, permitiendo la correcta interpretación de la información. - La entrada incorrecta de datos será mostrada al usuario claramente, señalando los campos donde se encuentra el error.
5	**Fiabilidad**	- Ante una inserción de datos, vista de detalles o modificación, el sistema debe responder en no más de tres segundos.

2.4 Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y teniendo en cuenta esto el programador realiza una estimación de esfuerzo necesario para cada una de ellas. Se realizan acuerdos sobre el material a entregar en la primera iteración y en correspondencia se genera un cronograma junto al cliente.

2.4.1 Estimación de esfuerzos

La medida utilizada para la estimación del esfuerzo asociado a la implementación es el punto. Un punto equivale a una semana ideal de programación. Esta medida generalmente toma valores de uno a tres.

Tabla 7 *Plan de estimación de esfuerzo.*

Nº	Historia de usuario	Punto de estimación
1	Mostrar ejemplo de modelación.	1
2	Insertar problema de Programación Lineal.	2
3	Insertar problema de redes.	1
4	Calcular problemas de redes de flujo máximo.	2
5	Calcular problemas de redes de flujo de costo mínimo.	2
6	Mostrar ejemplo prototipo de toma de decisiones.	1
7	Calcular problemas de toma de decisiones sin experimentación.	1

2.4.2 Plan de Iteraciones

Para cada entrega fueron escogidas algunas HU, teniendo en cuenta el orden definido. Este plan define las HU que deben ser implementadas en cada iteración y las fechas de liberación. A continuación, se muestran dos iteraciones y el número de historias por cada una.

Tabla 8 *Plan de iteraciones.*

Iteración	Historias de usuario	Duración total (semanas)

1	Mostrar ejemplo de modelación.	4
	Insertar problema de Programación Lineal.	
	Insertar problema de redes.	
2	Calcular problemas de redes de flujo máximo.	6
	Calcular problemas de redes de flujo de costo mínimo.	
	Mostrar ejemplo prototipo de toma de decisiones.	
	Calcular problemas de toma de decisiones sin experimentación.	

2.4.3 Plan de entregas

En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación.

Tabla 9 *Plan de entrega de las iteraciones.*

Artefacto	Iteración	Entrega
Mostrar ejemplo de modelación.	1	12 de mayo
Insertar problema de Programación Lineal.		
Insertar problema de redes.		
Calcular problemas de redes de flujo máximo.	2	23 de junio
Calcular problemas de redes de flujo de costo mínimo.		
Mostrar ejemplo prototipo de toma de decisiones.		
Calcular problemas de toma de decisiones sin experimentación.		

2.5 Fase de diseño

“La arquitectura del software de un programa o sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos”. (Pressman, 2010) La fase de diseño se encarga de definir dichas estructuras.

2.5.1 Tarjetas CRC

Las tarjetas Clase-Responsabilidad-Colaboración (CRC) permiten ver las clases no como un depósito de datos, sino que permiten conocer el comportamiento de cada una en un alto nivel. La metodología XP estipula su uso como un artefacto obligatorio durante el desarrollo de un proyecto, debido a los beneficios que aportan a los desarrolladores. A continuación, se muestran las tarjetas de una de las clases que responden a funcionalidades de alta prioridad en el sistema. El resto de las tarjetas pueden visualizarse en los anexos seis y siete del presente documento.

Tabla 10 *tarjeta CRC de la clase Modelo.*

Nombre de la clase: Modelo	
Responsabilidades	Clases relacionadas

Esta clase es la encargada de crear un modelo:
def __init__(self, funcionObjetivo, matrizCoeficientes, ladoDerecho, cantidadVariables, cantidadRestricciones,signos):
Sus funcionalidades son:
def introducirCerosRest1(self, cantV, pos):
def introducirCerosRest2(self, cantV, pos):
def introducirCerosRest3(self, cantV, pos):
def despejarFuncionObjetivo(self):
def ampliarProblema(self):
def buscandoVariableEntrante(self, criterio):
def buscandoVariableSaliente(self, posVariableEntrante):
def calcularValorNuevo(self, valorViejo, valorFila, valorColumna,
	Problema
	M

pivote):
def calcularFuncionObjetivo(self):
def calcularValoresFuncionObjetivoIteracion(self, filaPivote, columnaPivote, posicionVariableEntrante, pivote):
def calcularLadoDerechoFO(self, valorFilaPivote, aux, pivote):
def iterar(self):	

La tarjeta CRC anterior muestra la clase Modelo, con todos sus atributos y responsabilidades en el lado izquierdo y las clases colaboradoras en el lado derecho, en este caso las clases Problema y M. Esta distribución como ya se ha explicado permite una mejor comprensión de las clases y su comportamiento. El resto de las tarjetas de la presente investigación se encuentran en los anexos.

2.6 Patrones de diseño del sistema

Para comenzar la implementación de un software no basta con definir la arquitectura, es necesario además establecer las directrices que permitan lograr un sistema bien estructurado, para así construir una solución eficaz. Estas directrices son llamadas Patrones de Diseño de Software. Para satisfacer las necesidades de la sociedad cada día se hace más necesario desarrollar un software de gran alcance y complejidad, en lo que son de gran utilidad los patrones de diseño empleados como mecanismos de reutilización.

Para la realización del sistema se emplearon algunos patrones **GRASP**[9], a continuación se muestra la selección y aplicación de los mismos:

- ✓ **Experto**

Empleando el patrón Experto se garantiza que cada clase del sistema asuma las responsabilidades que le conciernen, según las funcionalidades que se quieren implementar y a partir de la información que posee; por lo que cada clase contendrá la información necesaria para cumplir su responsabilidad. Como se muestra en la imagen, la clase **"M"**, es experta en la suma, resta, multiplicación, división, comparación y visualización de los objetos tipo M.

[9] **GRAPS:** Patrones generales para asignar responsabilidades o del inglés *General Responsibility Assignment Software Patterns*.

Figura 1*Patrón experto: Clase M*

- ✓ **Creador**

La creación de objetos es una actividad muy frecuente en los sistemas orientados a objetos, por lo tanto, es conveniente asignar esta responsabilidad de manera que potencie un bajo acoplamiento, una mayor claridad y una alta reutilización. La nueva instancia debe ser creada por la clase que tenga la información mínima necesaria para hacerlo. La clase **"Problema"**, es la encargada de crear instancias de la clase **"Modelo"**, como se muestra en la imagen.

```python
from metodos_solucion.Modelo import Modelo

__author__ = 'dayan'


class Problema:
    def __init__(self, nombre, cantVar, cantRest, criterio, funcionObjetivo, matrizCoeficientes, ladoDerecho, signos):
        self.nombre = nombre
        self.cantVar = cantVar
        self.cantRest = cantRest
        self.criterio = criterio
        self.funcionObjetivo = funcionObjetivo
        self.matrizCoeficientes = matrizCoeficientes
        self.ladoDerecho = ladoDerecho
        self.modelo = Modelo(funcionObjetivo, matrizCoeficientes, ladoDerecho, self.cantVar, self.cantRest, signos)
        self.signos = signos
```

