SISNOVA - SISTEMA DE ELEMENTOS DE SOFTWARE REUTILIZABLES, COMO APOYO A LA CONSTRUCCIÓN DE APLICACIONES ESCALABLES Y ROBUSTAS QUE PERMITAN LA RESOLUCIÓN DE PROBLEMAS DE TRANSPORTES

Universidad de Nariño

HÉCTOR ANDRÉS MORA PAZ
DALILA MERCEDES PACHAJOA PACHAJOA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2013

SISNOVA - SISTEMA DE ELEMENTOS DE SOFTWARE REUTILIZABLES, COMO APOYO A LA CONSTRUCCIÓN DE APLICACIONES ESCALABLES Y ROBUSTAS QUE PERMITAN LA RESOLUCIÓN DE PROBLEMAS DE TRANSPORTES

HÉCTOR ANDRÉS MORA PAZ
DALILA MERCEDES PACHAJOA PACHAJOA

Trabajo de grado presentado como requisito parcial para optar el título de Ingenieros de Sistemas

M. Sc. LUIS VICENTE CHAMORRO MARCILLO
Director

M. Sc. OSCAR REVELO SÁNCHEZ
Codirector

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
SAN JUAN DE PASTO
2013

NOTA DE RESPONSABILIDAD

"Las ideas y conclusiones aportadas en la tesis de grado, son responsabilidad exclusiva de la autora".

Artículo 1^0 del Acuerdo N^0 324 de Octubre 11 de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

_____
Firma del Director de proyecto

San Juan de Pasto, diciembre de 2013

Con toda gratitud a todas las personas que siempre creyeron en nuestra capacidad, es grato saber la fuerza y determinación que poseemos cuando queremos alcanzar algo.

A Dios por todo su amor y su fortaleza, porque está presente en cada paso que damos y es nuestro guía en el camino a la verdadera felicidad.

A mi madre Miriam la mejor madre de mundo a la cual amo con todas las fuerzas de mi alma, por su protección constante, su amor verdadero y su lucha continúa. Por hacer de nosotros los mejores hijos. Gracias por estar siempre ahí y recibirme con todo su cariño todos los días.

A mi compañero de vida Andrés, sabes que siempre he creído en tus grandes capacidades, que mi fe es tan fuerte como el amor que juntos hemos construido. Amor que ha sobrepasado lo impensable. Gracias por ser mi inspiración y por luchar día a día por alcanzar nuestros sueños. Te Amo.

A mi hermana Ximena porque me demostró que con esfuerzo y dedicación se puede alcanzar grandes triunfos, sabes que siempre me enorgullecido por ti.

A mi princesa Danna, la inspiración más grande de ingenio y ternura. Por ti también lucho y prometo ser tu escudera toda la vida.

Dalila Pachajoa

Esta composición está dedicada a Dios por dotar a los átomos dl universo de vibraciones oscilantes que llenan el espacio tiempo y conforman el compás de nuestras vidas.

A mis padres por entregarme las pautas para leer las notas resonantes del tiempo, guiarme con entonación de su experiencia y enseñarme que siempre es posible corregir las desafinadas pisadas de la vida.

A mis hermanos por proporcionarme los acordes que se expanden en mí ser y guían mi conducta a través de mi alma.

A mis abuelos que en el cielo están por trazar el coro genético en cada sonido profundo entre mis células.

A Miriam Pachajoa por convertirse en una clave de sol cual madre a un hijo iluminando con sus implacables cuidados.

A mi Dali por inspirar las melodías de mi corazón, las tonadas de mis ideas, las partituras de mis metas y por interpretar a dúo esta sinfonía interpretada seguramente hasta la eternidad.

Héctor Andrés Mora

AGRADECIMIENTOS

Al ingeniero Vicente Chamorro, por manifestarnos su interés en dirigir nuestro trabajo de grado.

Al ingeniero Oscar Revelo, por su colaboración, apoyo, revisiones puntuales y aportes precisos en los temas tratados en este proyecto.

Al ingeniero Alexander Barón, por su valioso apoyo e interés en la construcción de este documento.

A nuestra Alma Mater, la Universidad de Nariño por darnos tantos espacios y oportunidades de adquirir conocimiento.

A nuestros docentes de la Universidad de Nariño que compartieron sus conocimientos, dentro y fuera de clase, haciendo posible que nuestra formación profesional se resuma en satisfacciones académicas.

A nuestros compañeros de estudio con los cuales compartimos y construimos grandes momentos.

CONTENIDO

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

MARCAS REGISTRADAS

.NET Framework es una marca registrada de Microsoft Corp.

ANDROID es una marca registrada de Google Inc.

APACHE es una marca registrada de ASF (Apache Software Foundation).

DREAMWEAVER® es una marca registrada de Adobe Systems.

ECLIPSE es una marca registrada de Eclipse Foundation.

GOOGLE MAPS™ es una marca registrada de Google Inc.

IIS es una marca registrada de Microsoft Corp.

LINUX es una marca registrada de Linus Torvalds.

MAC (Macintosh) es una marca registrada de Apple Inc.

PYTHON es una marca registrada de Python Software Foundation.

RED HAT es una marca registrada de Red Hat Inc. Compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux.

RUP es una marca registrada de International Business Machines Corporation.

UML es una marca registrada de OMG, consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objeto.

VISUAL BASIC® es una marca registrada de Microsoft Corp.

VISUAL C#® es una marca registrada de Microsoft Corp.

VISUAL STUDIO® es una marca registrada de Microsoft Corp.

WINDOWS 7® es una **l**ínea de sistemas operativos registrada por Microsoft Corp.

WINDOWS SERVER™ es una línea de productos para servidores de Microsoft Corp.

GLOSARIO

- **Acoplamiento**: indica el nivel de dependencia entre las unidades de software de un sistema informático, es decir, el grado en que una unidad puede funcionar sin recurrir a otras; dos funciones son absolutamente independientes entre sí (el nivel más bajo de acoplamiento) cuando una puede hacer su trabajo completamente sin recurrir a la otra.

- **API:** Interfaz de programación de aplicaciones, es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

- **Bucles**: un bucle, en programación, es una sentencia que se realiza repetidas veces a un trozo aislado de código, hasta que la condición asignada ha dicho bucle deje de cumplirse. Generalmente, un bucle es utilizado para hacer una acción repetida sin tener que escribir varias veces el mismo código, lo que ahorra tiempo, deja el código más claro y facilita su modificación en el futuro.

- **CLI**: Interfaz de Línea de Comandos, es un método que permite a las personas dar instrucciones a algún programa informático por medio de una línea de texto simple. Las CLI pueden emplearse interactivamente, escribiendo instrucciones en alguna especie de entrada de texto, o pueden utilizarse de una forma mucho más automatizada (archivo batch), leyendo comandos desde un archivo de scripts.

- **Componente software**: es un elemento de un sistema software que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces definidas. Un componente de software debe poseer las siguientes características: ser reutilizable, ser intercambiable, poseer interfaces definidas y ser cohesivos.

- **DLL**: Una biblioteca de enlace dinámico, es el término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación es exclusiva a los sistemas operativos Windows siendo ".dll" la extensión con la que se identifican estos ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.

- **Escalabilidad**: es la propiedad deseable de un software que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

- **Extensibilidad**: es un principio de diseño de sistema en el que la aplicación tiene en cuenta el crecimiento futuro. Es una medida sistémica de la capacidad de extender un sistema y el nivel de esfuerzo que se requiere para poner en práctica la extensión. Las extensiones pueden ser a través de la adición de nuevas funcionalidades o mediante la modificación de las funcionalidades existentes.

- **Framework**: La palabra inglesa "framework" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- **Grafo**: es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Son objeto de estudio de la teoría de grafos. Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras.

- **Heterogeneidad**: Un sistema heterogéneo es aquel que se encuentra compuesto por hardware con características físicas distintas entre sí, y software con características operativas distintas entre sí, pero que se pueden comunicar utilizando medios comunes.

- **Integración**: consiste en conectar distintas aplicaciones informáticas con la intención de que la información sea compartida entre ellas. El objetivo de la integración de sistemas es doble.

- **Interoperabilidad**: habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

- **Latencia**: suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

- **Bibliotecas**: es un conjunto de implementaciones de comportamiento, escritas para un lenguaje de programación, que tienen una interfaz bien definida para el comportamiento que se invoca.

- **Matriz**: es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, los elementos de la matriz. Desde el punto de vista lógico una matriz se puede ver como un conjunto de elementos ordenados en fila (o filas y columnas si tuviera dos dimensiones).

- **Middleware**: es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos. De esta forma se provee una solución que mejora la calidad de servicio, seguridad, envío de mensajes, directorio de servicio, etc.

- **Monolíticos**: En los Sistemas Operativos Monolíticos, existen módulos grandes en el núcleo, los cuales interactúan entre sí, para poder tener esta estructura, las diferentes partes del kernel son compiladas por capas.

- **Multi-acceso**: Un sistema multi-acceso es el que permite a varios usuarios cada uno desde su terminal, a hacer uso de un mismo ordenador simultáneamente.

- **Multilenguaje**: capacidad de un software de reutilizarse en diferentes lenguajes de programación.

- **Multinúcleo**: es aquel que combina dos o más microprocesadores independientes en un solo paquete, a menudo un solo circuito integrado. Un dispositivo de doble núcleo contiene solamente dos microprocesadores independientes.

- **Multiplataforma**: se dice que un elemento (programas, lenguajes de programación, elementos de hardware...) es multiplataforma cuando tiene la capacidad de funcionar en más de un sistema operativo con similares características y sin que su funcionalidad varíe en exceso.

- **Multiplicidad**: indica que cantidad de objetos de una clase se relaciona con un objeto de la clase, asociar la línea utilizada para esta relación es una línea continua y en los extremos se coloca la cantidad de objetos que se relaciona, es el equivalente a la cardinalidad en caso de uso.

- **Multiusuario**: característica de un sistema operativo o programa que permite proveer servicio y procesamiento a múltiples usuarios simultáneamente.

- **OData**: El Open Data Protocol (OData) es un protocolo de acceso de datos para la Web. OData proporciona una manera uniforme para consultar y manipular conjuntos de datos a través de operaciones CRUD (crear, leer, actualizar y eliminar).

- **Parser**: Un analizador sintáctico (o parser) es una de las partes de un compilador que transforma su entrada en un árbol de derivación. El análisis sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada.

- **Performance**: desempeño con respecto al rendimiento de una computadora, un dispositivo, un sistema operativo, un programa o una conexión a una red. En informática, medida o cuantificación de la velocidad/resultado con que se realiza una tarea o proceso.

- **Portabilidad**: es la capacidad de uso del mismo software en diferentes entornos. Cuando el software con la misma funcionalidad es producido por varias plataformas de computación, la portabilidad es la cuestión clave para la reducción de los costes de desarrollo.

- **Problema de transporte**: El problema general del transporte se refiere a la distribución de mercancía desde cualquier conjunto de centro de suministro, denominados orígenes, hasta cualquier conjunto de centros de recepción,

llamados destinos, de tal forma que se minimicen los costos totales de distribución.

- **Problema Dual**: es una programación lineal definida en forma directa y sistemática a partir del modelo original (o primal) de programación lineal. Los dos problemas están relacionados en forma tan estrecha que la resolución óptima de un problema produce en forma automática la resolución óptima del otro.

- **Protocolos**: es el conjunto de reglas y estándares que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red, como teléfonos o computadoras.

- **Proxy**: es un programa o dispositivo que realiza una acción en representación de otro, esto es, si una hipotética máquina A solicita un recurso a una C, lo hará mediante una petición a B; C entonces no sabrá que la petición procedió originalmente de A. Esta situación estratégica de punto intermedio suele ser aprovechada para soportar una serie de funcionalidades: proporcionar caché, control de acceso, registro del tráfico, prohibir cierto tipo de tráfico, etc.

- **Recurrente**: un método usual de simplificación de un problema complejo es la división de este en subproblemas del mismo tipo. Esta técnica de programación se conoce como divide y vencerás y es el núcleo en el diseño de numerosos algoritmos de gran importancia, así como también es parte fundamental de la programación dinámica.

- **RPC**: llamada a Procedimiento Remoto, es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

- **Reutilización de software**: la reutilización de software aparece como una alternativa para desarrollar aplicaciones y sistemas SW de una manera más eficiente, productiva y rápida. La idea es reutilizar elementos y componentes SW en lugar de tener que desarrollarlos desde un principio.

- **Robustez**: capacidad y proceso de reacción apropiada ante condiciones que se encuentren fuera del alcance del software, estas condiciones son excepcionales.

- **URI**: identificador uniforme de recursos es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema. Los URI pueden ser localizadores uniformes de recursos (URL).

- **URL**: localizador uniforme de recursos, es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

- **Windows Forms**: es el nombre dado a la interfaz de programación de aplicaciones (API) que se incluye como parte de Microsoft. Visual Studio proporciona un entorno de desarrollo integrado que ayuda a escribir el código, así como un completo conjunto de controles escrito con .NET Framework. La funcionalidad de estos controles se complementa con el código escrito por el desarrollador, lo que permite desarrollar fácil y rápidamente las soluciones deseadas.

RESUMEN

SISNOVA, es un sistema de componentes software reutilizables y configurables, con características multiplataforma, multiusuario, multi-acceso y facilidad de uso, para la resolución del problema de transportes. Presentándose como solución innovadora y referente en cuanto a las necesidades de construir software de forma ágil, imitando a los procesos de construcción de la industria contemporánea y manifestándose como apoyo a los usuarios programadores.

Teniendo en cuenta el estado del arte de las arquitecturas enfocadas a la reutilización, se opta por desarrollar un sistema que se consuma bajo las arquitecturas SOA y BCSD, amparadas en los estándares de la comunicación y la información. Contemplando de esta manera una solución multiplataforma apoyada con la garantía que ofrece la metodología RUP.

SISNOVA está conformado por un componente principal que encapsula la lógica de negocios y los datos para la solución del problema de transportes en un servicio Web WCF hospedado en IIS, y una biblioteca de controles de usuario con la capacidad de integrarse en el API Windows Forms de .NET.

El servicio encapsula la lógica de negocios bajo una abstracción del problema de transportes sintetizada en 3 momentos de resolución: balanceo, solución inicial y mejoramiento de la solución soportada bajo un adecuado modelado de artefactos de información, Además, permite el envío y recepción de datos compuestos, peticiones síncronas y asíncronas, consumo mediante SOAP o REST y en este último permite la recepción y envió de información a través de los formatos XML y JSON.

Finalmente, las pruebas de funcionamiento de SISNOVA se realizaron mediante la construcción de prototipos de aplicación de escritorio, en los lenguajes C# y VB de .NET y JAVA; aplicación Web en PHP hospedada en WINDOWS SERVER y RED HAT; y aplicación móvil ANDROID desarrollada en JAVA. Las cuales concluyen exitosamente.

Palabras Claves: servicio Web, reutilización, lógica de negocios, multiplataforma, arquitectura.

ABSTRACT

SISNOVA is a system made of software components which are reusable and configurable, having features of multiplatform, multiuser and multi-access which make it easy to use it in order to work out problems of transportation. It is presented as an innovative solution giving a response to the needs of making up effective software that imitates the processes of building of the modern industry which become a support for the software developers.

Having in mind the state of the art in the architecture focused on the reuse, the development of a system is aimed to be applied for the architectures SOA and BCSD, working under the standards of communication and information. In this way, a multiplatform solution is considered being supported by the methodology RUP.

SISNOVA is made with a main component which encloses the business logics and the data for the solution of the problem of transportation in a Web service WCF host in IIS, and a library of controls of the user having the capacity of being integrated in the API Windows Forms of .NET.

The service encloses the business logics through the abstraction of the problem of transportation synthesized in three moments of solution: balancing, initial solution and improvement of the solution supported by the adequate modeling of information devices. In addition to that, it allows the delivery and reception of composed data, synchronic and asynchronic requests and the consumption through SOAP or REST, and this last one allows the reception and delivery of information through the formats XML and JSON.

Finally, the samples of working of SISNOVA were developed though the construction of prototypes of application of desktop in language C# and VB of .NET and JAVA; Web application in PHP host in WINDOWS SERVER and RED HAT; and mobile application ANDROID developed in JAVA, which resolve successfully.

Keywords: Web service, reuse, business logics, multiplatform, architecture.

INTRODUCCIÓN

Esta investigación se apoya en metodologías y diseños estructurados, como los que ofrece la Ingeniería de Software y el conocimiento de otras disciplinas del saber, logrando así con un trabajo interdisciplinario el desarrollo de sistemas complejos, con altos estándares de calidad y desempeño.