Figura 2 *Patrón creador: Clase Problema.*

- ✓ **Bajo Acoplamiento**

Este patrón valora la importancia de hacer que las piezas sean intercambiables. Si dos trozos de código están "poco acoplados", entonces los cambios realizados a uno de los trozos tendrán poco o ningún efecto sobre el otro. Las URLconfs de Django son un buen ejemplo de este principio aplicado. En la aplicación, las definiciones de URL[10] y las funciones de vista a las que llaman están poco acopladas; esto es, la decisión de cómo tiene que ser una URL para una función dada y la propia implementación de la función residen en dos lugares separados. Esto permite cambiar una pieza sin afectar a la otra, como muestra la siguiente imagen, en la que, si se cambia la URL de /io/base/crearProblema a /io/base/crear/Problema, no representa un problema para la implementación subyacente de la función. De igual forma se puede cambiar la función de vista asignada a la URL, lo cual no incidirá en ninguna de las dos entidades.

```python
url(r'^io/programacionLineal/', PL, name='PL'),
url(r'^io/metodoSimplex/', simplex, name='simplex'),
url(r'^io/problemaPL/problema/', modeling.views.crearProblema, name='crearProblema'),
url(r'^io/problemaPL/problemaAmpliado/', modeling.views.ampliarProblema, name='ampliarProblema'),
url(r'^io/problemaPL/funcionObjetivoCalculada/', modeling.views.calcularFO, name='calcularFO'),
```

Figura 3 *Patrón bajo acoplamiento: URLconf de la solución propuesta.*

- ✓ **Alta cohesión**

[10] **URL:** *Uniform Resource Locator*, en español Localizador de Recursos Uniforme.

Django permite asignar responsabilidades con una alta cohesión ya que los controladores definen las acciones para las plantillas y colaboran con otras clases para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades. Las vistas controladoras contienen diferentes funcionalidades que se encuentran estrechamente relacionadas posibilitando que el software sea flexible frente a grandes cambios

También se utilizarán patrones GOF[11], los que a continuación se detallan, así como su uso en la solución que se presenta.

- ✓ **Fachada**

Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. Se evidencia en la creación de una plantilla para el cuerpo de las interfaces, como se evidencia en las siguientes imágenes:

```
{% extends 'base.html' %}
{% block content %}
    <br>
    <br>
    <div role="tabpanel" class="bs-example bs-example-tabs">
        <ul role="tablist" class="nav nav-tabs" id="myTab"...>
        <div class="tab-content" id="myTabContent">
            <div aria-labelledby="problema-tab" id="problema" class="tab-pane fade active in" role="tabpanel">
                <nav>
                    <ul class="pager">
                        <li class="next"><a id="sigPro">Siguiente <span aria-hidden="true">&rarr;</span></a></li>
                    </ul>
                </nav>
                {% block problema ... %}
            </div>
            <div aria-labelledby="variable-tab" id="variable" class="tab-pane fade" role="tabpanel">
                <nav>
                    <ul class="pager">
                        <li class="previous"><a id="antVar"><span aria-hidden="true">&larr;</span> Anterior</a></li>
                        <li class="next"><a id="sigVar">Siguiente <span aria-hidden="true">&rarr;</span></a></li>
                    </ul>
                </nav>
                {% block variable ... %}
            </div>
            <div aria-labelledby="funObj-tab" id="funObj" class="tab-pane fade" role="tabpanel">
                <nav>
                    <ul class="pager">
                        <li class="previous"><a id="antFO"><span aria-hidden="true">&larr;</span> Anterior</a></li>
                        <li class="next"><a id="sigFO">Siguiente <span aria-hidden="true">&rarr;</span></a></li>
                    </ul>
```

Figura 4 *Patrón fachada 1: Página ejemploBase.html, define la plantilla para los ejemplos de modelación.*

[11] GOF: *Gang of Four*, en español Banda de los cuatro.

```
{% extends 'modelacionHtml/ejemploBase.html' %}
{% block problema %}
    <p align="justify"...>
    <br>
    <p align="center"...>
    <table border="2" align="center" class="table table-hover"...>
    <br>
    <p align="center"...>
    <table border="2" align="center" class="table table-hover"...>
{% endblock %}

{% block variable %}
    <p...>
{% endblock %}

{% block funcion %}
    <p...>
{% endblock %}

{% block restricciones %}
    <p...>
{% endblock %}

{% block solucion %}
    <p...>
{% endblock %}
```

Figura 5 *Patrón fachada 2: Página ejemplo01.html, que hereda su estructura de la página ejemploBase.html.*

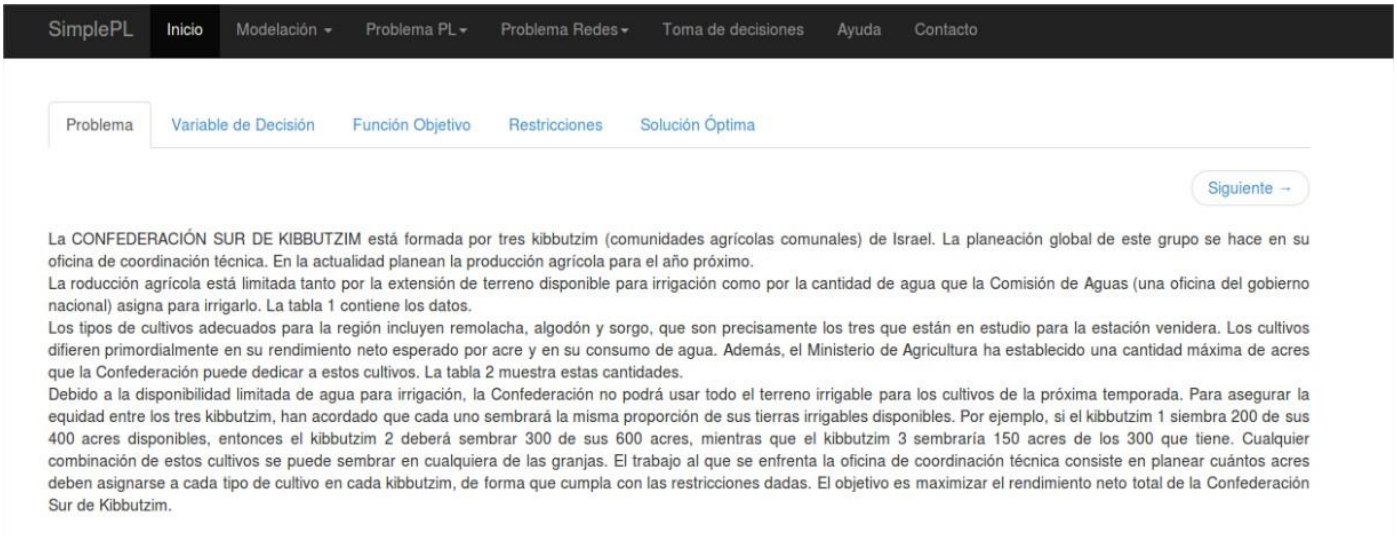


Figura 6 *Patrón fachada 3: Página ejemplo01.html.*

- ✓ **Iterador**

Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. El mismo se utiliza en el recorrido de la matriz de coeficientes de los problemas de PL, que pueden tener números tanto reales, como de tipo M, evidenciándose en la siguiente imagen:

```
109     # Objetivo: Despejar la funcion objetivo
110     # Ejecucion: Itera por la funcion objetivo, despejando la misma para igualarla a 0
111     # Return: Devuelve la funcion objetivo despejada
112     def despejarFuncionObjetivo(self):
113         funcionDespejada = []
114         for i in self.funcionObjetivo:
115             if not (isinstance(i, M)):
116                 funcionDespejada.append(i * (-1))
117             else:
118                 funcionDespejada.append(i)
119         self.funcionObjetivoModeloAmpliado = funcionDespejada
```

Figura 7 *Patrón iterador: Función despejarFuncionObjetivo(self):.*

2.7 Diseño arquitectónico

El diseño arquitectónico es un proceso creativo en el que se intenta establecer una organización del sistema que satisfaga los requerimientos funcionales del propio sistema. Debido a que es un proceso creativo, las actividades dentro del proceso difieren radicalmente dependiendo del tipo de sistema a desarrollar, el conocimiento y la experiencia del arquitecto del sistema y los requerimientos específicos del mismo. (Somerville, 2005)

2.7.1 Patrón arquitectónico

El patrón llamado MVT (*Model-View-Template*) en Django hace referencia a Modelo-Vista-Plantilla, es sencillo de entender, el modelo sigue siendo el modelo, en este caso, la vista no es una vista, sino que más bien es un controlador que se llama vista, y las plantillas son las vistas del Modelo-Vista-Controlador, es decir, los formularios van en las plantillas, hacen peticiones a las vistas, y las vistas obtienen datos de los modelos (Foundation, 2005-2016).

A continuación, se detalla en qué consisten las tres abstracciones del patrón (Mestras, 2013):

- **Modelo:** Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

- **Plantilla:** Muestra la información al usuario. Pueden existir múltiples plantillas del modelo. Cada plantilla tiene asociado un componente controlador.

- **Vista:** Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de los botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicios para el modelo o la plantilla.

2.7.2 Descomposición modular orientada a objetos

Un modelo arquitectónico orientado a objetos estructura el sistema en un conjunto de objetos débilmente acoplados con interfaces bien definidas. Los objetos realizan llamadas a los servicios ofrecidos por otros objetos. Está relacionada con las clases objetos, sus atributos y sus operaciones. Cuando se implementa, los objetos se crean a partir de estas clases y se usan algunos modelos de control para coordinar las operaciones de los objetos. (Somerville, 2005)

Entre las ventajas que reporta el uso de la descomposición modular orientada a objetos, está que debido a que los objetos están débilmente acoplados, la implementación de los objetos puede afectarse sin afectar a otros objetos. Los objetos son a menudo representaciones de entidades del mundo real por lo que la estructura del sistema es fácilmente comprensible. Debido a que las entidades del mundo real se usan en diferentes sistemas, los objetos pueden reutilizarse.

La siguiente figura ilustra la descomposición orientada a objetos de la actual investigación, donde el lenguaje de programación hace uso de las bibliotecas NumPy y SciPy, y a su vez el marco de trabajo Django utiliza Python. Este estilo permite además la incorporación de nuevos módulos en caso de ser necesarios.

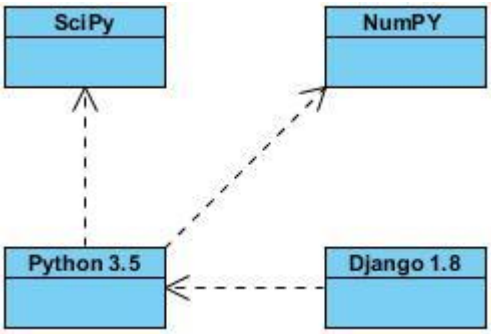


Figura 8 *Descomposición modular orientada a objetos de la presente solución.*

2.8 Conclusiones parciales del capítulo

En este capítulo se abordaron los aspectos fundamentales del análisis y diseño de la propuesta de solución. La descripción de la misma permitió crear una expresión en lenguaje natural que exprese en principio básico el funcionamiento que se desea para la propuesta de solución. La realización de las HU permitió determinar cuáles eran las funcionalidades básicas que deseaban los profesores del departamento de la asignatura IO.

Estas funcionalidades fueron traducidas a requisitos funcionales y en el proceso se obtuvieron además algunas especificaciones que fueron convertidas en requisitos no funcionales. La estimación del esfuerzo y el plan de iteraciones permitieron crear un cronograma que facilita la implementación teniendo en cuenta los compromisos con el cliente.

La creación de un plan de entrega permitió ajustar la realización del sistema a menos de 4 meses. La creación de las tarjetas CRC permitió la separación de las funcionalidades implementadas por el marco de trabajo y las que se necesitan implementar en el desarrollo.

Capítulo 3 Implementación y pruebas

3.1 Introducción

La concepción de la propuesta del sistema presentada en el capítulo anterior, ayuda al programador a entender mejor las funcionalidades que busca el cliente y, por tanto, ser capaz de llevarlas a código entendible por la computadora, o lo que es lo mismo implementar la herramienta. Durante toda la fase de implementación y después de esta se desarrollan un conjunto de pruebas con el objetivo de asegurar la calidad de la aplicación y que la misma cumpla con todas las peticiones del cliente. Como principales elementos en este capítulo se definen los estándares de codificación. Además de especificar los tipos de pruebas aplicados a la solución desarrollada para validar su correcto funcionamiento.

3.2 Implementación

La implementación es un proceso que comienza cuando se crea una aplicación en el equipo de un desarrollador y termina cuando está instalada y lista para ejecutarse en el equipo de un usuario. Describe cómo los elementos del modelo de diseño se implementan en términos de componentes, en otras palabras, toma el resultado del modelo de diseño para generar el código final del sistema. Dicho código está determinado por el lenguaje de programación y tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño.

3.2.1 Estándares de código

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. (Tristán, 2016)

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará la notación PascalCasing (cada palabra comienza con letra mayúscula). Por ejemplo (Modelo o SoluciónPasoAPaso).

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará la notación CamelCasing (es parecido al PascalCasing con la excepción que la letra inicial del identificador debe estar en minúscula), y con solo leerlo se sabe el propósito de la misma. Por ejemplo (ampliarProblema o buscarVariableEntrante).

Nomenclatura de las variables

El nombre de las variables se escribe con la primera letra en minúscula, si es un nombre compuesto se empleará la notación de CamelCasing descrita anteriormente. Por ejemplo (nombreProblema o cantRestricciones)

Nomenclatura de los comentarios

Dado que la metodología utilizada establece como principal artefacto generado el código, es de suma importancia que el mismo esté bien comentado, permitiendo así un mejor entendimiento y reutilización del mismo en caso de ser necesario. Los comentarios deben ser precisos de forma tal que se entienda el propósito de lo que se está desarrollando. Siempre se pondrán encima de la función a la cual hace alusión dicho comentario. Por ejemplo:

```
# Función para determinar la variable a introducir en la base de la solución

# Devuelve la posición de la variable no base, que pasa a ser base en la iteración actual

def buscarVariebleEntrante(self):

    ….

    return posVariableEntrante
```

3.3 Pruebas de software

“Las pruebas de software son el proceso que permiten verificar, garantizar y mostrar la calidad de un producto, a través de resultados registrables que proporcionan una evaluación y representa una revisión final de las especificaciones, del diseño y de la codificación. Son empleadas para identificar posibles fallos durante el proceso de desarrollo.” (Pressman, 2010)

Luego de generado el código fuente, es necesario realizarle pruebas para detectar y corregir la mayor cantidad de errores posibles con el objetivo de brindar una solución con la calidad requerida. Existen diferentes tipos de pruebas, las cuales tienen como objetivo detectar alguna anomalía en los resultados. En su mayoría están orientadas a comprobar las funcionalidades y lógica del negocio, verificando que el sistema se comporte a la altura de las especificaciones que pide el cliente.