Como una medida de solución para la crisis del software en los 80, nació la ingeniería de requisitos, para el aseguramiento de la calidad del software, como consecuencia la producción del software se vio insuficiente ante los niveles de demanda. Entonces se propone un modelo de reutilización y buenas prácticas para cada una de las etapas de los procesos software, imitando los métodos de producción de productos tangibles. Aunque las mejoras en la producción de programas computacionales avanzan sustancialmente con la reutilización, en tiempos de codificación y ejecución sigue existiendo algunos limitantes en cuanto al uso en múltiples plataformas y concurrencia de usuarios. Para abordar esta problemática se han desarrollado estándares y protocolos computacionales, de comunicación, de información y en su conjunto, nacen tecnologías para afrontar a estas limitaciones ya que las soluciones de hoy en día deben llevar intrínsecas toda la información disponible para cualquier persona, en cualquier lugar, a través de cualquier dispositivo y bajo cualquier lenguaje de programación.

Dentro de las grandes decisiones que se han tomado en el transcurso de la historia, el hombre siempre ha ido en búsqueda de herramientas que le permita tomar las mejores decisiones, es entonces que durante la segunda guerra mundial surgió la investigación de operaciones, para hacer uso de la ciencia en las operaciones militares, en los que se usaron de forma óptima los recursos disponibles logrando así sus objetivos. Ya hoy en día, la investigación de operaciones es catalogada como optimización de sistemas, dominante e indispensable para la toma de decisiones. Entre los primeros objetivos perseguidos estaban: el minimizar el costo de transporte y al mismo tiempo satisfacer los límites de oferta y demanda, así como reconocer que puede ser más económico el transporte pasando por puntos intermedios, por lo cual los primeros problemas que se abordaron fueron los de transportes.

Según lo anterior, se presenta un proyecto enfocado a la implementación de un sistema de recursos reutilizables que contribuyan al desarrollo de múltiples aplicaciones escalables, robustas, para los desarrolladores que necesiten componentes software cuyas funcionalidades permitan la resolución de problemas de transportes.

En este documento se describe la construcción de SISNOVA, y se constituye de la siguiente manera: descripción del problema, objetivos, justificación, marco

teórico, desarrollo del sistema, prototipos de aplicación y por último se contemplarán los resultados que se esperan de esta investigación. Donde con el desarrollo del sistema se pueda aportar en el impacto social, en la transformación de algún tipo de trabajo y hasta un el cambio en la vida diaria.

PLANTEAMIENTO DEL PROBLEMA

Actualmente la mayoría de construcciones se realizan con elementos prefabricados, dotados de un alto acoplamiento y flexibilidad, superiores a los de años anteriores. Aunque el software no se escapa de esta premisa y al respecto se han dado pasos agigantados, se pueden observar algunas limitaciones para el rehuso en los diferentes dispositivos, sistemas operativos y lenguajes de programación en cuanto a elementos reutilizables en tiempo de codificación y ejecución. Y a lo sumo se pueden encontrar en algunas soluciones, problemas de rendimiento y seguridad.

Los limitantes en cuanto a los diferentes artefactos de ejecución, se deben a que el hardware de un dispositivo puede presentar restricciones en el procesamiento y almacenamiento aleatorio o permanente, e incluso carecer de alguno de estos. Esto implica que programas que demanden una ejecución exigente tengan una respuesta lenta, inapropiada y en el peor de los casos simplemente no funcionar. En cuanto al software, las bibliotecas de clases y bibliotecas estáticas pueden acoplarse si el compilador del lenguaje traduce el código fuente a instrucciones compatibles al sistema operativo, pero esto no es suficiente, ya que el compilador además debe tener la misma estructura y especificaciones del fabricante. Otros elementos como las bibliotecas DLL y componentes COM que son código ejecutable, si bien pueden enlazarse en distintos lenguajes de programación, su mayor desventaja es que solo funcionan en la arquitectura y sistema operativo donde fueron concebidos.

Para solventar en gran medida estas restricciones y aprovechando la aparición de los estándares abiertos de las TIC, la cobertura de Internet, las arquitecturas cliente-servidor y distribuidas, nacen los servicios Web. Con ellos la ejecución se le encarga al servidor alivianando la tarea a clientes con características de hardware inferiores. Debido a que los protocolos de información y de comunicación son estándares, la incorporación en diferentes plataformas no solo se posibilita si no también se facilita en grandes proporciones.

Por otro lado, no todo puede estar solucionado y siempre harán falta elementos reutilizables. Uno de los tantos casos está en la investigación de operaciones u optimización de sistemas y más específicamente para los modelos de transportes, donde se cuenta con algunas aplicaciones pero el acceso al código fuente es de difícil consecución, y adaptación, en algunos casos no satisface las necesidades y además se le suma los limitantes nombrados anteriormente.

De prevalecer las anteriores limitaciones, un desarrollador dedicaría esfuerzo extra para permitir el funcionamiento deseado de las soluciones software existente en las diferentes plataformas, ya sea adaptando el hardware del dispositivo a la solución o viceversa. Cuando el código fuente a reutilizar este escrito en un lenguaje de programación A diferente al que el desarrollador maneja, esté tendría que migrar el código a su lenguaje de programación A o migrarse él al lenguaje de programación B, en cuyo caso se desaprovecharía el conocimiento, la pericia y experiencia del desarrollador sobre el lenguaje A. En cuanto a la seguridad se tendría que incorporar técnicas de cifrado y encriptación, y cambios en los protocolos de comunicación. Todo esto implicaría un aumento considerable en el tiempo de construcción y los costos del proyecto, conjuntamente se tendrían diferentes aplicaciones con el mismo objetivo, pequeñas en cuanto a tiempo de construcción e inaccesibilidad a las posteriores mejoras de los elementos prefabricados.

Formulación del problema

¿Cómo contribuir con los desarrolladores de software que requieran en sus aplicaciones la incorporación de funcionalidades reutilizables y configurables para ser adoptadas en tiempo de codificación, cuyo soporte tenga lugar en distintos dispositivos, sistemas operativos y lenguajes de programación y su consumo ocurra en tiempo de ejecución?

¿Cómo implementar funcionalidades que sean de apoyo al desarrollo de aplicaciones que tengan que ver con la resolución de problemas de transportes?

Sistematización del problema

- ¿Actualmente qué avances tecnológicos para la construcción de elementos software reutilizables logran la interoperabilidad en diferentes dispositivos, sistemas operativos y lenguajes de programación?
- ¿Qué tecnología se debería escoger para desarrollar los elementos de reutilización a construir?
- ¿Cómo detectar las posibles funcionalidades requeridas por el usuario para estos elementos reutilizables?
- ¿Qué funcionalidades para la resolución de problemas de transportes son necesarias incorporar?
- ¿Cómo llevar las funcionalidades a incorporar al contexto de solución?
- ¿Qué pasos se deben seguir para acoplar la solución a las aplicaciones de los programadores?
- ¿Cómo lograr un buen manejo de parte de los desarrolladores sobre la solución?

Alcance y delimitaciones

El presente trabajo se enfoca en la implementación de funcionalidades reutilizables y configurables que se han separado en dos componentes: la implementación, de un servicio Web y de una biblioteca de controles de usuario. Componentes que sirven como apoyo al desarrollo de diferentes aplicaciones relacionadas con la resolución de problemas de transportes.

Para el caso del servicio Web se ofrecen funcionalidades o recursos para la resolución de los problemas de transportes, bajo el soporte de mensajes SOAP y REST y donde se implemente en tiempo de codificación y su consumo se lleve a cabo en tiempo de ejecución y su uso ofrezca soporte en la mayoría de: dispositivos de ejecución, sistemas operativos y lenguajes de programación; teniendo en cuenta las siguientes limitaciones:

- El servicio solo resuelve problemas de transportes.
- SISNOVA, ofrece funcionalidades que permiten el desarrollo de los problemas tanto de forma iterativa como puntual.
- La solución factible inicial se obtiene solo para los métodos de esquina noroeste, costo mínimo, Vogel y Russell.
- Las iteraciones para la obtención de las sucesivas soluciones factibles se resuelven por el método de multiplicadores.

Líneas de investigación

Las líneas de investigación en las que se enmarca este trabajo de grado son las correspondientes a: Sistemas Computacionales y Optimización de Sistemas

Modalidad

Este trabajo de grado se enmarca en la modalidad de Trabajo de Investigación.

OBJETIVOS

Objetivo general

Implementar SISNOVA, sistema de componentes multiplataforma, reutilizables y configurables basado en los protocolos y estándares de las TIC y en tecnologías de la Web 2.0, como solución a las necesidades que a un desarrollador de software se le puedan presentar al cubrir los requerimientos en cuanto a las funcionalidades de sus aplicaciones en tiempo de codificación y ejecución para problemas de transportes.

Objetivos específicos

- Identificar las funcionalidades para la resolución de los problemas de transportes para garantizar soluciones a la medida de las necesidades a cubrir.

- Crear prototipos de prueba de aplicaciones Web, escritorio y móvil que consuman los recursos reutilizables de SISNOVA en diferentes plataformas de desarrollo.

- Verificación del funcionamiento de SISNOVA mediante la validación de los cuatro prototipos de aplicación.

JUSTIFICACIÓN

Cada vez se hace más apremiante optimizar el funcionamiento de sistemas reales teniendo en cuenta las limitaciones de recursos disponibles y los objetivos que se tengan. La investigación de operaciones ofrece métodos para abordar distintos tipos de problemas para obtener el mejor beneficio, es por esto que se hace indispensable adoptar técnicas de investigación de operaciones en aquellos tipos de sistemas que sean acordes con los modelos establecidos y de esta manera conseguir un mejoramiento continuo y una calidad garantizada. Por eso es de gran importancia difundir el objetivo de estas técnicas y ofrecer herramientas que faciliten la resolución de estos problemas. Para el caso de estudio se profundizo en los métodos de transportes para los cuales se integraran funcionalidades encapsuladas para que sea posible la construcción de distintas soluciones según sea sus requerimientos. La construcción de dichos servicios contempla la creación de soluciones de diferentes tipos entre ellas: las de contexto educativo que facilitan el aprendizaje, las de enfoque didáctico que permitan una visualización detallada del problema a resolver, o de cálculo donde lo que prima es la solución final del problema.

Pero la construcción de diferentes soluciones es difícil contemplarla sin un conjunto de servicios reutilizables, robustos y adaptables, es por esto que los elementos reutilizables a ofrecer deben contar con la capacidad de funcionar en distintos dispositivos de ejecución, sistemas operativos y lenguajes de programación, ya que de esta manera un desarrollador de software no tendría que realizar trabajo extra traduciendo código fuente, creando puentes de enlace o en el peor de los casos codificar todo el componente reutilizable que necesita. De existir el elemento reutilizable multiplataforma el programador solo se preocuparía por buscar el elemento, ver su metadata y acoplar las funcionalidades a la solución, es decir que el desarrollador programaría su solución de forma acostumbrada y el tiempo en terminar sus programas decrecería en gran medida y no solo en tiempo de codificación sino también en tiempo de prueba ya que los elementos

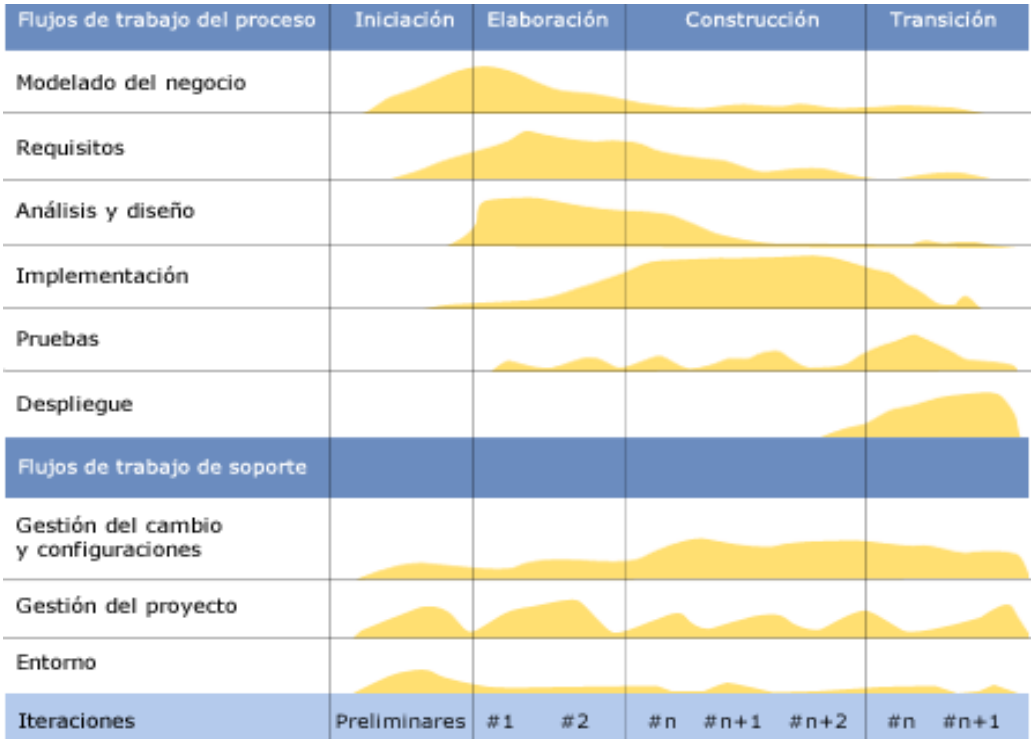
reutilizables acoplados se supone han sido probados con anterioridad. Lo anterior además lleva intrínsecos beneficios como la integración de hardware y software, amplia difusión y cobertura como también independizar y encapsular la lógica de negocio ofreciendo de esta manera una mayor integridad, seguridad de la información y beneficios económicos en el mercado, tanto para los clientes que requieren la aplicación como para los contratistas, debido a que la cantidad de trabajo y la producción son proporcionales al tiempo.

Para la academia SISNOVA se convertiría en un referente de construcción de elementos reutilizables multiplataforma, aportando conocimiento en las nuevas tecnologías de la información y la comunicación, adopción de estándares, arquitectura software, redes y lenguajes de programación.

METODOLOGÍA

Para el desarrollo de SISNOVA, se ha escogido al proceso de software RUP (Proceso Unificado Rational) el cual propone el desarrollo de software basado en un conjunto de fases, cada una de las cuales se desarrolla por medio de una o varias iteraciones donde por cada iteración se desarrollan un conjunto de actividades llamadas flujo de trabajo y flujo de soporte a continuación se muestra un esquema para RUP.

Figura 1. Esquema RUP.



En la figura 1, el eje horizontal representa el tiempo y muestra los aspectos del ciclo de vida del proceso. El eje vertical representa las disciplinas que agrupan actividades por su naturaleza.
RUP, está centrado en la arquitectura, dirigida por casos de uso, iterativo e incremental. RUP se dice que es centrado en arquitectura por que el trabajo se centra en el desarrollo de software, identificación de componentes software y definición de funcionalidad de componentes, gracias a que se definen por medio de requisitos.

Para lo cual en esta investigación se escogió el proceso RUP y las fases a seguir se contemplaron en un cronograma inicial.

Así mismo para la descripción del sistema se toma el Lenguaje de Modelamiento Unificado (UML), ya que es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

1. MARCO TEÓRICO

1.1 INTRODUCCIÓN

La tecnología moderna y los computadores han cambiado la forma de resolver las dificultades en el mundo actual. Así mismo, para lograr un gran cambio se necesita que estos sistemas sean óptimos y la implementación de procesos de reutilización de software forma parte fundamental para lograrlo. La optimización de sistemas es la selección del mejor elemento con respecto algún criterio de un conjunto de elementos disponibles. Elementos que pueden ya existir y solo se necesita utilizarlos[1]. La optimización de sistemas es el proceso de modificación de un sistema para hacer que algún aspecto del mismo funcione de manera más eficiente y/o utilizar menos recursos[2]. De acuerdo a estos conceptos, el emplear elementos software que permite reducir los tiempos y simplificar el desarrollo del software, mejorando la calidad y reduciendo su costo. La optimización entonces puede llevarse a cualquier contexto, para el caso de estudio la optimización se ha contextualizado a la construcción de software, específicamente para SISNOVA, derivándose en dos aspectos: La reutilización de software y la investigación de operaciones.

Con la acogida de la reutilización de software son innumerables las innovaciones que ha habido para crear nuevos sistemas que permitan, tanto a las personas como a las organizaciones realizar satisfactoriamente sus actividades. La reutilización entonces, se ha vuelto un apoyo indispensable, ya que permite el uso de artefactos software ya existente que ayudan a reducir tiempos y costos en la construcción de aplicaciones. En cada etapa de la construcción de software se cuenta con diferentes insumos a reutilizar, puntualmente para la etapa de implementación como gran innovación se cuenta con los servicios Web, que se presentan como una tecnología que utiliza un conjunto de protocolos y estándares que permiten el intercambio de datos entre diferentes aplicaciones, logrando así vencer limitaciones antiguas que no permitían avanzar más allá del desarrollo de simples aplicaciones de escritorio.