Con el objetivo de validar el correcto funcionamiento de acuerdo a los requisitos, descritos por el cliente en las HU, se realizan pruebas de aceptación. Además, para validar el correcto funcionamiento de las interfaces gráficas de usuarios, se aplicaron pruebas de caja negra utilizando el método de partición equivalente.

3.3.1 Pruebas de aceptación

El uso de cualquier productor de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlos es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada “prueba de aceptación”. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto de justifique(Pressman, 2010).

Para eliminar la influencia de conflictos de intereses, y para que sea lo más objetiva posible, la prueba de aceptación nunca debería ser responsabilidad de los ingenieros de software que han desarrollado el producto.

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente.

Este tipo de pruebas genera como artefacto los Casos de Pruebas (CP). “Los casos de pruebas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, permitiendo encontrar y documentar los defectos que puedan afectar la calidad del software”. (Pressman, 2010)

A continuación, se muestra uno de los CP correspondientes a las pruebas de aceptación realizadas a la aplicación, el resto se encuentra en los anexos 8 y 9 del presente documento:

Tabla 11 *Caso de Prueba de Aceptación para la HU "Crear Problema"*

Caso de Prueba de Aceptación	
Código: R1_HU2	**Historia de Usuario:** 2
Nombre: Crear Problema	
Descripción: El sistema debe permitir al usuario la introducción de los datos necesarios para crear el problema.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: 1. Clic en la opción "**Problema PL**" del menú ubicado en la parte superior del sistema. 2. Clic en la opción "**Nuevo**" del menú desplegable. 3. Introducir los datos en los campos (Nombre, Cantidad de Variables, Cantidad de Restricciones). Seleccionar el criterio de optimización en el campo (Criterio de optimización. Juego de datos (**"Confección de pantalones", "2", "3"**, seleccionar valor). 4. Clic en el botón "**Crear Problema**".	
Resultado esperado: Se redirecciona al usuario a la página correspondiente del problema recién creado.	
Resultado obtenido: Nueva página mostrando el formulario para la introducción de los coeficientes del problema.	
Evaluación de la prueba: Satisfactoria.	

3.3.2 Resultado de las pruebas de aceptación

Para la validación de los requisitos funcionales se realizaron tres iteraciones y una prueba de Regresión entre cada iteración. En la figura 9 se muestran los resultados obtenidos en cada una de las iteraciones de pruebas realizadas a la herramienta SimplePL. Como se puede apreciar en la primera iteración fueron detectadas 29 no conformidades (NC) de las cuales fueron resueltas 23 y quedaron pendientes seis. Al aplicar la prueba de regresión correspondiente se subsanaron las mismas. Luego en la segunda iteración se detectaron once NC, de las cuales fueron resueltas nueve y quedaron dos pendientes. Las NC pendientes fueron subsanadas al aplicar la prueba de regresión correspondiente. En la tercera iteración se

detectaron ocho NC, de las cuales las ocho fueron resueltas. Al aplicar la prueba de regresión correspondiente no se detectaron nuevas NC. Entre las no conformidades detectadas durante el proceso de pruebas se destacan las siguientes:

- ✓ Los mensajes y botones presentan problemas ortográficos y de idioma.
- ✓ Acciones de crear sin validación de los datos.
- ✓ Errores en los formatos de las páginas.
- ✓ Visualización numérica de los resultados.

La relación de las NC por cada iteración de pruebas se puede apreciar en la siguiente tabla:

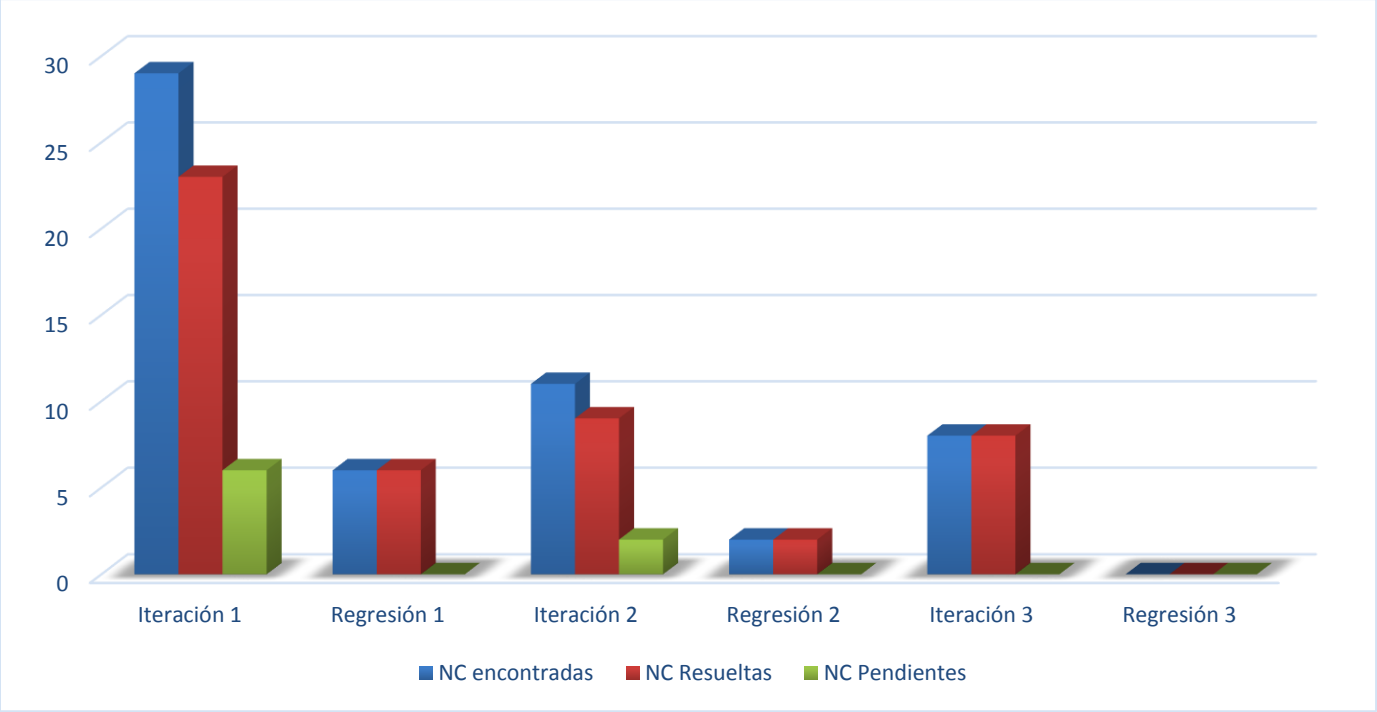


Figura 9 *Iteraciones de pruebas*

Terminada estas pruebas y corregidas las no conformidades, el cliente avaló la solución con la entrega del Certificado de Aceptación del Producto (ver Anexo 12) por parte del Ingeniero Yorgüy Antonio Batista Desdin, profesor del Departamento de Ciencias Básicas de la Facultad 3.

3.3.3 Pruebas de caja negra

Las pruebas de caja negra hacen referencia a pruebas que se llevan a cabo sobre la interfaz del software sin tener en cuenta el código de la aplicación. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Según (Pressman, 2010) estas pruebas tienen como propósito detectar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y de terminación

Las pruebas de caja negra comprenden un conjunto de diferentes técnicas como, por ejemplo: Partición de Equivalencia, Análisis de Valores Límites y Grafos de Causa-Efecto. De estas técnicas se selecciona para ser utilizada la Partición de Equivalencia, la cual divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Para la aplicación de la misma se realizaron 23 diseños de casos de prueba (DCP) uno por cada requisito funcional definido, los cuales se basan en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (Pressman, 2010)

Estas pruebas fueron ejecutadas por la Ing. Anabel Fé León Mendoza del Grupo de Calidad del Centro de Informatización de Entidades (CEIGE). Como resultado de la aplicación de las pruebas fueron identificadas un total de tres NC, las cuales se corrigieron en una única iteración. El acta de liberación emitida por dicho grupo de calidad se encuentra adjunta en el Anexo 11.

3.4 Conclusiones Parciales del capítulo

Este capítulo abordó las fases finales del proceso de desarrollo establecido por la metodología. En la fase de desarrollo se estableció el estándar de codificación a utilizar lo que permitió proyectar la calidad del mejor artefacto que genera la metodología, el código. En la fase de pruebas se examinó que la implementación realizada se ajustaba a lo especificado en las HU. De esta manera se concluyó el proceso de desarrollo y se determinó que el producto generado cumple con todas las exigencias del cliente.

Conclusiones

- El análisis de las herramientas utilizadas para el apoyo a la docencia de la asignatura Investigación de Operaciones, arrojó que no era factible utilizar ninguna de estas soluciones debido a que no se ajustan a las características requeridas en el Departamento de Ciencias Básicas de la Facultad 3 de la UCI, y por lo tanto confirmó la necesidad de desarrollar la nueva propuesta.
- El estudio de las herramientas, tecnologías y metodologías permitió determinar cuáles se ajustaban a las características del equipo de desarrollo y producto a realizar. Utilizando Pyhton, Django y PyCharm para el desarrollo del producto de esta investigación, regida por las políticas de desarrollo de la suite UNIMATH.
- El diseño permitió obtener una vista de la estructura estática del sistema teniendo en cuenta distintos patrones de diseño como el bajo acoplamiento, experto, creador, la alta cohesión, fachada e iterador. También se contribuyó a aumentar la escalabilidad del producto mediante el uso de estilos arquitectónicos como el MVP, lo cual propició la alta reutilización de las clases diseñadas.
- El proceso de implementación permitió desarrollar el producto a partir del diseño elaborado, cumpliendo con todos los requisitos identificados por los profesores del Departamento de Ciencias Básicas de la Facultad 3 de la UCI. El uso del estándar de codificación empleado contribuyó a la legibilidad del código, lo cual permitirá proveer una guía para futuros mantenimientos de la solución propuesta.
- El proceso de pruebas permitió garantizar un mínimo de calidad aceptable del producto desarrollado, lo cual se tomará como validación del cumplimiento de los requisitos en el módulo implementado. Se comprobó que cumplía con todas las especificaciones del cliente.

Glosario de términos

Criterio de la máxima posibilidad: En los problemas de toma de decisiones sin experimentación, identifique el estado más probable de la naturaleza (aquel que tiene la probabilidad a priori más grande). Para este estado de la naturaleza, encuentre la opción con el máximo pago. Elija esta alternativa de decisión.

Criterio del pago máximo: En los problemas de toma de decisiones sin experimentación, para cada opción posible, encuentre el pago mínimo sobre todos los estados posibles de la naturaleza. Después, encuentre el máximo de estos pagos mínimos. Elija la opción cuyo pago mínimo corresponde a este máximo.

Flujo de costo mínimo: El problema de flujo de costo mínimo tiene una posición medular entre los problemas de optimización de redes; primero, abarca una clase amplia de aplicaciones y segundo, su solución es muy eficiente. Igual que el problema del flujo máximo, toma en cuenta un flujo en una red con capacidades limitadas en sus arcos. Igual que el problema de la ruta más corta, considera un costo (o distancia) para el flujo a través de un arco. Igual que el problema de transporte o el de asignación, puede manejar varios orígenes (nodos fuente) y varios destinos (nodos demandas) para el flujo, de nuevo con costos asociados. De hecho, estos cuatro problemas son casos especiales del problema de flujo de costo mínimo.

Flujo máximo: Se trata de enlazar un nodo fuente y un nodo destino a través de una red de arcos dirigidos. Cada arco tiene una capacidad máxima de flujo admisible. El objetivo es el de obtener la máxima capacidad de flujo entre la fuente y el destino.

Hipermediales: Se entiende por hipertexto al sistema de presentación de textos extensos con o sin imágenes donde se puede adicionar sonido, formando una red con nodos que son unidades de información, con enlaces y arcos dirigidos hacia otros nodos, la red no es más que un grafo orientado, que se aparta de la forma secuencial tradicional del libro. Multimedia es la presentación de la información con grandes volúmenes de texto, con imágenes fijas, dibujos con animación y vídeo digital. Por lo tanto, la hipermedia es la combinación de hipertexto y multimedia (Nielsen, 1995).

Biblioteca: Conjuntos de código ya realizado que se puede reutilizar en los programas y que ahorran mucho esfuerzo en la programación.

Regla de decisión de Bayes: En los problemas de toma de decisiones sin experimentación, se utilizan las mejores estimaciones disponibles de las probabilidades de los respectivos estados de la naturaleza —en este momento las probabilidades a priori—, para calcular el valor esperado del pago de cada opción posible. Se elige la opción con el máximo pago esperado.

Trayectoria de aumento: Es una trayectoria dirigida del nodo fuente al nodo destino en la red residual, tal que todos los arcos en esa trayectoria tienen capacidad residual estrictamente positiva. El mínimo de estas capacidades residuales se llama capacidad residual de la trayectoria de aumento porque representa la cantidad de flujo que es factible agregar en toda la trayectoria. Por lo tanto, cada trayectoria de aumento proporciona una oportunidad de aumento más el flujo a través de la red original.

Wine: es una reimplementación libre de la API de Windows (Win16 y Win32), es decir, un proyecto que permite ejecutar programas diseñados para Windows bajo sistemas operativos de la familia Unix, como GNU/Linux.

Bibliografía

Albaladejo, Xavier. 2010. ¿Qué es SCRUM? *Proyectos Ágiles.* [En línea] 2010. [Citado el: 23 de Enero de 2016.] https://proyectosagiles.org/que-es-scrum/.

Beck, Kent. 1999. *Extreme Programming Explained. Embrace Change.* [trad.] Addison-Wesley. s.l. : Pearson Education, 1999.

Brito, Julio César. 2012. *Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v 2.0.* La Habana : Ediciones Futuro, 2012.

Cataldi, Zulma. 2000. *Una metodología para el diseño, desarrollo y evaluación de software educativo.* 2000. pág. 130.

Cohn, M. 2004. *User Stories Applied.* 2004. ISBN 0-321-20568-5.

developers, Numpy. 2016. Numpy - Numpy. *Numpy.* [En línea] 2016. [Citado el: 11 de Marzo de 2016.] http://www.numpy.org/.

developers, SciPy. 2016. Scientific Computer tools for Pyhton. *SciPy.org.* [En línea] 2016. [Citado el: 11 de Marzo de 2016.] https://www.scipy.org/about.html.