Para lograr el propósito de optimizar sistemas, uno de los factores que se debe tener en cuenta es contar con campos que podrían ser considerados con estrechas vinculaciones a la optimización de sistemas. Uno de estos campos es la investigación de operaciones.

---

[1] DANTIZING, GEORGE B. Linear Programming: 2: Theory and Extensions, Springer, 2003. ISBN 0-387-98613-8

[2] SEDGEWICK, ROBERT. “Algorithms”. 4º Ed., Pearson Education, Inc., 2011. ISBN 978-0-321-57351-3

La investigación de operaciones, ha contribuido en el desarrollo de la optimización de sistemas como un área independiente, que estudia sistemas reales y trata la optimización de procesos bajo múltiples restricciones, con la finalidad de optimizar su funcionamiento. El objetivo más importante de la investigación operativa es apoyar en la toma de decisiones de los sistemas y en la planificación de sus actividades. Además, uno de los temas tomados a fondo en esta investigación es el uso del modelo de transportes, el cual busca determinar un plan de transporte de una mercancía de varias fuentes a varios destinos.

Para el desarrollo de SISNOVA, este se soporta de tecnologías actuales, que permiten que el sistema sea robusto y configurable, y con la implementación y uso de las herramientas teóricas se tiene como finalidad dar una solución óptima para un determinado problema de transportes.

A continuación, se realiza una descripción de los temas principales que se abordan en SISNOVA.

1.2 ANTECEDENTES

En la actualidad se encuentran diversos elementos para la reutilización, al igual que herramientas para modelar y resolver problemas de Transportes. En esta sección se referencia algunas de estas herramientas con el propósito de acoger experiencia y conocimiento que sirvan como base para esta investigación.

En el ámbito internacional.

En cuanto a la reutilización en tiempo de codificación están los siguientes antecedentes.

- COMPONENTES COM: es una plataforma de Microsoft[3] para componentes de software introducida por dicha empresa en 1993. Esta plataforma permite la comunicación entre procesos y la creación dinámica de objetos en cualquier lenguaje de programación que soporte dicha tecnología. En cuanto a la creación de los componentes, COM permite la reutilización de objetos sin conocimiento de su implementación interna, porque fuerza a los implementadores de componentes a proveer interfaces bien definidas que están separados de la implementación.

- SERVICIOS WEB: desarrollado por la W3C[*] en 1999, es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar

[3] MICROSOFT, Corp. <http://WWW.microsoft.com/> [Consultado el 12 de octubre 2013].
[*] W3C: es un consorcio internacional que produce recomendaciones para la World Wide Web. Está dirigido por Tim Berners-Lee, el creador original de URL, HTTP y HTML.

datos entre aplicaciones. Para el desarrollo de SISNOVA se utilizan los servicios Web para intercambiar datos en redes de ordenadores como Internet, la interoperabilidad se consigue mediante la adopción de estándares abiertos.

En cuanto a la investigación de operaciones en su campo de transportes se encuentran los siguientes antecedentes.

- INVOP (INVestigación de OPeraciones 1.0): este software fue desarrollado por Beatriz Loubet y Sandra Segura[4]. Para el presente trabajo, INVOP ayuda a manera de guía en la resolución de la línea de investigación ya que resuelve problemas como: transporte, distancia, y ruta más corta.

- TORA: Software incluido en el libro de Hamdy Taha[5], 2004. Es una herramienta que permite resolver problemas de Programación Lineal, Modelos de Transportes, entre otros; por medio de los diferentes métodos existentes de Investigación de Operaciones. Este software aporta al presente trabajo la función de, que las entradas para el software son los modelos que deben ser previamente creados por el usuario, y la salida es la solución óptima según sea el caso, además presenta opciones de ver el proceso por pasos lo que es de mucha ayuda a los estudiantes que están aprendiendo los métodos de solución de estos problemas y lo que permite tomar como base para la implementación de funcionalidades interactivas.

En el ámbito nacional.

En Colombia se han encontrado paquetes que enfatizan en los métodos matemáticos para la resolución de los problemas de Investigación de Operaciones, como:

Simplex2004: es un paquete de software didáctico, diseñado para gerentes ejecutivos con títulos, pero también se convierte en herramienta didáctica para profesores universitarios y estudiantes que buscan atinar en el proceso de enseñanza - aprendizaje. Lo que permite tomarlo como base porque aplica el algoritmo Simplex de Programación Lineal.

En el ámbito regional.

En la región se encuentra:

[4] LOUBET, Beatriz. SEGURA, Sandra, Universidad Nacional de Cuyo, Argentina, 1997.
[5] TAHA, Hamdy, Investigación de Operaciones. 7ª Edición. University of Arkansas, Fayetteville. Person Educación. México. 2004. p. 33–35.

SIMPLEX MEDIA, un software multimedial de apoyo al estudio del método Simplex[6]. Esta herramienta sirve como base ya que soluciona mediante los procedimientos matemáticos pertinentes, problemas de Investigación de Operaciones a través del método Simplex que deben ser modelados anteriormente por el usuario.

DIONISIO, un prototipo de modelador de problemas de Simplex, asignación y trasportes dentro del contexto de la Investigación de Operaciones[7]. Es un gran aporte en cuanto a su modularidad, arquitectura y esquematización de la solución.

La mayor diferencia con los software nombrados anteriormente, es que SISNOVA abstrae las funcionalidades de la lógica de negocios en la Web para que puedan ser reutilizadas en cualquier plataforma y estén disponibles en cualquier momento.

1.3 REUTILIZACIÓN DE SOFTWARE

1.3.1 Generalidades. Con el estudio de la reutilización se quiere lograr que los usuarios desarrolladores tengan un alto nivel de reutilización en el proceso de desarrollo de sistemas, ya que representan una meta a conseguir y constituye una señal de madurez en cualquier disciplina de ingeniería. De acuerdo a las diferentes investigaciones sobre la reutilización se proponen una diversidad de definiciones entre las cuales están. La reutilización de software es la disciplina encargada de la creación de sistemas de software a partir de software pre-existente[8]. La reutilización es el proceso de creación de sistemas de software a partir de partes de software existente en lugar de construir desde cero[9]. Se puede definirla entonces como el empleo de elementos de software, creados en desarrollo anteriores, para de este modo reducir los tiempos y simplificar el desarrollo del software, mejorando la calidad y reduciendo su costo. A estos elementos a reutilizar se los define como componentes software.

1.3.1.1 Reutilización de componentes software. La aplicación de este concepto al desarrollo de software no es nueva. Las bibliotecas, comúnmente utilizadas desde la década de los setenta en el campo informático, representan uno de los primeros intentos por reutilizar software.

---

[6] ASCUNTAR, Sandra, CASTILLO, James, CORREA, Ronald, Simplex Media, un software multimedial de apoyo al estudio del método simplex, Universidad de Nariño, Pasto, 2009.
[7] RUANO, Jairo, CABRERA, Christian, Dionisio, un prototipo de modelador de problemas de simplex, asignación y trasportes dentro del contexto de la Investigación de Operaciones, Universidad de Nariño, Pasto, 2010.
[8] KRUEGER, Charles, Software Reuse, ACM Computing Surveys, Volume 24, 1992. p. 131-183.
[9] SOMMERVILLE, Ian. Ingeniería de Software, Pearson Educación, España, 2005. p. 379.

La reutilización de componentes de software es un proceso inspirado en la manera en que se producen y ensamblan componentes en la Ingeniería de Sistemas, y se pueden catalogar de la siguiente manera:

- **Activo de software**: es un producto diseñado expresamente para ser empleado de forma recurrente* en el desarrollo de muchos sistemas y aplicaciones. Entre los ejemplos de activos reutilizables están: algoritmos, patrones de diseño, esquemas de base de datos, arquitecturas de software, especificaciones de requerimientos, de diseño y de prueba, entre otros.

- **Componente de software**: como resultado de presiones crecientes sobre la industria del software orientadas a reducir drásticamente el costo y tiempo de desarrollo de sistemas y aplicaciones, sin afectar los niveles de calidad, ha surgido un nuevo activo reutilizable denominado Componente de Software. Un componente de software es una implementación opaca de funcionalidad, sujeta a composición por terceros. Con respecto al primer aspecto, un componente se considera una implementación opaca debido a que su distribución predominantemente es en formato binario y sus consumidores lo utilizan como una "caja negra" a través de su interfaz. Dicho aspecto está alineado con el principio de encapsulamiento de la programación orientada a objetos. Por otra parte, la composición por terceros implica que los componentes son intrínsecamente reutilizables debido a que un sistema basado en componentes puede ser ensamblado con relativa facilidad por un integrador empleando componentes suministrados por múltiples proveedores independientes.

Además de definir estos elementos a continuación, se especifican las unidades de software se pueden reutilizar.

1.3.1.2 Unidades de software reutilizables. Las unidades de software reutilizables pueden ser de diferentes tamaños.

- **Reutilización de sistemas de aplicaciones**: es a menudo la técnica de reutilización más efectiva. Implica la reutilización de activos de grano grueso que pueden ser rápidamente configurados para crear un nuevo sistema. Su totalidad se puede reutilizar, solo configurado para diferentes clientes, o aplicaciones con arquitectura común que se adapta a clientes particulares.

- **Reutilización de componentes**: la reutilización de componentes de una aplicación varía en tamaño desde subsistemas hasta objetos simples. Por ejemplo, un sistema de emparejamiento de patrones desarrollado como parte de un sistema de procesamiento de textos puede ser reutilizado en un sistema de gestión de base de datos.

* Recurrente se refiere a que algo que se repite

- **Reutilización de objetos y funciones**: reutilización de componentes con una única función, ejemplo: función matemática o clase de objetos. Reutilización basada en bibliotecas estándar que se utiliza desde hace ya 40 años, más aun en áreas como algoritmos matemáticos y gráficos.

Los sistemas y componentes software son entidades reutilizables específicas, pero su naturaleza específica significa algunas veces que el coste de modificarlos para una nueva situación resulta elevado. Una forma complementaria de reutilización es la **reutilización de conceptos**, en la que, en lugar de reutilizar un componente, la entidad reutilizada es más abstracta y se diseña para ser configurada y adaptada a una variedad de situaciones.

La reutilización de conceptos puede incluirse en aproximaciones, tales como: patrones de diseño, productos de sistemas configurables y generadores de programas. El proceso de reutilización, cuando se reutilizan los conceptos, incluye una actividad de instanciación en la que los conceptos abstractos se configuran para una situación concreta[10].

La ventaja obvia de la reutilización de software es que los costes totales de desarrollo deben reducirse. Se necesita especificar, diseñar, implementar y validar menos componentes software. Sin embargo, la reducción de costes es sólo una ventaja de la reutilización.

1.3.1.3 Beneficios de la reutilización. Los beneficios[11] que trae consigo la reutilización, son los siguientes:

- Incremento de la confiabilidad: un software ya usado debería ser más confiable, ya que sus fallos ya han sido encontrados y reparados.

- Reducción del riesgo del proceso: el coste del software existente ya es conocido mientras el coste del desarrollo no, con esto se reduce la estimación del coste.

- Uso efectivo de especialistas: no se hace lo mismo una y otra vez, se puede desarrollar software reutilizable, encapsulando su conocimiento.

- Cumplimiento de estándares: los estándares de interfaz de usuario se pueden reutilizar logrando mejorar la confiabilidad ya que los usuarios comenten menos errores cuando se encuentran una interfaz familiar.

[10] Ibid. p. 380.
[11] Ibid. p. 381.

- Calidad: Las correcciones de errores se acumulan de reutilización a reutilización por lo que se debe ser cuidadoso, la utilización de sistemas ya probados aumenta la fiabilidad y por tanto garantizan su calidad.

- Productividad: se logra debido a la menor cantidad de código que tiene que ser desarrollado.

- Interoperabilidad: Varios sistemas pueden trabajar mejor juntos si sus interfaces se aplican de forma coherente.

- Desarrollo acelerado: sacar al mercado un sistema tan pronto como sea posible es a menudo más importante que los costes totales de desarrollo. La reutilización del software puede acelerar la producción del sistema debido a que se reducen los tiempos de desarrollo y validación.

La reutilización de software es inevitable y positiva. Aunque esta no soluciona todas las problemáticas que se pudieran presentar genera un impacto positivo sobre los costes de desarrollo, la productividad, funcionalidad, calidad, fiabilidad, portabilidad, eficiencia y de mantenimiento de software.

El software con el uso de la reutilización es ahora un activo valioso, por lo que la reutilización es parte integral de grandes compañías. Compañías como Hewlett-Packard han experimentado con éxito la reutilización de programas.

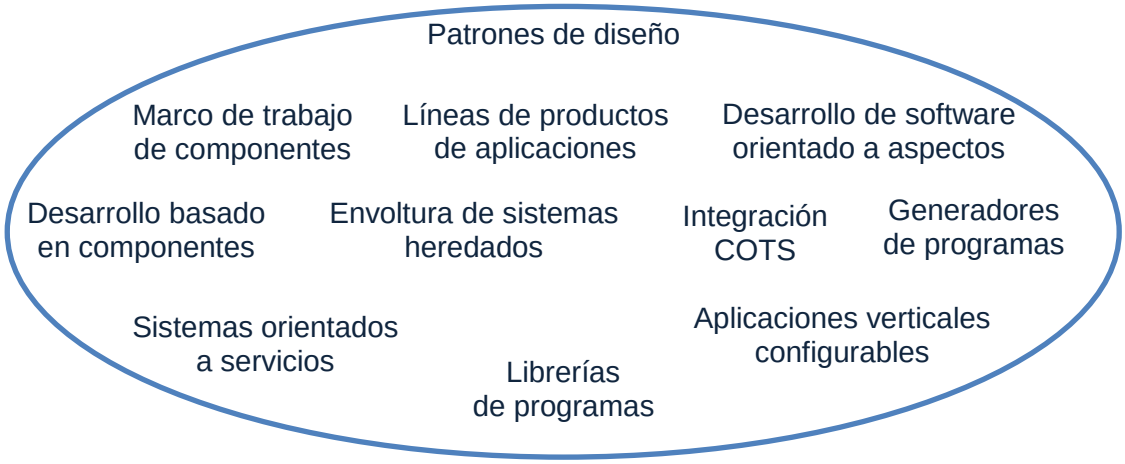
La cantidad de sistemas de reutilización disponibles es tal, que en la mayoría de las situaciones, existe la posibilidad de reutilizar software. Pero si se consigue o no la reutilización es a menudo una cuestión de gestión más que una cuestión de técnica, ya que los gestores pueden ser reacios a comprometer sus requerimientos para permitir el uso de componentes reutilizables, o pueden decidir que el desarrollo de componentes originales podría ayudar a crear una base de activos software. Es posible que no comprendan los riesgos asociados con la reutilización tan bien como entienden los riesgos del desarrollo original. Sin embargo, si bien los riesgos de un nuevo desarrollo software pueden ser más elevados, algunos gestores pueden preferir los riesgos conocidos a los desconocidos[12]. Esto lleva a concluir que aunque existan y estén disponibles componentes reutilizables existen muchos factores como este que no permiten su uso y difusión.

1.3.1.4 El campo de la reutilización. Durante los veinte últimos años, se han desarrollado muchas técnicas para soportar la reutilización del software. Éstas explotan el hecho de que los sistemas del mismo dominio de aplicación son similares y tienen potencial para la reutilización. Esta reutilización es posible a diferentes niveles (desde funciones simples a aplicaciones completas), y los

[12] Ibid. p. 382

estándares para componentes reutilizables facilitan la reutilización. La figura 2, muestra varias formas de soportar la reutilización del software.

Figura 2. El campo de la reutilización[13]



En la siguiente Tabla, se describe brevemente las formas de soportar la reutilización del software.

Tabla 1. Aproximaciones que soportan la reutilización del software.

Aproximación	Descripción
Patrones de diseño	Las abstracciones genéricas similares entre aplicaciones se representan como patrones de diseño que muestran los objetos abstractos y concretos y sus interacciones.
Desarrollo basado en componentes	Los sistemas se desarrollan integrando componentes (colecciones de objetos) que cumplen los estándares de modelado de componentes.
Marcos de aplicaciones	Las colecciones de bases concretas y abstractas pueden adaptarse y extenderse para crear sistemas de aplicaciones.
Envoltura de sistemas heredados	Sistemas heredados que pueden ser envueltos definiendo un conjunto de interfaces y proporcionando acceso a estos sistemas heredados a través de estas interfaces.
Sistemas orientados a servicios	Los sistemas se desarrollan enlazando servicios compartidos, que pueden ser proporcionados de forma externa.