Eguíluz, J. 2009. Libros Web. *Introducción a CSS.* [En línea] 2009. [Citado el: 15 de Noviembre de 2015.] http://librosweb.es/css/index.html..

Foundation, Django Software. 2005-2016. Django The web framework for perfectionists with deadlines. *Django The web framework for perfectionists with deadlines.* [En línea] Django Software Foundation, 2005-2016. [Citado el: 10 de Diciembre de 2015.] https://www.djangoproject.com/.

Foundation, Python Software. 1990-2016. General Python FAQ. *Python.* [En línea] 1990-2016. [Citado el: 25 de Febrero de 2016.] https://docs.python.org/2/faq/general.html#what-is-python.

Fowler, M, Beck, K y Brant, J. 1999. *Refactoring: Improving the Design of Existing Code.* s.l. : Addison-Wesley, 1999.

Graterol, Yohan. 2014. PyCharm: El mejor IDE para tus proyectos en Python. *CristaLab.* [En línea] 16 de Septiembre de 2014. [Citado el: 15 de Enero de 2016.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084l/.

Gutiérrez, J. J. 2011. Qué es un framework web? [En línea] 2011. [Citado el: 16 de Noviembre de 2015.] www.lsi.us.es/javierj/investigacion_ficheros/Framework.pdf..

Marquès, Pere. 1995. El software educativo. *Biblioteca virtual de tecnología educativa.* [En línea] 1995. [Citado el: 13 de 2 de 2016.] www.lmi.ub.es/te/any96/marques_software/.

Mestras, P.J. 2013. *Estructuras de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC).* [En línea] 2013. [Citado el: 10 de Febrero de 2016.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf ..

Miyashiro, Liodine y Batista Desdin, Yorgüy Antonio. 2015. *Informe semestral de la asignatura Investigación de Operaciones, curso 20142015.* Habana : s.n., 2015.

Mora, S. L. 2002. RUA. *Programación de aplicaciones web: historia, principios básicos y clientes web.* [En línea] 2002. [Citado el: 15 de Noviembre de 2015.] ttp://rua.ua.es/dspace/handle/10045/16995..

Orjuela Duarte, MSc. Ailin y Rojas C., MSc. Mauricio . 2008. *Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo.* 2008.

Pressman, Roger. 2010. *Ingeniería de software. Un enfoque práctico.* España : Mcgraw – Hill. Interamericana de España, 2010. ISBN 970 – 10 – 5473 – 3.

Rossum, Guido van. 2009. *El tutorial de Python.* [ed.] Jr Fred L. Drake. 2009.

S. Hillier, Frederick y J. Lieberman, Gerald. 1997. *Introducción a la investigación de operaciones.* La Habana : Editorial Félix Varela, 1997. Vol. 1.

Sánchez, A. F. 2013. ¿Que es Bootstrap? [En línea] 2013. [Citado el: 16 de Noviembre de 2015.] http://openwebcms.es/2013/que-es-bootstrap/..

Somerville, Ian. 2005. 11.1 Decisiones del diseño arquitectónico. *Ingeniería del Software.* 7. Madrid : s.n., 2005, 11.

Tristán, Fernando Becerra. 2016. Estándares de Codificación .Net v.1.0.0.0. [En línea] 2016. [Citado el: 5 de Mayo de 2016.] serk.kualtus.com/codigo.htm.

Zakas, N. C. 2009. Professional JavaScript for Web Developers. [aut. libro] N. C ZAKAS. *Professional JavaScript for Web Developers.* s.l. : Indianápolis, D. E. Wiley Publishing, 2009.

Anexos

Anexo 1 *Historia de Usuario: Mostrar ejemplo de modelación.*

<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número 1</td><td>Nombre de la Historia de Usuario: Mostrar ejemplo de modelación.</td></tr>
<tr><td colspan="2">Cantidad de modificaciones a la Historia de Usuario: Ninguna</td></tr>
<tr><td>Usuario: Estándar</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad en negocio: Baja</td><td>Riesgo en desarrollo: Bajo</td></tr>
<tr><td colspan="2">Descripción: El sistema debe permitir al usuario la visualización de problemas prototipos de la modelación de problemas de PL y redes. Para ello el actor debe dar clic en el menú en la opción “Modelación”, al hacerlo se desplegará un submenú listando los diferentes tipos de programación que existe (Lineal, Redes) y por cada uno de estos los tipos específicos (Lineal – Entera, Mixta, Bivalente, Redes –Transporte, Asignación). El estudiante o profesor elige la opción deseada y el sistema lo enviará a la página correspondiente. En la misma el estudiante o profesor puede instruirse sobre el tipo específico de problema que eligió y acceder a varios ejemplos prototipos presentes en dicha página.</td></tr>
<tr><td colspan="2">Prototipo:
 
</td></tr>
</table>

Anexo 2 *Historia de Usuario: Historia de usuario- Insertar problema de Redes.*

<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número 3</td><td>Nombre de la Historia de Usuario: Insertar problema de Redes.</td></tr>
<tr><td colspan="2">Cantidad de modificaciones a la Historia de Usuario: Ninguna.</td></tr>
<tr><td>Usuario: Estándar.</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad en negocio: Alta.</td><td>Riesgo en desarrollo: Bajo.</td></tr>
<tr><td colspan="2">Descripción: El sistema debe permitir al estudiante o profesor la inserción de los datos necesarios para la modelación del problema de redes. Para ello el estudiante o profesor involucrado debe de llenar el formulario que el sistema le provee. Dicho formulario cuenta con los siguientes campos: “Nombre del Problema”, “Cantidad de Nodos Origen”, “Cantidad de Nodos Destino”, “Criterio de la Función Objetivo” y “Variable de Decisión”. Donde el campo “Nombre del Problema” se refiere al nombre del problema que se pretende crear, este campo admite valores alfanuméricos (text, 255), el campo “Cantidad de Nodos Origen” hace referencia a la cantidad de nodos origen que presenta el problema, este campo admite sólo valores numéricos (number, max=20), el campo “Cantidad de Nodos Destino” hace referencia a la cantidad de nodos destino que presenta el problema, este campo admite sólo valores numéricos (number, max=20), el campo “Criterio de la Función Objetivo” hace referencia al fin que está destinado el problema, puede tomar solamente dos valores (Maximizar, Minimizar) y el campo “Variable de Decisión” se refiere al tipo de variable que se estará utilizando en el problema, este campo puede tomar uno de los cuatro valores presentes en el formulario (Continuos positivos, Enteros positivos, Binario, No acotado) . Una vez llenados dichos campos el estudiante o profesor debe de hacer clic en el botón “Crear Problema” y automáticamente el sistema envía al estudiante o profesor a la página para introducir los valores necesarios para el procesamiento del problema. En caso de presentar algún error en el contenido introducido por el estudiante o profesor en el formulario, se muestra un mensaje de error en el campo donde se cometió el mismo y se mantiene en el formulario actual. También, si el estudiante o profesor determina que los datos no están correctos y desea borrarlos para introducir nuevos, el botón “Limpiar Datos” le facilita esta acción.</td></tr>
</table>

Prototipo:

Anexo 3 *Historia de Usuario: Calcular problema de redes de flujo máximo.*

Historia de Usuario	
Número 4	**Nombre de la Historia de Usuario:** Calcular problema de redes de flujo máximo.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Estándar.	**Iteración asignada:** 2
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Bajo.
Descripción: El sistema debe permitirle al estudiante o profesor una vez hayan sido introducidos los datos calcular la solución para el problema en cuestión aplicando el método de redes de flujo máximo. Para ello luego de creado el problema e introducidos los coeficientes, el estudiante o profesor podrá dar clic en el botón **"Solución"** y será enviado a la página que contiene la solución del problema. También, si el estudiante o profesor determina que los datos no están correctos y desea borrarlos para introducir nuevos, el botón **"Limpiar Datos"** le facilita esta acción.	