[13] Ibid. p. 383

Líneas de productos de aplicaciones	Un tipo de aplicación se generaliza alrededor de una arquitectura común para que pueda ser adaptada para diferentes dientes.
Integración COTS	Los sistemas se desarrollan integrando sistemas de aplicaciones existentes.
Aplicaciones verticales configurables	Un sistema genérico se diseña para que pueda configurarse para las necesidades de dientes de sistemas particulares.
Bibliotecas de programas	Están disponibles para reutilización las bibliotecas de funciones y de bases que implementan abstracciones comúnmente usadas.
Generadores de programas	Un sistema generador incluye conocimiento de un tipo de aplicación particular y puede generar sistemas o fragmentos de un sistema en ese dominio.
Desarrollo del software orientado a aspectos	Componentes compartidos entrelazados en una aplicación en diferentes lugares cuando se compila el programa.

Dada esta lista de técnicas para reutilización, la cuestión clave es preguntarse ¿cuál es la técnica más adecuada a utilizar? Esto depende de los requerimientos del sistema a desarrollar, la tecnología y activos reutilizables disponibles, y la experiencia del grupo de desarrollo [14]. Los factores clave que deberían considerarse a la hora de planificar la reutilización son:

1. **La agenda de desarrollo del software**. Si el software tiene que desarrollarse rápidamente, debería intentarse reutilizar sistemas comerciales en vez de componentes individuales. Éstos son activos reutilizables de grano grueso. Si bien el cumplimiento de los requerimientos puede no ser perfecto, esta aproximación minimiza la cantidad de desarrollo requerido.

2. **Vida esperada del software**. Si se está desarrollando un sistema de larga vida, habría que centrarse en la mantenibilidad del sistema. En esas circunstancias, no solamente debería pensarse en las posibilidades inmediatas de la reutilización, sino también en las implicaciones a largo plazo. Se tendrá que adaptar el sistema a nuevos requerimientos, lo cual probablemente signifique hacer cambios a los componentes y cómo éstos son utilizados.

3. **Los conocimientos, habilidades y experiencia del grupo de desarrollo**. Todas las tecnologías de reutilización son bastante complejas y se necesita bastante tiempo para comprenderlas y usarlas de forma efectiva. Sin embargo, si el grupo de desarrollo posee habilidades en un área particular, probablemente habría que centrarse en ella.

[14] Ibid. p. 382-384.

4. **La criticidad del software y sus requerimientos no funcionales**. Para un sistema crítico, se tiene que crear un caso de confiabilidad para el sistema. Esto es difícil si no se tiene acceso al código fuente del software. Si el software tiene requerimientos de rendimiento estrictos, puede ser imposible utilizar estrategias tales como reutilización a través de generadores de programas. Estos sistemas tienden a generar código relativamente ineficiente.

Además de estos factores, es importante tener en cuenta la arquitectura que se utiliza en la reutilización.

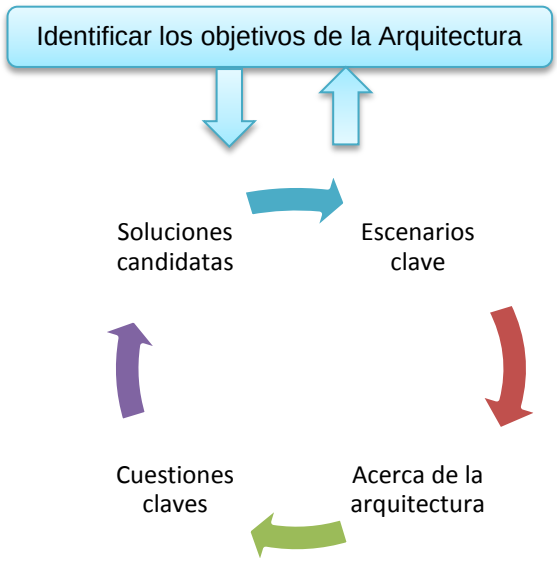
1.3.2 Arquitecturas diseñadas para la reutilización. La arquitectura de software juega un papel fundamental dentro del desarrollo de un sistema, ya que este proceso toma los requisitos de los usuarios, los analiza y produce un diseño para obtener un software que satisface necesidades.

Así para lograr tener un diseño exitoso de software este debe sopesar las complejidades inevitables que surgen debido a requisitos conflictivos. Además, una arquitectura contundente implica tener mucha experiencia en temas teóricos y prácticos, así como la visión necesaria para convertir lo que al parecer son escenarios y requisitos comerciales imprecisos en diseño de trabajo sólidos y prácticos.

En esta investigación, la arquitectura en principio implico una serie de decisiones basadas en una amplia gama de factores, y cada una de las decisiones debía tener un considerable impacto sobre la calidad, rendimiento, mantenimiento y éxito general de este sistema. Por lo que se optó por escoger la Arquitectura Orientada a Servicios, tema que se lleva a fondo más adelante.

De acuerdo a Microsoft Corp., el objetivo para escoger una arquitectura adecuada, es lograr que la arquitectura del sistema genere diseños candidatos que satisfagan los requisitos de alto nivel, el diseño después se revisa y se pone a prueba frente a los escenarios clave, teniendo así un ciclo iterativo que finalmente convergirá en los requisitos y escenarios clave. En la figura 3, se muestra este enfoque iterativo.

Figura 3. Proceso de diseño arquitectónico iterativo



1.3.2.1 Ciclo de desarrollo de la arquitectura. Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de desarrollo de la arquitectura de software. Este desarrollo, que precede a la construcción del sistema, está dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación.

- **Requerimientos**. La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura.

- **Diseño**. La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura, y no simplemente incorporar diversas tecnologías porque estas están de moda.

- **Documentación**. Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada.

- **Evaluación**. Dado que la arquitectura de software juega un papel crítico en el desarrollo, es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos. La ventaja de evaluar el diseño es que es una actividad que se puede realizar de manera temprana, y que el costo de corrección de los defectos identificados a través de

la evaluación es mucho menor al costo que tendría el corregir estos defectos una vez que el sistema ha sido construido.

En cuanto al ciclo de vida se debe también tener en cuenta que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo.

Además, cabe señalar que los sistemas orientados a servicios en la actualidad están teniendo gran importancia en el desarrollo de software óptimo, en esta medida la investigación toma como arquitectura, la arquitectura orientada a servicios.

1.3.3 Arquitectura Orientada a Servicios – SOA. El desarrollo de la Web trajo consigo que los computadores cliente tuviesen acceso a los servidores remotos situados fuera de sus propias organizaciones. Si estas organizaciones convertían su información a formato HTML, entonces ésta podía ser accedida por estos computadores. Sin embargo, el acceso se realizaba solamente a través de un navegador Web y el acceso directo a los almacenes de información por otros programas, no son prácticos. Esto implicaba que las conexiones oportunistas entre servidores no eran posibles.

Para solucionar este problema, se propuso la noción de un servicio Web. Un servicio Web es una representación estándar para cualquier recurso computacional o de información que pueda ser usado por otros programas. En la sección 1.3.3.3 se habla más a fondo de este tema.

La esencia de un servicio, por lo tanto, es que la provisión de servicio es independiente de la aplicación que usa el servicio. Los proveedores de servicios pueden desarrollar servicios especializados y ofertarlos a un cierto número de usuarios de servicios desde diferentes organizaciones.

1.3.3.1 ¿Qué es SOA?. La Arquitectura Orientada a Servicios – SOA, establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios[15]. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas[*] en un conjunto de servicios e implementar esta funcionalidad en forma modular.

La Arquitectura Orientada a Servicios (SOA) se define también como una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (IT), con las necesidades de negocio, permitiendo a los usuarios responder de forma más rápida y adaptarse adecuadamente a las presiones del mercado.

[15] MICROSOFT Corporation, <WWW.microsoft.com/soa> [Publicado: Diciembre 2006]
[*] Monolítico: referente a que presenta una gran cohesión, firmeza y solidez.

La Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. De acuerdo a esto la forma más habitual de implementarla es mediante la tecnología de los Servicios Web, con la que SOA puede descomponer aplicaciones compactas en un conjunto de servicios e implementar esta funcionalidad en forma modular. En la figura 4, se muestra las características de SOA.

Figura 4. Características de SOA

Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de los usuarios.

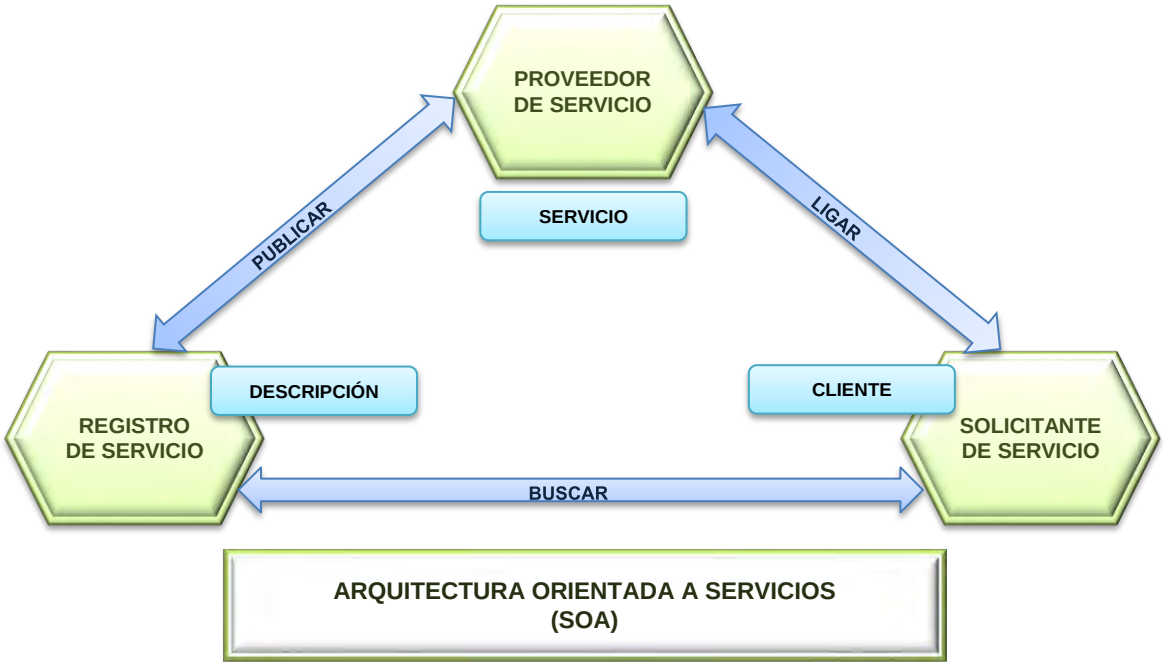
Aunque la adopción de una solución de diseño basada en SOA no exige implantar servicios Web, no obstante, como ya se comentó anteriormente, los servicios Web son la forma más habitual de implementar SOA.

1.3.3.2 Actores de SOA. La arquitectura orientada a servicios contiene tres actores.

- **Un solicitante de servicio**: es el responsable de encontrar una descripción de servicio publicada en uno o más registros de servicios, y de utilizar las descripciones de servicio para invocar los servicios Web hospedados por los proveedores de servicios.
- **Un proveedor de servicio**: es el responsable de crear una descripción de servicio, publicando la descripción en uno o más registros, y recibir mensajes de invocación de servicios Web de uno o más solicitantes de servicio.
- **Un registro de servicios**: es el responsable de anunciar descripciones de servicios Web publicados por los proveedores de servicio y permitir a los solicitantes buscar en la colección de descripción de servicios contenidos en el registro. Una vez encontrada la información, el resto de la interacción se da directamente entre el solicitante y el proveedor.

En la figura 5, se muestra los tres actores y su interacción dentro de la arquitectura orientada a servicios.

Figura 5. Arquitectura orientada a servicios (SOA)



Beneficios de SOA. SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones SOA permiten entre otras cosas:

- **Aplicaciones más productivas y flexibles**. La estrategia de orientación a servicios permite conseguir una mayor productividad de los recursos existentes como pueden ser las aplicaciones y sistemas ya instalados y obtener mayor valor de ellos sin necesidad de aplicar soluciones de integración desarrolladas. La orientación a servicios permite además el desarrollo de una nueva generación de aplicaciones compuestas que ofrecen capacidades avanzadas y con independencia de las plataformas y lenguajes de programación que soportan los procesos de base. Más aún: puesto que los servicios son entidades independientes de la infraestructura, una de sus características más importantes es su flexibilidad a la hora del diseño de cualquier solución.

- **Desarrollo de aplicaciones más rápido y económico**. El diseño de servicios basado en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades. Con ello se reduce el coste

del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo. Y el uso de un entorno y un modelo de desarrollo unificados simplifica y homogeneiza la creación de aplicaciones, desde su diseño y prueba hasta su puesta en marcha y mantenimiento.

- **Aplicaciones más seguras y manejables**. Las soluciones orientadas a servicios proporcionan una infraestructura común y una documentación común también para desarrollar servicios seguros, predecibles y gestionables. Conforme van evolucionando las necesidades, SOA facilita la posibilidad de añadir nuevos servicios y funcionalidades para gestionar los procesos de negocio críticos. Se accede a los servicios y no a las aplicaciones, y gracias a ello la arquitectura orientada a servicios optimiza las inversiones realizadas potenciando la capacidad de introducir nuevas capacidades y mejoras. Y además, puesto que se utilizan mecanismos de autenticación y autorización robustos en todos los servicios y puesto que los servicios existen de forma independiente unos de otros y no se interfieren entre ellos la estrategia de SOA permite dotarse de un nivel de seguridad superior.

1.3.3.3 Servicios Web. La alta conectividad entre computadores ha sido una meta desde que comenzó la informática personal. Con el auge de las redes internas dentro de las empresas llega el deseo de unir máquinas de forma programática. Es decir, un programa en una máquina debería poder llamar a métodos del programa en otra máquina sin la necesidad de intervención humana, para lo que se utilizaron diferentes tecnologías. Si se intenta centrar el estado actual del desarrollo de aplicaciones basadas en Web, se puede encontrar una gran cantidad de tecnologías, muchas de ellas incompatibles entre sí.

Se encuentra también que como la Internet se ha convertido en una herramienta de trabajo habitual, actualmente no es más que una fuente de datos y no de servicios dirigidos a facilitar el trabajo del usuario.

En este sentido, los servicios que ofrecen las nuevas tecnologías deberían cooperar para beneficio de los usuarios. Los sitios Web aislados y los diferentes dispositivos deberían trabajar juntos para ofrecer soluciones mucho más valiosas. Se trata de ofrecer a través de Internet no sólo datos, sino también software y servicios que puedan ser fácilmente accesibles, servicios que integren y busquen la información que se necesita, pudiendo acceder a esta información en cualquier momento y desde cualquier dispositivo.

Un concepto clave para solucionar estos problemas es la implementación de servicios Web como los que ofrece SISNOVA. En la figura 6, se presenta el proceso de los servicios Web.

Figura 6. Proceso de servicios Web

¿Qué es un servicio Web?

Un servicio Web es un componente de software independiente de plataforma e implementación, que lleva a cabo un servicio concreto y que puede integrarse con otros servicios Web para dar un servicio diferente. Los servicios Web se auto-describen y auto-exponen, permitiendo a los consumidores (aplicaciones clientes) localizarlos en Internet, para ser invocados y escuchados sobre protocolos estándar[16].

Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información que permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración.

Microsoft anunció por vez primera su modelo de servicios Web en septiembre de 1999, y a partir de ese momento se inició una corriente innovadora que ha transformado profundamente el panorama de la arquitectura de aplicaciones. Desde la aparición de la versión 1.0 de .NET Framework, las inversiones de Microsoft en herramientas y su alto nivel de compromiso con los servicios Web dentro de la plataforma Windows han contribuido al fuerte desarrollo actual de la orientación a servicios. Poco después Microsoft comenzó a colaborar con IBM para desarrollar la organización Web Services Interoperability Organization (WS-I), institución que promueve la interoperabilidad entre plataformas, sistemas operativos y lenguajes de programación. Actualmente en WS-I hay más de 150

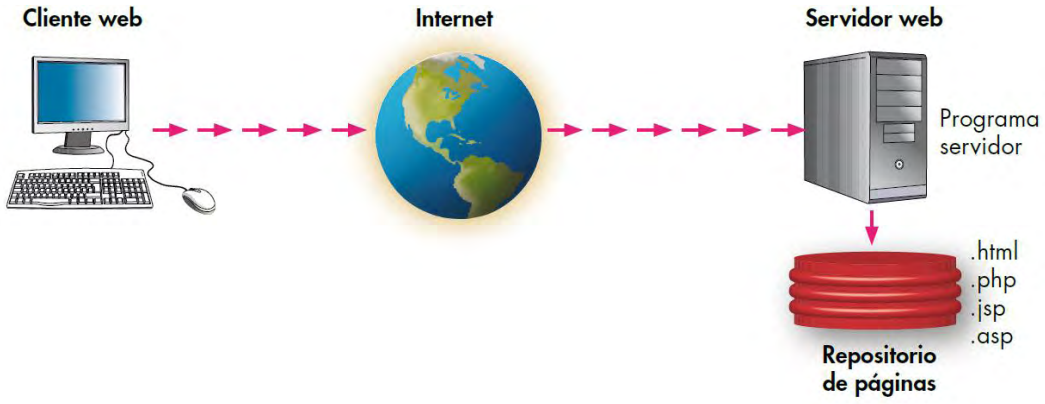
[16] SOLER, Pablo Peris, Servicios Web, Facultad de Informática, Universidad Politécnica de Valencia, 2012.

miembros, y ha creado servicios Web que resuelven distintas áreas críticas en aspectos como la interoperabilidad, seguridad y fiabilidad de la mensajería[17].