Prototipo: -

Anexo 4 *Historia de Usuario: Calcular problema de redes de flujo de costo mínimo.*

Historia de Usuario	
Número 5	**Nombre de la Historia de Usuario:** Calcular problema de redes de flujo de costo mínimo.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Estándar.	**Iteración asignada:** 2
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Bajo.
Descripción: El sistema debe permitirle al estudiante o profesor una vez hayan sido introducidos los datos calcular la solución para el problema en cuestión aplicando el método de redes de flujo de costo mínimo. Para ello luego de creado el problema e introducidos los coeficientes, el estudiante o profesor podrá dar clic en el botón **"Solución"** y será enviado a la página que contiene la solución del problema. También, si el estudiante o profesor determina que los datos no están correctos y desea borrarlos para introducir nuevos, el botón **"Limpiar Datos"** le facilita esta acción.	
Prototipo: -	

Anexo 5 *Historia de Usuario: Mostrar ejemplo prototipo de toma de decisiones.*

Historia de Usuario	
Número 6	**Nombre de la Historia de Usuario:** Mostrar ejemplo prototipo de toma de decisiones.
Cantidad de modificaciones a la Historia de Usuario: Ninguna.	
Usuario: Estándar.	**Iteración asignada:** 2

Prioridad en negocio: Baja.	**Riesgo en desarrollo:** Bajo.
Descripción: El sistema debe permitir al usuario la visualización de problemas prototipos del apoyo a la toma de decisiones. Para ello el actor debe dar clic en el menú en la opción "**Toma de decisiones"**, al hacerlo será re direccionado a una página que contendrá una breve explicación sobre el objetivo del apoyo a la toma de decisiones y varios ejemplos prototipos de la puesta en práctica de la misma.	
Prototipo: -	

Anexo 6 *Tarjeta CRC de la clase M.*

Nombre de la clase: M	
Responsabilidades	Clases relacionadas

Esta clase es la encargada de crear objetos de tipo M:
def __init__(self, real, m):
Sus funcionalidades son:
def real(self):
def m(self):
def sumar(self, x):
def restar(self, x):
def multiplicar(self, x):
def dividir(self, x):
def comparar(self, x):
def __str__(self):
	Modelo
	Problema

Anexo 7 *Tarjeta CRC de la Clase Problema.*

Nombre de la clase: Problema	
Responsabilidades	Clases relacionadas

Esta clase es la encargada de crear objetos de tipo Problema:
def __init__(self, nombre, cantVar, cantRest, criterio, funcionObjetivo, matrizCoeficientes, ladoDerecho, signos):
Sus funcionalidades son:
def generarModeloAmpliado(self):
def modelo(self):
	M
	Modelo

Anexo 8 *Caso de prueba: Ampliar problema.*

Caso de Prueba de Aceptación	
Código: R2_HU2	**Historia de Usuario:** 2
Nombre: Ampliar problema	
Descripción: El sistema debe permitir al usuario la introducción de los coeficientes en el formulario y posteriormente ampliar el mismo.	
Condiciones de Ejecución: Debe haber sido creado el problema anteriormente.	
Entrada/Pasos de ejecución: 1. Introducir los datos en el formulario. Juego de datos (3, 5; 1, 0, 4; 0, 2, 12; 3, 2, 18) 2. Clic en la opción **"Resolver".**	
Resultado esperado: Se redirecciona al usuario a la página correspondiente del problema ampliado.	
Resultado obtenido: Nueva página mostrando la tabla con los datos pertenecientes al modelo ampliado.	
Evaluación de la prueba: Satisfactoria.	

Anexo 9 *Caso de prueba: Crear problema de transporte.*

Caso de Prueba de Aceptación	
Código: R1_HU2	**Historia de Usuario:** 2
Nombre: Crear problema de transporte	
Descripción: El sistema debe permitir al usuario la introducción de los datos necesarios para crear el problema de transporte.	
Condiciones de Ejecución:	
Entrada/Pasos de ejecución: 1. Clic en la opción "**Problema Redes**" del menú ubicado en la parte superior del sistema. 2. Clic en la opción "**Nuevo problema de transporte**" del menú desplegable. 3. Introducir los datos en los campos (Nombre, Cantidad de nodos origen, Cantidad de nodos destino). Seleccionar el criterio de optimización en el campo (Criterio de optimización). Juego de datos (**"Confección de pantalones", "2", "3"**, seleccionar valor). 4. Clic en el botón "**Crear Problema**".	
Resultado esperado: Se redirecciona al usuario a la página correspondiente del problema recién creado.	
Resultado obtenido: Nueva página mostrando el formulario para la introducción de los coeficientes del problema.	
Evaluación de la prueba: Satisfactoria.	

Anexo 10 *Arquitectura del Departamento de Ciencias Básicas de la Facultad 3.*

UNIMATH; SUITE DE HERRAMIENTAS DE APOYO A LA DOCENCIA DEL DEPARTAMENTO DE CIENCIAS BÁSICAS FACULTAD 3

El universo de herramientas y tecnologías que persiguen como objetivo apoyar los procesos de análisis matemáticos, análisis estadísticos, solución de modelos matemáticos (lineales y no lineales), inferencia estadística y otros procedimientos matemáticos es inmenso. El número de soluciones y su heterogeneidad es tan diverso como las necesidades que las generan, principalmente provenientes del mundo académico y del mundo empresarial. La implementación de soluciones informáticas que realicen estas operaciones debido a diversos factores no comparte un formato o estándar y en no pocas ocasiones no mantienen ninguna relación y sus desarrolladores implementan las mismas funcionalidades o los mismos

procedimientos de manera distinta cada vez. Esta separación ha provocado no solo el incremento del número de soluciones sino además los gastos en tiempo y dinero para los entes interesados en obtener dichas herramientas.

La utilización de asistentes matemáticos para realizar algunas de estas operaciones es una actividad extendida. Dentro de los asistentes más utilizados se pueden mencionar MATLAB, Octave, R, Mathematica, Derive, MAPLE, Excel entre otros. No todas estas herramientas son libres por lo que están limitadas a su uso o modificaciones bajo licencia. Además, condicionan su uso al conocimiento previo de los lenguajes particulares de cada uno de ellos.

La utilización de una arquitectura de desarrollo única para las matemáticas crearía homogeneidad en las soluciones, la definición de un lenguaje de programación estandarizaría todos los códigos y permitiría la reutilización de los mismos. La utilización de Python fue prácticamente unánime en la discusión con los miembros del departamento teniendo en cuenta las facilidades que brinda este lenguaje para el trabajo de las matemáticas. Python cuenta con bibliotecas especializadas como son:

- Pyomo: Trabajo con optimización de funciones
- NumPy: Arreglos optimizados para el trabajo numérico.
- SciPy: Herramientas y algoritmos matemáticos.
- Matplotlib: Generación de gráficos a partir de datos contenidos en listas.