En otro concepto un servicio Web se trata de un recurso residente en la Web, con una dirección URL accesible y que devuelve información al cliente que quiera utilizarlo, pero los detalles de implementación y despliegue del servicio Web no son relevantes para el programa que invoca el servicio.

Normalmente los servicios Web no son para uso público o no están fácilmente accesibles. Son las aplicaciones las que acceden internamente para transmitir datos. Es por ello que la seguridad no suele ser muy buena ya que el programador piensa que sólo su aplicación va a consumir los servicios, pero realmente el servicio Web se encuentra en Internet esperando a que cualquiera conecte con él para solicitarle datos. En la figura 7, se muestra el funcionamiento básico de servicios Web.

Figura 7. Esquema básico de funcionamiento del servicio Web



El concepto de servicio Web se apoya en los estándares HTML y XML. El desarrollador puede crear programas accesibles desde cualquier dispositivo que soporte estos estándares, aprovechando la conectividad de Internet. Se pueden crear servicios accesibles desde Internet que realmente proporcionen una utilidad real.

Estándares de los servicios Web. La reutilización del software ha sido un tema de investigación durante muchos años, todavía quedan pendientes muchas dificultades prácticas. Uno de los principales problemas ha sido que los estándares para componentes reutilizables han sido desarrollados hace relativamente poco tiempo. Sin embargo, la iniciativa de los servicios Web ha estado guiada por estándares desde su nacimiento, y los estándares que incluyen muchos aspectos de los servicios Web están desarrollándose.

[17] MICROSOFT Corporation, <WWW.microsoft.com/> [Publicado: abril 2013]

Algunos estándares desarrollados por la W3C y otras organizaciones basados en XML pueden garantizar la confidencialidad, integridad y disponibilidad de los Servicios Web, estos protocolos proveen mecanismos para cifrar, firmar digitalmente, autenticar y certificar documentos XML.

En este sentido los estándares se han convertido en un elemento esencial para la integración de los servicios Web[18]. Los cuatro estándares fundamentales usados en el desarrollo de SISNOVA, que permiten la comunicación entre servicios Web son:

- **XML (Lenguaje de Etiquetado Extensible).** Inició en Febrero de 1998 y ha revolucionado la forma en la que se estructura, describe e intercambia información. Independientemente de múltiples formas en que utiliza hoy en día el XML todas las tecnologías de servicios Web se basan en XML.

- **SOAP (Protocolo de Acceso de Objeto Simple)**. Es un protocolo para iniciar las conversaciones con un servicio UDDI. El SOAP simplifica el acceso a los objetos, permitiendo a las aplicaciones invocar métodos objeto o funciones, que residen en sistemas remotos. Más adelante se detalla este protocolo.

- **WSDL (Lenguaje de Descripción de Servicios Web).** Es el estándar que define cómo pueden representarse las interfaces de servicio y sus características de implementación. El WSDL describe los mensajes SOAP que definen un servicio Web en particular.

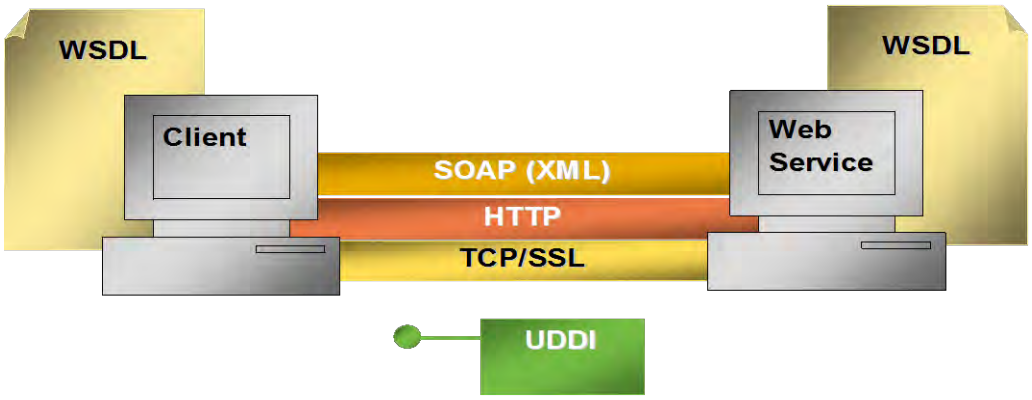
- **HTTP (Protocolo de Transferencia de Hipertexto)**. Es el protocolo usado en cada transacción de la WWW. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura Web para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL)[*].

Estos estándares se basan en XML y se muestran en la figura 8.

[18] ISAZA E, Gustavo A., Vector, Volumen 2, Diciembre 2007. p. 51–58.

[*]URL: localizador uniforme de recursos, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

Figura 8. Estándares de los servicios Web



Cabe resaltar que también se utiliza REST, (Transferencia de Estado Representacional), técnica de arquitectura software para sistemas hipermedia distribuidos como la WWW. El término se originó en el año 2000 de una tesis doctoral sobre la Web escrita[19]. Su autor es uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo. Más adelante se amplía este tema.

Ventajas de los servicios Web. A continuación se muestran las ventajas que conlleva el uso de los servicios Web en el desarrollo de las aplicaciones.

- **Interoperabilidad**. Nuevas relaciones pueden ser construidas dinámicamente y automáticamente ya que los servicios Web aseguran una interoperabilidad completa entre sistemas. Cualquier servicio Web puede interactuar con cualquier otro servicio Web o cliente, gracias a que la comunicación entre ambos se lleva a cabo en XML vía Internet (HTTP). Un servicio Web podrá estar escrito en cualquier plataforma o lenguaje que soporte estos estándares, lo cual no importará para su utilización o integración.

- **Fácil implementación**. Los conceptos en los que se basan los servicios Web son fácilmente entendibles y actualmente existen herramientas que permiten desarrollar y crear un servicio Web prácticamente teniendo sólo algunas nociones de programación.

- **Accesibilidad**. Los servicios Web pueden ser completamente descentralizados y distribuidos sobre Internet y accedidos a través de una gran variedad de dispositivos.

[19] FIELDING, Roy Thomas, Estilos arquitectónicos y el diseño de arquitecturas de software basados en la red, 2000, Doctor en Filosofía de la Información y Ciencias de la Computación.

- **Especificaciones universalmente aceptadas**. Los servicios Web se basan en especificaciones estándar para el intercambio de datos, mensajería, búsqueda, descripción de la interfaz y coordinación de los procesos.

- **Integración**. Los servicios Web proporcionan mayor agilidad y flexibilidad debido a una mejor integración con los sistemas existentes.

Operaciones de Servicios Web:

- **Publicar/Cancelar**. Los proveedores de servicios publican la disponibilidad de su servicio a uno o más registros de servicios, o cancelan la publicación de su servicio.

- **Búsqueda**. Los solicitantes de servicios interactúan con uno o más registros de servicios para descubrir un conjunto de servicios con los que pueden interactuar para encontrar una solución.

- **Ligar, Unir**. Los solicitantes de servicios negocian con los proveedores de servicios para acceder e invocar los servicios.

Protocolos de comunicación. Los protocolos de comunicación son una serie de normas que usan los computadores para gestionar sus diálogos en los intercambios de información. Dos equipos diferentes de marcas diferentes se pueden comunicar sin problemas en el caso en que usen el mismo protocolo de comunicaciones.

Se define al protocolo de comunicaciones como un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información por medio de cualquier tipo de variación de una magnitud física[20]. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos.

Como se sabe a lo largo del tiempo ha ido mejorando la tecnología de las comunicaciones, Así mismo han ido apareciendo protocolos más útiles para las nuevas máquinas.

Los protocolos utilizados en SISNOVA también son los más usados en la actualidad por los desarrolladores de servicios Web, estos son: SOAP y REST.

1.3.3.4 SOAP. (Protocolo de Acceso de Objeto Simple) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un

[20] RODRÍGUEZ - ARAGÓN, Licesio J., "T-, D- and c- Optimum Designs for BET and GAB Adsorption Isotherms", Chemometrics and Intelligent Laboratory Systems. 2007, Volumen: 89.

protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros, y está actualmente bajo el auspicio de la W3C. Es uno de los protocolos más utilizados en los servicios Web.

Características de SOAP. SOAP ofrece un framework de mensajería básica en la cual los servicios Web se puedan construir. Este protocolo basado en XML consiste de tres partes: un sobre, el cual define qué hay en el mensaje y cómo procesarlo; un conjunto de reglas de codificación para expresar instancias de tipos de datos; y una convención para representar llamadas a procedimientos y respuestas. El protocolo SOAP tiene tres características principales:

- **Extensibilidad**: la seguridad es una extensión aplicada en el desarrollo.
- **Neutralidad**: puede ser utilizado sobre cualquier protocolo de transporte.
- **Independencia**: SOAP permite cualquier modelo de programación.

Mensajes de SOAP. La especificación de SOAP define el formato de los mensajes para comunicar aplicaciones, normalmente dichos mensajes son una forma de comunicación de una única vía entre un emisor y un receptor (mensajes asincrónicos), sin embargo pueden ser combinados de manera de implementar patterns de requerimiento/respuesta (mensajes sincrónicos).

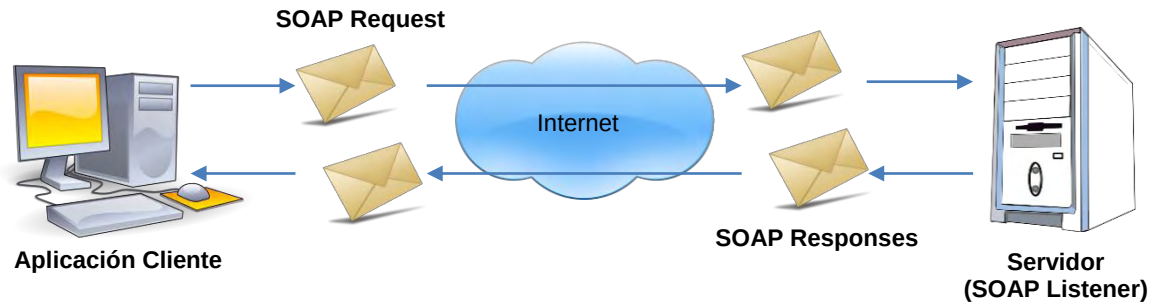
En la versión 1.0 de la especificación de SOAP se establecía como mandatorio el uso de HTTP como protocolo de transporte, sin embargo a partir de la versión 1.1 se desacopla SOAP del uso de HTTP, lo cual permite una mayor variedad de protocolos de transporte. En definitiva la especificación no establece un protocolo de transporte particular pero si específica cómo se realiza el transporte en caso de usar HTTP.

Arquitectura básica de SOAP. Desde el punto de vista de la presente investigación interesa usar SOAP para comunicar aplicaciones, realizando invocaciones RPC e implementando el pattern requerimiento/respuesta, por lo cual el comportamiento es similar al de una arquitectura cliente-servidor, donde el proceso es iniciado por el cliente y el servidor responde al requerimiento del mismo.

Este tipo de comunicación se basa en un sistema de mensajes SOAP sincrónicos, codificados en XML que son transportados por HTTP.

En la figura 9, se observa la arquitectura básica de un sistema, análogo al descripto, construido sobre SOAP y los mensajes que definen la interacción entre la aplicación cliente y la aplicación servidor. Generalmente, la aplicación cliente envía un mensaje (Request vía HTTP), el cual al ser recibido por la aplicación servidor genera una respuesta (Response) que es enviada a la aplicación cliente vía HTTP.

Figura 9. Arquitectura básica de un sistema construido sobre SOAP



En definitiva en el entorno de sistemas distribuidos funcionando sobre Internet, las tecnologías existentes presentan serias limitaciones debido a la heterogeneidad del entorno, la presencia de firewalls y al uso de formatos propietarios para representar datos. Bajo estas condiciones SOAP es una alternativa sumamente útil para comunicar aplicaciones heterogéneas (interoperabilidad), fundamentalmente por el uso de XML que es un estándar basado en texto e independiente de plataformas y lenguajes y por el uso de HTTP como transporte, el cual normalmente atraviesa los firewalls dado que estos no bloquean el puerto HTTP. Además al no ser un protocolo sofisticado y al no definir modelos de programación permite que las aplicaciones tengan un bajo acoplamiento, lo cual es ideal en el entorno de Internet.

De todas maneras puede verse que SOAP no es una tecnología que reemplace a las existentes sino una tecnología que permite la interoperabilidad de aplicaciones heterogéneas.

1.3.3.5 REST. (Transferencia de Estado Representacional) es un estilo de arquitectura software especialmente pensada para sistemas distribuidos, como la WWW. En los últimos años se ha popularizado un estilo de arquitectura software conocido como REST. Este nuevo estilo ha supuesto una nueva opción de estilo de uso de los servicios Web. Se basa en que el cliente que accede a una aplicación Web va cambiando su estado en función de los enlaces que se van eligiendo. Este enfoque plantea una aplicación Web como una máquina de estados virtual, dónde las transiciones entre los mismos son hiperenlaces.

Entre otras definiciones para REST, la refieren estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y divididos. El término frecuentemente es utilizado en el sentido de describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como hace SOAP.

Cabe destacar que REST no es un estándar, ya que es tan solo un estilo de arquitectura. Pero aunque REST no es un estándar, está basado en los siguientes estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG
- Tipos MIME: text/xml, text/html

REST es un estilo de arquitectura para el diseño de aplicaciones en red. La idea es que, en lugar de utilizar los mecanismos complejos, tales como RPC o SOAP para la conexión entre máquinas, se utiliza HTTP para hacer llamadas entre las máquinas. En muchos sentidos, la WWW en sí, basado en HTTP, se puede ver como una arquitectura basada en REST.

Principios de REST

El estilo de arquitectura subyacente a la Web es el modelo REST. Los objetivos de este estilo de arquitectura se listan a continuación:

- **Escalabilidad de la interacción con los componentes**. La Web ha crecido exponencialmente; una prueba de ellos es la variedad de clientes que pueden acceder a través de ella tales como: estaciones de trabajo, sistemas industriales, dispositivos móviles, etc.

- **Generalidad de interfaces**. Gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. Esto no es del todo cierto para otras alternativas, como SOAP para los servicios Web.

- **Puesta en funcionamiento independiente**. Este hecho es una realidad que debe tratarse cuando se trabaja en Internet. Los clientes y servidores pueden ser puestas en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URIs[*], a través de la habilidad para crear nuevos métodos y tipos de contenido.

- **Compatibilidad con componentes intermedios.** Los más populares intermediaros son varios tipos de proxys para la Web. Algunos de ellos, la cache, se utilizan para mejorar el rendimiento. Otros permiten reforzar las

[*]URI: identificador uniforme de recursos, que identifica un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.)

políticas de seguridad: firewalls. Y por último, otro tipo importante de intermediarios, Gateway, permiten encapsular sistemas no propiamente Web. Por tanto, la compatibilidad con intermediarios que permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas.

REST logra satisfacer estos objetivos aplicando las siguientes restricciones:

- **Identificación de recursos y manipulación de ellos a través de representaciones**. Esto se consigue mediante el uso de URIs. HTTP es un protocolo centrado en URIs. Los recursos son los objetos lógicos a los que se le envían mensajes. Los recursos no pueden ser directamente accedidos o modificados, más bien se trabaja con representaciones de ellos. Internamente el estado del recurso puede ser cualquier cosa desde una base de datos relacional a un fichero de texto.

- **Mensajes auto descriptivos**. REST establece que los mensajes HTTP deberían ser tan descriptivos como sea posible. Esto hace posible que los intermediarios interpreten los mensajes y ejecuten servicios en nombre del usuario. Uno de los modos que HTTP logra esto por medio del uso de varios métodos estándares, muchos encabezamientos y un mecanismo de direccionamiento.