Además, incluye por defecto la biblioteca Math que permite trabajar con funciones trigonométricas, creación de sucesiones aleatorias, redondeos, evaluación de funciones, entre otras opciones. El departamento de Ciencias Básicas de la Facultad 3 propone la siguiente línea tecnológica para el desarrollo de las herramientas de apoyo a la docencia:

1. Python 3.5
2. Django 1.8.0
3. Bootstrap 3.0

Esta línea tecnológica permite además incluir todas las facilidades de la web al desarrollo. Se escoge trabajar sobre la web con una tecnología que permita la ejecución de la herramienta en distintas estaciones de trabajo o en un servidor al cual tengan acceso los usuarios. UNIMATH contendría todas las soluciones creadas por y para el departamento, facilitando el proceso de enseñanza aprendizaje y los trabajos metodológicos en los colectivos de las asignaturas pertenecientes al departamento.

Elizabeth Rodríguez Stiven
Jefa de departamento

Yorgüy Antonio Batista Desdin
Líder del proyecto UNIMATH

Anexo 11 *Carta de Liberación de Calidad del Centro de Calidad del CEIGE.*

Universidad de las Ciencias Informáticas

CENTRO DE INFORMATIZACIÓN DE ENTIDADES

Validación y Verificación de Requisitos (V&V)

ACTA DE LIBERACIÓN DEL SISTEMA SIMPLEPL

Acta de Liberación, Versión 1.0, 28/05/2016

Índice de contenidos

Control del documento

Título: Herramienta de apoyo a la docencia de la asignatura Investigación de Operaciones, SimplePL.

Versión: 1.0

	Nombre	Cargo	
Elaborado por	Ing. Anabel Fé León Mendoza	RGA	
Revisado por	Ing. Yisel Niño Benitez	Asesor de Calidad	
Aprobado por	MsC. Maybel Díaz Capote	**Firma**	
Cargo	Subdirector I+D+i	**Fecha**	28/06/2016

Reglas de confidencialidad

Clasificación: Uso Interno

Forma de distribución: PDF Digital

Este documento contiene información propietaria del **CENTRO DE INFORMATIZACIÓN DE ENTIDADES,** y es emitido confidencialmente para un propósito específico.

El que recibe el documento asume la custodia y control, comprometiéndose a no reproducir, divulgar, difundir o de cualquier manera hacer de conocimientos público su contenido, excepto para cumplir el propósito para el cual se ha generado.

Las reglas son aplicables a las 7 páginas de este documento.

Control de cambios

Versión	Lugar*	Tipo**	Fecha	Autor	Descripción
Versión 1.0	Todo el documento	A	28/06/2016	Ing. Anabel Fé León Mendoza	Creación

1 Introducción

1.1 Objetivo

A través de este documento se emite el acta de liberación de la Herramienta de apoyo a la docencia de la asignatura Investigación de Operaciones, SimplePL.

1.2 Alcance

Está destinado a todos los miembros del Tribunal de Tesis.

1.3 Definiciones y acrónimos

Plantilla: Documento de ejemplo que sustenta un formato y que describe los lineamientos para la elaboración de documentos similares.

1.4 Referencias

IEEE. 1991. *IEEE Standard Glossary of Software Engineering Terminology*. Spring 1991 Edition. 1991. IEEE Standard 610.12–1990.

2 Datos del producto

Emitida a favor de: Herramienta de apoyo a la docencia de la asignatura Investigación de Operaciones, SimplePL.

Fecha de emisión del acta: 28/06/2016

Responsable: Ing. Anabel Fé León Mendoza

Cargo: RGA

2.1 Clasificado como:

- Aplicación

2.2 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidades

2.3 Cantidad de iteraciones:

Para la revisión se emplearon un total de 2 iteraciones de trabajo para lograr el resultado de 0 (cero) No Conformidad.

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
Aplicación	Versión 1.0	No Conformidades	2 iteraciones total	Pruebas Funcionales	28/06/2016

3 Elementos revisados o probados y herramientas utilizadas

Elemento	Herramienta
Aplicación	Diseño de Caso de Pruebas

3.1 Cantidad total de horas empleadas y rango de fechas:

Se emplearon un total de 4 horas efectivas de trabajo los días 23/06/2016 y 28/06/2016

3.2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 2 turnos de trabajo, con 1 probador, toda la actividad estuvo dirigida por un Jefe de Pruebas.

4 Evaluado por:

4.1 Especialista principal Asignado:

Ing. Yisel Niño Benítez

4.2 Otro personal especializado participante:

Nombres y Apellidos	Cargo
Ing. Anabel Fé León Mendoza	RGA

Ing. Yisel Niño Benítez
Asesor de Calidad

Ing. Anabel Fé León Mendoza
RGA

Anexo 12 *Certificado de Aceptación del Producto*

27 de junio de 2016
"Año 58 de la Revolución"

A quien pueda interesar:

En el departamento de Ciencias Básicas de la Facultad 3 de la UCI, se está apostando desde el curso escolar 2014-2015 por el diseño e implementación de un conjunto de soluciones bajo la concepción de un mismo marco de trabajo como apoyo a las asignaturas asociadas a dicho departamento. En un principio se comenzó con MATHNUM®, potente instrumento matemático que facilita la representación numérica y geométrica de las soluciones aproximadas según los métodos estudiados en la asignatura Matemática IV. La idea de incorporar herramientas de apoyo a dichas asignaturas viene dada principalmente por la poca motivación que sienten los estudiantes de la carrera por el estudio de las Ciencias Básicas. Además, la introducción de herramientas al proceso de enseñanza aprendizaje provee

de nuevos mecanismos para que los estudiantes se nutran de los conocimientos necesarios y básicos de cada disciplina.

La solución que se ofrece en el trabajo de diploma titulado: **Herramienta de apoyo a la enseñanza de la Investigación de Operaciones "Simple PL"** del autor: Dayan Hernández Ramos constituye un aporte de gran valor teórico-práctico para aplicar en las clases de la asignatura a la que corresponde y apoyar el proceso de enseñanza aprendizaje de la misma. Con el uso de esta herramienta se disminuye considerablemente el tiempo dedicado en las clases a la realización de operaciones matemáticas, posibilitando proceder al debate y al análisis de las soluciones a los problemas de programación lineal. La herramienta permite aplicar el método Simplex a problemas de programación lineal incorporando la creación de modelos ampliados no presente en las herramientas existentes hasta el momento. La herramienta presenta un componen informativo con ejemplos de problemas tipos de los 4 temas de la asignatura.

El mayor aporte consiste en que el investigador ofrece una solución que de incorporarse a las clases de la asignatura Investigación de Operaciones (IO) permitirá reforzar varios de los actuales procesos docentes que en ella se realizan. SimplePL permite la presentación de los resultados de los métodos que se utilizan en la asignatura, favoreciendo a la comprensión de los mismos.

Lo anterior permite validar la herramienta SimplePL como una herramienta, que cumple con las necesidades del departamento y puede utilizarse como apoyo a la asignatura IO.

_____	_____
MSc. Elizabeth Rodríguez Stiven	Ing. Yorgüy Antonio Batista Desdin
Profesora Auxiliar	Profesor de la asignatura IO
Jefe de Departamento de Ciencias Básicas	Facultad 3
Facultad 3	