- **Hipermedia como un mecanismo del estado de la aplicación**. El estado actual de una aplicación Web debería ser capturada en uno o más documentos de hipertexto, residiendo tanto en el cliente como en el servidor. El servidor conoce sobre el estado de sus recursos, aunque no intenta seguirle la pista a las sesiones individuales de los clientes. Esta es la misión del navegador, saber cómo navegar de recurso a recurso, recogiendo información que el necesita o cambiar el estado que el necesita cambiar.

En la actualidad muchos sitios Web comprometen uno más de estos principios, como por ejemplo, seguir la pista de los usuarios moviéndose a través de un sitio. Esto es posible dentro de la infraestructura de la Web, pero daña la escalabilidad, volviendo un medio sin conexión en todo lo contrario. Utilizando la analogía de la carta.

"Supongamos que usted se dispone a enviar una carta utilizando el esquema tradicional utilizando un sobre, en este caso es SOAP, pero si prefiere no usar un sobre y enviar una POSTAL estaría utilizando REST. Las postales son más fáciles de manejar para el receptor, no desperdicia papel (consume menos ancho de banda) y tiene un contenido corto (tampoco está limitado en longitud)"[21]

[21] CATALANI, Exequiel A., REST vs SOAP, <http://exequielc.wordpress.com/> [Publicado: 12 de octubre de 2012]

Estándares de la comunicación. Los estándares son de vital importancia en las comunicaciones. Establecer un lenguaje común permite que múltiples sistemas desarrollados independientemente por distintos fabricantes puedan interoperar,

Decía con ironía el catedrático de ciencias de computación A. S. Tanenbaum[22] en uno de sus libros que lo mejor de los estándares es que haya tantos para elegir. Obviamente, no es bueno que existan muchas normas para un mismo propósito, ya que la multiplicidad de estándares lleva a la incompatibilidad de los sistemas, a la fragmentación del mercado y a una mayor complejidad técnica en equipos con soporte multiestándar. En el caso de SISNOVA entre los estándares de comunicación utilizados están XML y JSON.

1.3.3.6 XML. Es un Lenguaje de Etiquetado Extensible, muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. Entre las tecnologías XML disponibles se pueden destacar:

1. **XSL**: Lenguaje Extensible de Hojas de Estilo, cuyo objetivo principal es mostrar cómo debería estar estructurado el contenido, cómo debería ser diseñado el contenido de origen y cómo debería ser paginado en un medio de presentación como puede ser una ventana de un navegador Web o un dispositivo móvil.

2. **XPath**: Lenguaje de Rutas XML, es un lenguaje para acceder a partes de un documento XML.

3. **XLink**: Lenguaje de Enlace XML, es un lenguaje que permite insertar elementos en documentos XML para crear enlaces entre recursos XML.

4. **XPointer**: Lenguaje de Direccionamiento XML, que permite el acceso a la estructura interna de un documento XML, a elementos, atributos y contenido.

5. **XQL**: Lenguaje de Consulta XML, es un lenguaje que facilita la extracción de datos desde documentos XML. Ofrece la posibilidad de realizar consultas flexibles para extraer datos de documentos XML en la Web.

[22] TANENBAUM, Andrew S., Redes de computadoras, 4º edición, Pearson Educación, México, 2003. p. 71-77, ISBN: 970-26-0162.

XML no ha nacido sólo para su aplicación para Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

1.3.3.7 JSON. Es un formato alternativo de envío y recepción de datos, es decir remplaza a XML o el envío de texto plano. Este formato de datos es más liviano que XML. Hace el código más sencillo ya que utiliza el código JavaScript como modelo de datos.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON.

Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.

JSON se basa en dos estructuras:

1. Una colección de pares nombre/valor. En varios idiomas esto se realiza como un objeto, registro, estructura, diccionario, o una matriz asociativa.

2. Una lista ordenada de valores. En la mayoría de los idiomas, esto se realiza en forma de matriz, vector, lista o secuencia.

Se trata de estructuras de datos universales. Prácticamente todos los lenguajes de programación modernos las acogen de una forma u otra. Tiene sentido que un formato de datos que es intercambiable con los lenguajes de programación también se basa en estas estructuras.

Para concluir, JSON es empleado con frecuencia en entornos Web donde el flujo de los datos es alto entre el cliente y el servidor, y donde los tiempos de procesamiento de los datos son de vital importancia. JSON optimiza los tiempos de respuesta debido a su leve peso. Y aunque es frecuente ver a JSON enfrentado contra XML, no deben ser excluyentes ya que incluso pueden ser complementarios el uso de JSON y XML en la misma aplicación.

1.3.4 Ingeniería de Software Basada en Componentes – ISBC. El desarrollo de software basado en componentes permite reutilizar piezas de código pre elaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.

La ISBC es un acercamiento basado en la reutilización para definir, implementar, y componer, componentes débilmente acoplados en sistemas.

Los fundamentos de la ingeniería del software basada en componentes son:

1. **Componentes independientes**, que son completamente especificados por sus interfaces. Debería haber una clara separación entre la interfaz de los componentes y su implementación para que una implementación de un componente pueda reemplazar por otro sin cambiar el sistema.

2. **Estándares de componentes**, que facilitan la integración de los componentes. Estos estándares se incluyen en un modelo de componentes y definen, en el nivel más bajo cómo las interfaces de componentes deberían especificarse y cómo se comunican los componentes. Si los componentes cumplen con los estándares, entonces su funcionamiento es independiente de su lenguaje de programación.

3. **El middleware**, que proporciona soportes software para la integración de componentes. Para conseguir que componentes independientes y distribuidos trabajen juntos, necesita un soporte middleware que maneje las comunicaciones de los componente. Un producto middleware utilizado para implementar un modelo de componentes puede proporcionar soporte para asignación de recursos, gestión de transacciones, seguridad y concurrencia.

4. **Un proceso de desarrollo**, que se adapta a la ingeniería del software basada en componentes. Si se intenta añadir una aproximación basada en componentes a un proceso de desarrollo que está adaptado a la producción de software original.

El desarrollo basado en componentes se está adoptando cada vez más como una aproximación fundamental a la ingeniería del software incluso aunque los componentes reutilizables no estén disponibles. La ISBC está asentada sobre unos principios de diseño sólidos que soportan la construcción de software comprensible y mantenible. Los componentes son independientes para que no interfieran en su funcionamiento unos con otros. Los detalles de implementación se ocultan, por lo que la implementación de los componentes puede cambiarse sin afectar al resto del sistema.

En SISNOVA un componente reutilizable es el ingrediente de las aplicaciones que se juntan y combinan para llevar a cabo una tarea. Es algo muy similar a lo que se puede observar en un equipo de música que hay en alguna sala. Cada componente de aquel aparato ha sido diseñado para acoplarse perfectamente sus partes, las conexiones son estándar y el protocolo de comunicación está ya preestablecido. Al unirse las partes, se obtiene música para los oídos.

El paradigma de ensamblar componentes y escribir código para hacer que estos componentes funcionen se conoce como Desarrollo de Software Basado en Componentes.

1.3.4.1 Ventajas de la ISBC. El uso de este paradigma posee algunas ventajas:

- **Reutilización del software**, el desarrollo de software basado en componentes es un acercamiento basado en la reutilización para definir, implementar, y componer, componentes débilmente acoplados en los sistemas.

- **Simplifica las pruebas**, las pruebas son ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes.

- **Simplifica el mantenimiento del sistema**, el desarrollador es libre de actualizar o agregar componentes según sea necesario sin afectar otras partes del sistema, esto es posible cuando existe un débil acoplamiento entre componentes.

- **Mayor calidad**: la calidad de una aplicación basada en componentes mejora con el paso del tiempo, dado que un componente puede ser construido y mejorado continuamente.

1.3.4.2 Componentes y modelos de componentes. Hay un consenso general en la comunidad, de que un componente es una unidad de software independiente que puede estar compuesta por otros componentes y que se utiliza para crear un sistema software[23].

Lo que tiene en común la comunidad, es que están de acuerdo en que los componentes son independientes y que son unidades de composición fundamentales en un sistema.

Estas definiciones formales de componentes son bastante abstractas y no proporcionan realmente una idea clara de lo que hace un componente. Una de las formas más útiles de considerar un componente es como un proveedor de servicios independiente. Cuando un sistema necesita algún servicio, llama a un componente para proporcionar dicho servicio sin tener en cuenta dónde se está ejecutando el componente o qué lenguaje se ha utilizado para desarrollar el componente. Por ejemplo, un componente en un sistema de biblioteca proporciona un servicio de búsqueda que permite a los usuarios buscar diferentes catálogos de bibliotecas; un componente que convierta de un formato gráfico a otro (por ejemplo. TIFF a JPEG) proporciona un servicio de conversión de datos.

[23] Ibid. SOMERVILLE, Ingeniería de Software, p. 404.

Para entender entonces la funcionalidad de componentes reutilizables se debe comprender lo que está ocurriendo en este momento, por ejemplo si se habla de transporte y comunicación, a mediados del siglo XIX llegó el ferrocarril. Se hicieron enormes cantidades de dinero moviendo personas, carbón y trigo de un lugar a otro.

Con una demanda arrolladora de transporte, eventualmente todo Estados Unidos estaba interconectado por ferrocarril. No solo la gente empezó a viajar a nuevos lugares para conocer nuevas culturas, sino que ahora los comerciantes podían vender artículos de formas nunca antes pensadas. Pero más importante aún, es que el movimiento de los artículos despertó la expectativa de que las cosas funcionen en conjunto. Antes del ferrocarril simplemente no importaba si los bienes de un fabricante eran incompatibles con los de otro fabricante.

En este sentido, desde el punto de vista de los autores del presente trabajo, el desarrollo de software basado en componentes desde siempre fue la idea revolucionaria que llevó a pensar que sí era posible el construir software de calidad en corto tiempo y con la misma calidad que la mayoría de las industrias actuales, convirtiéndose así en el pilar de la Revolución Industrial del Software y se proyecta hoy en día en diversas nuevas formas de hacer software de calidad con los costos más bajos del mercado y en tiempos que antes eran impensables.

1.3.4.3 Controles Windows Forms. Los controles son objetos contenidos en formularios. Cada tipo de control tiene su propio conjunto de propiedades, métodos y eventos que lo hacen adecuado para un propósito en particular. Se puede manipular los controles en el diseñador y escribir código para agregar controles dinámicamente, en tiempo de ejecución. Cuando se diseñe y modifique la interfaz de usuario de aplicaciones de Windows Forms, se tendrá que agregar, alinear y situar controles.

Para crear controles, .NET Framework ofrece abundante tecnología para su creación. Los creadores ya no están limitados al diseño de controles compuestos que funcionan como una colección de controles preexistentes. A través de la herencia, se puede crean controles propios a partir de controles compuestos preexistentes o controles de formularios Windows Forms preexistentes. También se puede diseñar controles propios que implementen el dibujo personalizado. Estas opciones permiten una gran flexibilidad en el diseño y funcionalidad de la interfaz visual.

1.3.5 Metodologías para el desarrollo de software reutilizable. Actualmente, existen numerosas propuestas de desarrollo de software que incluyen implícitamente la reutilización, otras propuestas de desarrollo de software, no incluyen implícitamente la reutilización, sin embargo, proporcionan la flexibilidad requerida para implementarlas.

1.3.5.1 Ingeniería de software. La Ingeniería del Software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza[24]. En esta definición, existen dos frases clave:

1. **Disciplina de la ingeniería**. Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde sean convenientes, pero las utilizan de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlos. Los ingenieros también saben que deben trabajar con restricciones financieras y organizacionales, por lo que buscan soluciones tomando en cuenta estas restricciones.

2. **Todos los aspectos de producción de software**. La ingeniería del software no sólo comprende los procesos técnicos del desarrollo de software, sino también, con actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

En general, los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más efectiva de producir software de alta calidad. Sin embargo, aunque la ingeniería consiste en seleccionar el método más apropiado para un conjunto de circunstancias, un enfoque más informal y creativo de desarrollo podría ser efectivo en algunas circunstancias. El desarrollo informal es apropiado para el desarrollo de sistemas basados en Web, los cuales requieren una mezcla de técnicas de software y de diseño gráfico.

Proceso de software. Un proceso del software es un conjunto de actividades y resultados asociados que producen un producto de software. Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software. Estas actividades son:

1. Especificación del software donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
2. Desarrollo del software donde el software se diseña y programa.
3. Validación del software donde el software se válida para asegurar que es lo que el cliente requiere.
4. Evolución del software donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

El proceso general de Ingeniería de Software es seguido en la mayoría de los desarrollo de software, incluso esta investigación sigue este proceso que tiene las

[24] Ibid. SOMERVILLE, Ingeniería de Software, p. 6.

siguientes fase que guían el proceso de desarrollo del sistema de un manera organizada y estructurada.

1 **Ingeniería de requisitos**: por lo general los clientes piensan que ellos saben lo que el software tiene que hacer, pero lo cierto es que se necesita de un proceso de descubrimiento, refinamiento, modelado y especificación, donde se depuran en detalle los requisitos del sistema y el papel asignado al software, para esta investigación se toman requisitos iniciales y básicos para que den solución a lo propuesto y puedan evolucionar de tal manera que se ajusten a las necesidades de los usuarios finales.

2 **Análisis**: se comprende de forma detallada cual es el problema a resolver, obteniendo así toda la información necesaria y suficiente para afrontar la solución propuesta por el sistema.

3 **Diseño**: ya con la información suficiente del problema a solucionar, se determina la estrategia que está enfocada en el uso de los sistemas orientados a los servicios Web y en la reutilización para así la resolver el problema.

4 **Implementación**: mediante el uso de las diferentes herramientas y el conocimiento de los temas necesarios se procede al desarrollo del sistema el cual da solución a los problemas planteados y se verifica su funcionalidad en diferentes plataformas.

5 **Pruebas**: para garantizar el correcto funcionamiento del sistema se realizan las pruebas bajo el mayor número de situaciones posibles a las que se pueda enfrentar tanto el servicio Web como las aplicaciones que dan resolución a los problemas de transportes.

6 **Documentación**: dentro de esta investigación se tomaron muchos temas que son de gran impacto y evolución en la actualidad, por lo cual es importante documentar cada proceso del desarrollo del sistema, para que además pueda ser modificado y sus funcionalidades también evolucionen y den solución a todos los problemas relacionados con él.

Un proceso de Ingeniería de Software requiere herramientas que apoyen todas las actividades del ciclo de vida de los sistemas. Como respuesta a ello Rational Software Corp. puso a disposición herramientas que están diseñadas para hacer más eficiente la aplicación de metodologías como el RUP. De esa forma se permite la creación de un ambiente de desarrollo óptimo.

Además junto con UML constituyen la metodología estándar para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

1.3.5.2 RUP (Proceso Unificado de Rational). El Proceso Unificado de Rational (RUP) es un ejemplo de un modelo de proceso moderno que proviene del trabajo en el UML y el asociado Proceso Unificado de Desarrollo de Software[25]. RUP fue utilizado en esta investigación dentro de la metodología para llevar a cabo el desarrollo de SISNOVA.

El RUP reconoce que los modelos de procesos genéricos presentan un sola enfoque del proceso. En contraste, el RUP se describe normalmente desde tres perspectivas:

1. Una perspectiva dinámica que muestra las fases del modelo sobre el tiempo.
2. Una perspectiva estática que muestra las actividades del proceso que se representan.
3. Una perspectiva práctica que sugiere buenas prácticas a utilizar durante el proceso.

La mayor parte de las descripciones del RUP intentan combinar las perspectivas estática y dinámica en un único diagrama (Krutchen). Esto hace el proceso más difícil de entender, por lo que aquí se utilizan descripciones separadas de cada una de estas perspectivas.

El RUP es un modelo en fases, que identifica cuatro fases diferentes en el proceso del software. Estas fases están mucho más relacionadas con asuntos de negocio más que técnicos. Las fases en el RUP son:

1. **Inicio**. El objetivo de la fase de inicio es el de establecer un caso de negocio para el sistema. Se deben identificar todas las entidades externas que interactuarán con el sistema y definir estas interacciones. Si esta aportación es de poca importancia, se puede cancelar el proyecto después de esta fase.

2. **Elaboración**. Los objetivos de la fase de elaboración son desarrollar una comprensión del dominio del problema, establecer un marco de trabajo arquitectónico para el sistema, desarrollar el plan del proyecto e identificar los riesgos clave del proyecto. Al terminar esta fase, se debe tener un modelo de los requerimientos del sistema, una descripción arquitectónica y un plan de desarrollo del software.

3. **Construcción**. La fase de construcción fundamentalmente comprende el diseño del sistema, la programación y las pruebas. Durante esta fase se desarrollan e integran las partes del sistema. Al terminar esta fase, debe tener un sistema software operativo y la documentación correspondiente lista para entregarla a los usuarios.

[25] RUMBAUGH, James, “The Rational Unified Process an Introduction”, 3º edición, Pearson Education, Inc., Boston, 2004. ISBN 0-321-19770-4.

4. **Transición**. La fase final del RUP se ocupa de mover el sistema desde la comunidad de desarrollo a la comunidad del usuario y hacerlo trabajar en un entorno real. Esto se deja de lado en la mayor parte de los modelos de procesos del software ya que es, en realidad, una actividad de alto costo y a veces problemática. Al terminar esta fase, se debe tener un sistema software documentado que funciona correctamente en su entorno operativo.

El Proceso Unificado de Rational fue escogido en esta investigación porque es una metodología que permite construir software adaptable al contexto y a las necesidades expuestas.

Además, el conjunto de herramientas de Rational está basado en seis principios fundamentales para el desarrollo de software que son: administración de los requerimientos, desarrollar en forma iterativa, modelar visualmente, pruebas continuas, arquitecturas basadas en componentes y control de cambios[26].

Principios que se adaptan al manejo de la investigación que se lleva a cabo.

1. **Administrar los requerimientos**: el desafío de administrar los requerimientos de un sistema de software es que son dinámicos, es decir, pueden cambiar durante la vida de un proyecto. Por tanto identificar los verdaderos requerimientos de un sistema debe hacerse de tal manera que logre satisfacer tanto objetivos económicos como técnicos en un proceso continuo.

2. **Desarrollar software iterativamente**: un enfoque iterativo facilita la implementación de nuevos cambios tácticos de requerimientos y características del cronograma. Con este enfoque se fortalece la identificación temprana de los riesgos del proyecto.

3. **Modelar software visualmente**: hacer modelos es importante, ya que permite visualizar, especificar, construir y documentar la estructura y comportamiento de la arquitectura de un sistema, si además se utiliza un lenguaje de modelamiento estándar se mejora la capacidad del equipo de trabajo para administrar la complejidad del software.

4. **Pruebas continuas**: la revisión continua permite encontrar las fallas antes de la puesta en producción de un sistema. Si además se desarrolla el software iterativamente, se está revisando la versión en cada iteración, logrando así un proceso de evaluación continuo y cuantitativo.

5. **Utilizar arquitecturas basadas en componentes**: el desarrollo basado en componentes que es el enfoque que se utiliza en esta investigación permite la reutilización o adaptación de componentes existentes y disponibles.

[26] Ibid

6. **Control de cambios**: la coordinación de las actividades y los productos que se van generando implica administrar los cambios al software, este control permite una mejor asignación de los recursos.

Lenguaje de modelamiento unificado (UML). UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Es el lenguaje de modelamiento de sistemas de software más conocido y utilizado en la actualidad que ofrece un estándar para describir el modelo de un sistema,

Con UML, se puede definir diagramas propios como también poder entender el modelamiento de los diagramas que se crean en el sistema. Así mismo, se han definido criterios para la extensión de esta metodología a otros niveles de abstracción tales como sistemas basados en componentes.

1.3.5.3 Programación orientada a objetos. Se supone que el desarrollo actual se basa en el uso de clases, interfaces, propiedades, métodos, etc., sin embargo, no se le da la importancia debida, ya que a menudo se encuentran métodos implementados con cientos de líneas de código, mezclando diferentes tipos de implementaciones: lógica de negocio, transacciones, acceso a datos, entre otras cosas.

El diseño orientado a objetos y la programación orientada a objetos representan un cambio de perspectiva con respecto a la programación estándar por procedimientos. En lugar de pensar en el flujo del programa desde la primera hasta la última línea de código, se debe pensar en la creación de objetos: componentes independientes de una aplicación que tienen funcionalidad privada además de la funcionalidad que se puede exponer al usuario[27].

La programación orientada objetos (POO) ayuda a diseñar adecuadamente una aplicación, no es el hecho de implementar un número considerable de clases en una aplicación sin una responsabilidad clara. Se tiene que partir de un diseño que permita dar un fácil mantenimiento al código, que sea legible y de fácil entendimiento, y lo más importante, procurar desacoplar las responsabilidades de la aplicación para lograr una escalabilidad más transparente e incluso incorporar funcionalidades con un menor impacto en tiempo y costos.

Fundamentos. La POO es una técnica para desarrollar soluciones computacionales utilizando componentes de software.

[27] MICROSOFT, Programación orientada a objetos, <http://msdn.microsoft.com/es-es/library/> 2013.

- **Objeto**: Componente o código de software que contiene en sí mismo tanto sus características (campos) como sus comportamientos (métodos); se accede a través de su interfaz o signatura.
- **Campo**: Es una característica de un objeto, que ayuda a definir su estructura y permite diferenciarlo de otros objetos. Se define con un identificador y un tipo, el cual indica los valores que puede almacenar. El conjunto de valores de los campos definen el estado del objeto.
- **Método**: Es la implementación de un algoritmo que representa una operación o función que un objeto realiza. El conjunto de los métodos de un objeto determinan el comportamiento del objeto.

La POO se basa en el modelo objeto donde el elemento principal es el objeto, el cual es una unidad que contiene todas sus características y comportamientos en sí misma, lo cual lo hace como un todo independiente pero que se interrelaciona con objetos de su misma clase o de otras clase, como sucede en el mundo real.

Modularidad en POO. La modularidad es el proceso de crear partes de un todo que se integran perfectamente entre sí para que funcionen por un objetivo general, y a las cuales se les pueden agregar más componentes que se acoplen perfectamente al todo, o extraerle componentes sin afectar su funcionamiento. En el caso que se requiera actualizar un módulo, no hay necesidad de hacer cambios en otras partes del todo. Un ejemplo clásico es un conjunto de módulos que, al integrarlos conforman un armario, el cual puede agregarle más funcionalidad si se le agregan más módulos, o al contrario. También se puede cambiar su finalidad si se acomodan esos módulos para darle otro objetivo: volverlo una mesa por ejemplo.

Esto ayuda a la descomposición de problemas en subproblemas, es decir, a la solución de problemas por composición de soluciones a subproblemas.

Reutilización en POO. Para reutilizar código se crean nuevas clases pero, en lugar de partir de cero, se parte de clases, relacionadas con clases propias, que han sido ya creadas y depuradas. Lo ideal es usar las clases sin “ensuciar” el código existente.

Una forma de hacer esto es crear objetos de clases existentes dentro de la nueva clase. Esto se conoce como composición porque la nueva clase está compuesta de objetos de clases existentes. Se está entonces, reutilizando la funcionalidad del código, y no la forma.

Otra forma es crear una nueva clase como un tipo de una clase ya existente. Se toma la forma de la clase existente y se añade código a la nueva, sin modificar la clase existente. Esta forma de crear nuevos objetos se llamada herencia, y lo que se hace es extender la clase en la que se basa para crear la nueva.

Cuando se hereda, se está cogiendo una clase existente y creando una versión especial de esa clase. En general, esto significa que se toma una clase de propósito general, especializándola para un caso o necesidad particular. Aunque escribir código para que sea reutilizable puede llevar un esfuerzo de planeación mayor que simplemente sentarse y codificar, a la larga acorta los tiempos de desarrollo y hace más fácil desarrollar aplicaciones confiables.

1.4 INVESTIGACIÓN DE OPERACIONES

Cada vez que un problema requiera para su solución de altos procesos manuales, complejos y repetitivos, se debe tener en cuenta la intervención de tecnologías que presentan alternativas de desarrollo. Por ejemplo si se tiene que calcular una integral y ya está el método para su solución, ese conocimiento es propicio que este en una máquina. El ser humano lo que debe hacer es dedicarse a generar nuevo conocimiento.

1.4.1 ¿Qué es la investigación de operaciones?. En la actualidad, las empresas deben de enfrentar problemas de todo tipo, las cuales en algunos casos pueden poner en riesgo, no sólo la estabilidad, sino también su permanencia en el mercado, por lo que deben de resolverlos en forma rápida y viable. Estos problemas pueden ser complejos, debido al número de variables y parámetros que se conozcan y por el nivel de certidumbre de información que se maneja. Para resolverlos, el ser humano crea modelos y aplica uno de los tres procesos de solución que existen: procesos algorítmicos, procesos heurísticos o la simulación.

La investigación de operaciones se puede definir como la aplicación del método científico en la solución de problemas, cuyo enfoque es la modelación, es decir, crea modelos para representar los problemas y utiliza diferentes técnicas, como la programación lineal y el análisis de decisiones, para establecer la solución del mismo[28].

Si bien es indispensable, para la persona que estudia la investigación de operacional, el estudiar los problemas generales que se presentan y los algoritmos clásicos que permiten resolverlos, debe estar también totalmente persuadido de que las situaciones prácticas que encontrará serán mucho más complicadas y que deberá emprender una tarea original para dar satisfacción al encargado de tomar decisiones ofreciéndole la posibilidad de optimizar según su propio criterio.

Es necesario, pues, en función de las motivaciones del responsable de la decisión, identificar los fenómenos a estudiar mediante un análisis profundo de la situación.

[28] TAHA Hamdy A., Investigación de Operaciones, 7ª Edición, University of Arkansas, Fayetteville. Person Educación, México. 2004. p. 1-3.

Las fases principales de la implementación de la investigación de operaciones en la práctica comprenden:

1. La definición del problema.
2. La construcción del modelo.
3. La solución del modelo.
4. La validación del modelo.
5. La implementación de la solución.

Aunque el ritmo de desarrollo de nuevas técnicas de la Investigación de Operaciones ha disminuido con el tiempo, ha aumentado las áreas donde se aplica, así como las magnitudes de los problemas que pueden ser resueltos con mayor facilidad con las metodologías de la Investigación de operaciones logrando que el sistema se optimo.

1.4.2 Optimización de sistemas. El análisis de un sistema normalmente se orienta a comprender su funcionamiento y la interrelación entre sus componentes. Sin embargo, bajo un enfoque más utilitario, el objetivo del modelo que describe el sistema podrá orientarse a determinar el manejo del sistema y sus variables con vistas a obtener las mejores salidas. Para este caso la optimización ayuda a encontrar la solución que brinda los mejores resultados, da la utilidad más alta o el resultado con el mínimo costo.

Así mismo para los sistemas informáticos es indispensable la optimización, por ejemplo en el aseguramiento de tiempos de respuesta aceptables para usuarios en línea en un ambiente operativo cuyos niveles de carga son impredecibles y rápidamente fluctuantes es uno de los requerimientos básicos de los sistemas de información modernos, particularmente los que dan servicio por medio de Internet, como en el caso de SISNOVA que presenta como una de sus funcionalidades un servicio Web. Donde lo importante aquí es partir desde el diseño de sistemas óptimos.

1.4.3 Método simplex. El método Simplex se ha convertido en una herramienta estándar de gran importancia para numerosas organizaciones comerciales e industriales. Además, casi cualquier organización social tiene que ver con la asignación de recursos en algún contexto y existe un reconocimiento creciente de la extremadamente amplia aplicabilidad de la técnica del método. Este no solo aporta la solución óptima de las variables sino, su utilidad máxima, y costo mínimo, sino también una gran cantidad de valiosa información económica. Por esta razón, resulta de vital importancia tanto el aprender las técnicas de solución a través de este método y además dominar ampliamente la interpretación de resultados.

En el año 1947, el doctor George Dantzig presentó el algoritmo que desarrolló y que denominó SIMPLEX. A partir de este logro se pudieron resolver problemas

que por más de un siglo permanecieron en calidad de estudio e investigación con modelos formulados pero no resueltos[29]. El desarrollo paralelo de la computación digital, hizo posible su rápido desarrollo y aplicación empresarial a todo tipo de problemas.

El método Simplex disminuye sistemáticamente un número infinito de soluciones hasta un número finito de soluciones básicas factibles. El algoritmo Simplex utiliza el conocido procedimiento de eliminación en la solución de ecuaciones lineales de Gauss- Jordan y, además aplica los llamados criterios del Simplex con los cuales se asegura mantener la búsqueda dentro de un conjunto de soluciones factibles al problema; así valora una función económica Z, exclusivamente en vértices factibles (posibles)[30].

Así entonces el Método Simplex, es un procedimiento iterativo que permite ir mejorando la solución a cada paso. El proceso concluye cuando no es posible seguir mejorando más dicha solución. Una propiedad general del método Simplex es que resuelve la programación lineal en iteraciones. Cada iteración desplaza la solución a un nuevo punto esquina que tiene potencial de mejorar el valor de la función objetivo. El proceso termina cuando ya no se pueden obtener mejoras.

Algoritmo del método simplex. Este proceso que se repite una y otra vez, siempre inicia en un punto extremo de la región factible que normalmente es el origen, en cada iteración se mueve a otro punto extremo adyacente hasta llegar a la solución óptima.

Los pasos del Método Simplex son los siguientes:

1. Utilizando la forma estándar, determinar una solución básica factible inicial igualando a las n-m variables igual a cero (el origen).
2. Seleccionar la variable de entrada de las variables no básicas que al incrementar su valor pueda mejorar el valor en la función objetivo. Cuando no exista esta situación, la solución actual es la óptima; si no, ir al siguiente paso.
3. Seleccionar la variable de salida de las variables básicas actuales.
4. Determinar la nueva solución al hacer la variable de entrada básica y la variable de salida no básica, ir al paso 2.

El método Simplex implica cálculos tediosos y voluminosos, lo que hace que sistemas computacionales sean una herramienta esencial para resolver los problemas de programación lineal. Por consiguiente, las reglas computacionales del método Simplex se adaptan para facilitar el cálculo automático.

---

[29] DANTZIG, George, “Linear Programming: Theory and Extensions”, Springer; Edición: 2003. ISBN 0387986138.

[30] TAHA, Hamdy A., Investigación de Operaciones, 7ª Edición, University of Arkansas, Fayetteville. Person Educación, México. 2004. p. 71.

Una gran parte de software para cálculos están estrictamente basados en el método Simplex, facilita la interpretación de datos en poco tiempo es decir que gracias a este método y a los programas que se basan en el ejercicios que se tardarían días en resolverse se llevan a cabo en tan solo horas y hasta minutos optimizando el trabajo de todo aquel que necesite realizar este tipo de cálculo.

1.4.4 Modelo de transportes. En los últimos años, el transporte ha ido cobrando cada vez una mayor importancia en la economía y en las decisiones administrativas, en donde se ha convertido en una actividad básica desde el punto de vista económico y social. Una de las principales funciones del transporte es la de poner en contacto a consumidores y productores, potenciando la especialización productiva y el acceso de los consumidores a variedad de productos cada vez mayor y de más calidad. Con frecuencia la disponibilidad de transporte económico es crítica para la sobrevivencia de una empresa.

El Modelo de Transportes es un caso especial simplificado del método Simplex, que determina la manera más óptima de transportar bienes. El problema general de transporte se refiere a la distribución de cualquier bien desde cualquier grupo de centros de suministro, llamados orígenes, a cualquier grupo de centros de recepción, llamados destinos, de tal manera que se minimicen los costos totales de distribución.

Los dos objetivos comunes de estos problemas, son:

- Minimizar el costo de enviar *n* unidades hasta *m* destinos o
- Maximizar las utilidades de enviar *n* unidades a *m* destinos.

Con un ejemplo supóngase que un fabricante tiene tres plantas que producen el mismo producto. Estas plantas a su vez mandan el producto a cuatro almacenes. Cada planta puede mandar productos a todos los almacenes, pero el costo de transporte varía con las diferentes combinaciones. El problema es determinar la cantidad que cada planta debe mandar a cada almacén con el fin de minimizar el costo total de transporte[31].

La manera más fácil de reconocer un problema de transporte es por su naturaleza o estructura: de un origen hacia un destino, de una fuente hacia un usuario, del presente hacia el futuro, de aquí hacia allá. Al enfrentar este tipo de problemas, la intuición dice que debe haber una manera de obtener una solución. Se conocen las fuentes y los destinos, las capacidades y demandas y los costos de cada trayectoria. Debe haber una combinación óptima que minimice el costo o maximice la ganancia. La dificultad consiste en el gran número de combinaciones posibles.

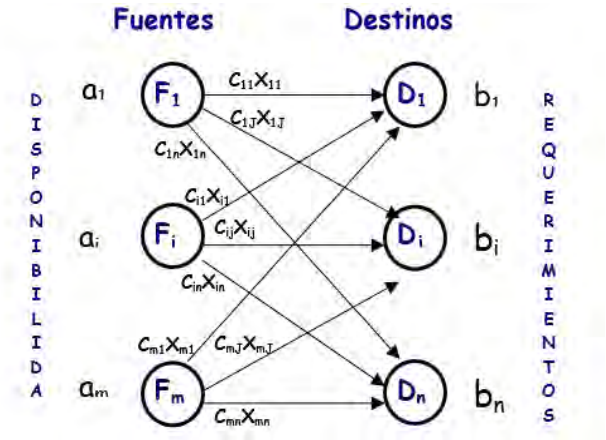
[31] Ibid. p. 165.

1.4.4.1 Modelo general del problema de transportes. Es un caso especial de problema de programación Lineal, en el que todos los coeficientes de las variables en las restricciones tienen coeficiente 1, esto es:

$$a_{ij} = 1; \text{ para todo } i, \text{ para todo } j$$

Gráficamente

Figura 10. Modelo general del problema de transportes



X_{ij} = Unidades a enviar desde la fuente i-ésima $(i = 1, \dots, m)$ al destino j-ésima $(j = 1, \dots, n)$

C_{ij} = Costo de enviar una unidad desde la fuente i-ésima $(i = 1, \dots, m)$ al destino j-ésima $(j = 1, \dots, n)$

a_i = Disponibilidad (oferta) en unidades, de la fuente i-ésima $(i = 1, \dots, m)$

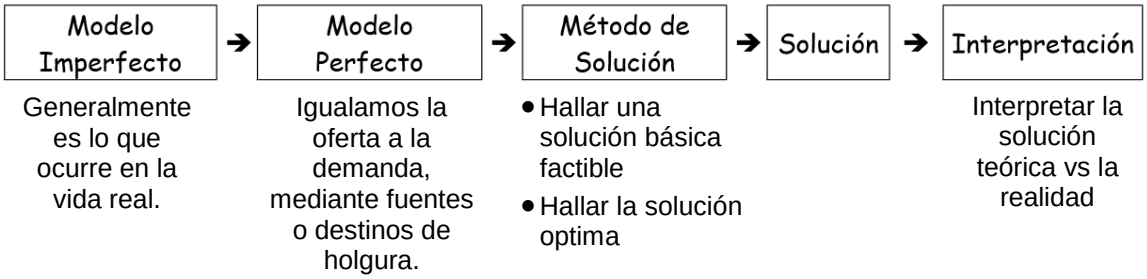
a_j = Requerimiento (demanda) en unidades, del destino j-ésima $(j = 1, \dots, n)$

Matemáticamente

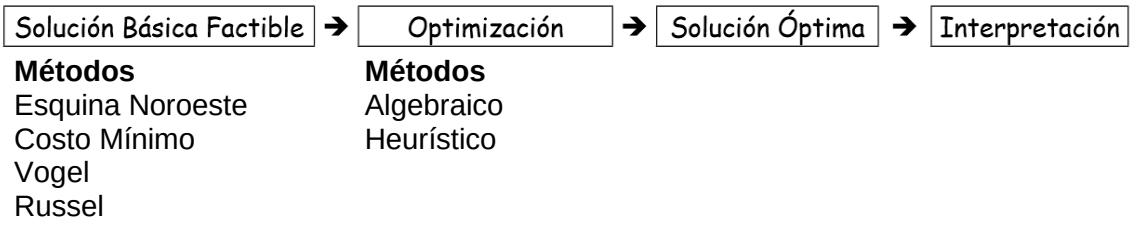
Minimizar $Z = C_{1,1}X_{1,1} + \cdots + C_{1,j}X_{1,j} + \cdots + C_{1,n}X_{1,n} + \cdots + C_{i,1}X_{i,1} + \cdots + C_{i,j}X_{i,j} + \cdots + C_{i,n}X_{i,n} + \cdots + C_{m,1}X_{m,1} + \cdots + C_{m,j}X_{m,j} + \cdots + C_{m,n}X_{m,n}$

$X_{11} + \dots + X_{1j} + \dots + X_{1n} = a_1$	$X_{11} + \dots + X_{ij} + \dots + X_{mn} = b_1$	$X_{ij} \geq 0$
$\vdots \quad \vdots \quad \vdots \quad \vdots$	$\vdots \quad \vdots \quad \vdots \quad \vdots$	
$X_{i1} + \dots + X_{ij} + \dots + X_{in} = a_i$	$X_{1j} + \dots + X_{ij} + \dots + X_{mj} = b_j$	$\forall_i , \forall_j$
$\vdots \quad \vdots \quad \vdots \quad \vdots$	$\vdots \quad \vdots \quad \vdots \quad \vdots$	
$X_{m1} + \dots + X_{mj} + \dots + X_{mn} = a_m$	$X_{m1} + \dots + X_{mj} + \dots + X_{mn} = b_n$	
Todo lo disponible es enviado	**Todo lo enviado fue requerido**	**!! No se pierde nada !!**

Metodología General



Metodología de solución



El algoritmo de transportes. El algoritmo de transportes sigue exactamente los mismos pasos que el método Simplex (sección 1.4.3). Sin embargo, se aprovecha la ventaja de la estructura especial del modelo de transporte para organizar los cálculos en una forma más cómoda[32].

Se debe agregar que el algoritmo especial de transporte fue desarrollado por primera vez cuando la norma eran los cálculos a mano, y se necesitaba soluciones "con método abreviado".

Los pasos del algoritmo de transporte son:

PASO 1. Determinar una solución básica factible de inicio y seguir con el paso 2.

PASO 2. Usar la condición de optimalidad del método Simplex para determinar la variable de entrada entre todas las variables no básicas. Si se satisface la condición de optimalidad, detenerse. En caso contrario seguir en el paso 3.

PASO 3. Usar la condición de factibilidad del método Simplex para determinar la variable de salida entre todas las variables básicas en ese momento, y determinar la nueva solución básica. Regresar al paso 2.

[32] Ibid. p. 165-168.

Determinación de la solución de inicio

La estructura especial del modelo de transporte permite asegurar que haya una solución básica no artificial de inicio, obtenida con uno de los cuatro métodos siguientes:

1. **Método de la esquina noroeste.**
2. **Método del costo mínimo.**
3. **Método de aproximación de Vogel.**
4. **Método de Russell.**

Los cuatro métodos difieren en la "calidad" de la solución básica de inicio que obtienen, en el sentido de que una mejor solución de inicio produce un valor objetivo menor. En general, el método de Russell produce la mejor solución básica de inicio, y el de la esquina noroeste produce la peor. La compensación es que el método de la esquina noroeste implica el mínimo de cálculos.

1.4.5 Métodos de soluciones iniciales en el problema de transportes. Una vez formulado el modelo matemático del problema de transportes, el siguiente paso consiste en resolver el modelo, es decir, en obtener los mejores valores numéricos para las variables de decisión. La forma en que se obtengan estos valores depende del tipo específico del modelo matemático utilizado. Por lo tanto, una vez definido el modelo, se podrá elegir un método apropiado para resolverlo.[*] En el problema de transportes estos métodos pertenecen a una de estas categorías:

- **Métodos óptimos**, que permiten obtener los mejores valores para las variables de decisión, es decir, aquellos valores que satisfacen simultáneamente todas las restricciones y proporcionan el mejor valor para la función objetivo.
- **Métodos Heurísticos**, que permiten obtener valores para las variables de decisión que satisfacen todas las restricciones. Aunque no necesariamente óptimos, estos valores proporcionan un valor aceptable para la función objetivo.

En comparación con los métodos óptimos, los métodos heurísticos son computacionalmente más eficientes y, por lo tanto, se utilizan cuando la obtención de soluciones óptimas conlleva excesivo tiempo o bien es imposible por el modelo matemático demasiado complejo. En el desarrollo de esta sección se habla de dichos métodos para la resolución del problema de transportes.

1.4.5.1 Método de esquina noroeste. El método de la esquina es un método de programación lineal hecho a mano para encontrar una solución inicial factible del modelo, muy conocido por ser el método más fácil al determinar una

[*]Nuevos métodos para la obtención de soluciones iniciales en el problema de transportes, Francisco López Ruiz. Departamento Organización de Empresas. E.U.I.T.I.-San Sebastián.

solución básica factible inicial, pero al mismo tiempo por ser el menos probable para dar una solución inicial acertada de bajo costo, debido a que ignora la magnitud relativa de los costos. Es un proceso utilizado para resolver problemas de transporte o asignación, si bien es un método no exacto tiene la ventaja de poder resolver problemas manualmente y de una forma rápida, muy cercano al valor óptimo.

Este método tiene como ventaja frente a sus similares la rapidez de su ejecución, y es utilizado con mayor frecuencia en ejercicios donde el número de fuentes y destinos sea muy elevado. Su nombre se debe al génesis del algoritmo, el cual inicia en la ruta, celda o esquina Noroeste. Es común encontrar gran variedad de métodos que se basen en la misma metodología de la esquina Noroeste, dado que se puede encontrar de igual manera el método de la esquina Noreste, Sureste o Suroeste.

Figura 11. Esquina noroeste

Algoritmo de resolución de la esquina noroeste. Se parte por esbozar en forma matricial el problema, es decir, filas que representen fuentes y columnas que representen destinos, luego el algoritmo debe de iniciar en la celda, ruta o esquina Noroeste de la tabla (esquina superior izquierda). En la figura 10, se muestra la ubicación de este método.

PASO 1: En la celda seleccionada como esquina Noroeste se debe asignar la máxima cantidad de unidades posibles, cantidad que se ve restringida ya sea por las restricciones de oferta o de demanda. En este mismo paso se procede a ajustar la oferta y demanda de la fila y columna afectada, restándole la cantidad asignada a la celda.

PASO 2: En este paso se procede a eliminar la fila o destino cuya oferta o demanda sea 0 después del "Paso 1", si dado el caso ambas son cero arbitrariamente se elige cual eliminar y la restante se deja con demanda u oferta cero según sea el caso.

PASO 3: Una vez en este paso existen dos posibilidades, la primera que quede un solo renglón o columna, si este es el caso se ha llegado al final el método, "detenerse".

La segunda es que quede más de un renglón o columna, si este es el caso iniciar nuevamente el "Paso 1".

1.4.5.2 Método de costo mínimo. El método del costo mínimo o de los mínimos costos es un algoritmo desarrollado con el objetivo de resolver problemas de transporte o distribución, arrojando mejores resultados que métodos como el de la esquina noroeste, dado que se enfoca en las rutas que presentan menores costos[33].

Este método asigna lo más posible a la celda de menor costo. Es factible que los vínculos se rompan de manera arbitraria. Las filas y columnas que han sido completamente asignadas no se tienen en cuenta y el proceso de asignación continua. El procedimiento se completa cuando se satisfacen todos los requerimientos de fila y columna.

Algoritmo de resolución del costo mínimo

PASO 1: De la matriz se elige la ruta (celda) menos costosa (en caso de un empate, este se rompe arbitrariamente) y se le asigna la mayor cantidad de unidades posible, cantidad que se ve restringida ya sea por las restricciones de oferta o de demanda. En este mismo paso se procede a ajustar la oferta y demanda de la fila y columna afectada, restándole la cantidad asignada a la celda.

PASO 2: En este paso se procede a eliminar la fila o destino cuya oferta o demanda sea 0 después del "Paso 1", si dado el caso ambas son cero arbitrariamente se elige cual eliminar y la restante se deja con demanda u oferta cero (0) según sea el caso.

PASO 3: Una vez en este paso existen dos posibilidades, la primera que quede un solo renglón o columna, si este es el caso se ha llegado al final el método, "detenerse". La segunda es que quede más de un renglón o columna, si este es el caso iniciar nuevamente el "Paso 1".

1.4.5.3 Método de aproximación Vogel. El método de aproximación de Vogel, es una versión mejorada del método del costo mínimo, que en general produce mejores soluciones de inicio. Entre otras definiciones esta: Vogel es un método heurístico de resolución de problemas de transporte capaz de alcanzar una solución básica no artificial de inicio, este modelo requiere de la realización de un número generalmente mayor de iteraciones que los demás métodos heurísticos

[33] Ibid. p. 179.

existentes con este fin, sin embargo producen mejores resultados iniciales que los mismos.

De todos los métodos existentes para la obtención de una solución básica realizable es el más efectivo, tanto que acerca a la solución óptima y en muchos casos la proporciona directamente.

Algoritmo de resolución de Vogel. El método consiste en la realización de un algoritmo que consta de 3 pasos fundamentales y 1 más que asegura el ciclo hasta la culminación del método[34].

PASO 1: Determinar para cada fila y columna una medida de penalización restando los dos costos menores en filas y columnas.

PASO 2: Escoger la fila o columna con la mayor penalización, es decir que de la resta realizada en el "Paso 1" se debe escoger el número mayor. En caso de haber empate, se debe escoger arbitrariamente (a juicio personal).

PASO 3: De la fila o columna de mayor penalización determinada en el paso anterior se debe de escoger la celda con el menor costo, y en esta asignar la mayor cantidad posible de unidades. Una vez se realiza este paso una oferta o demanda quedará satisfecha por ende se tachará la fila o columna, en caso de empate solo se tachará 1, la restante quedará con oferta o demanda igual a cero.

PASO 4: de ciclo y excepciones

- Si queda sin tachar exactamente una fila o columna con cero oferta o demanda, detenerse.
- Si queda sin tachar una fila o columna con oferta o demanda positiva, determine las variables básicas en la fila o columna con el método de costos mínimos, detenerse.
- Si todas las filas y columnas que no se tacharon tienen cero oferta y demanda, determine las variables básicas cero por el método del costo mínimo, detenerse.
- Si no se presenta ninguno de los casos anteriores vuelva al paso 1 hasta que las ofertas y las demandas se hayan agotado.

1.4.5.4 Método de Russell. Proporciona una solución inicial mejor que la que se obtuvo por el método de Vogel, debido a que la solución obtenida con este procedimiento está más cerca a la óptima que se quiere, debido a que la distancia total recorrida aun es menor. Sin embargo, el método de Russell requiere de muchos más cálculos que el método de Vogel.

[34] Ibid. p. 180-181.

Este método proporciona otro criterio excelente y fácil de llevarlo a la práctica en un ordenador pero no para la forma manual, debido a que es necesario realizar numerosos cálculos.

Todavía se necesita más experimentación para determinar cuál es más eficiente en promedio (respecto al método de Vogel), pero con frecuencia este criterio ha proporcionado una solución mejor[35]. En un problema grande, quizás pueda ser interesante aplicar ambos criterios y posteriormente utilizar la mejor solución que se obtenga para iniciar las iteraciones que permitan obtener la solución óptima.

El procedimiento es el siguiente:

- Calcúlense las cantidades $ui(i = 1, \ldots\ldots, m) y\ vj(j = 1, \ldots\ldots\ldots, n)$ sabiendo que:

  $Ui = Max(Cij) y\ vj = Máx(cij).$

- Encuéntrese la mejor variable xij , para formar una base usando los estimadores encontrados en (1) de acuerdo con el criterio de Dantazing; esto es:

  $$Xij = Máx(ij)[(ui + vj - Cij) > 0]$$

- Se introduce a la base:
  $Xij = Min\ (ai, bi).\ \ Si\ ai < bi$ hágase:
  $Bj = bj - ai$ Se elimina la fila i
  Si $bi < ai\ hagase\ ai = ai - bi$ se elimina la comuna j
  Si $ai = bj\ eliminese\ la\ fila\ i\ o\ la\ columna\ j$

- Se termina el método si todos los $ai\ y\ bj$son ceros, si no es así debe repetir el proceso desde 1.

Solución óptima: El desarrollo de una solución óptima para el problema de transporte implica evaluar cada celda no utilizada para determinar sin un cambio en ella resulta ventajoso desde el punto de vista del costo total. Solo se hace el cambio, y se repite el proceso cuando todas las celdas han sido evaluadas y se han hecho los cambios pertinentes, el problema está resuelto.

1.4.6 Método de multiplicadores. En el proceso de la resolución de problemas de transporte, se utilizó el método de multiplicadores para las iteraciones que determinan la variable de entrada. La variable que entra se determina mediante el uso de la condición de optimalidad del método Simplex. Los cálculos de los

[35] RUIZ, Francisco López, Nuevos métodos para la obtención de soluciones iniciales en el problema de transporte. En: Revista de Dirección y Administración de Empresas. Número 10, diciembre 2002, p. 159-173.

coeficientes de la función objetivo están basados en las relaciones primales duales[36].

Explicación del método de los multiplicadores con el método simplex. La relación entre el método de los multiplicadores y el método Simplex se puede explicar de acuerdo con la estructura especial de la programación lineal que representa el modelo de transporte, el problema dual* asociado se puede escribir en la forma:

$$Maximizar\ z = \sum_{i=1}^{m} a_i\, u_i + \sum_{j=1}^{n} b_j\, v_j$$

Sujeto a

$$u_i + v_j \leq\ c_{ij} \text{ para toda } i \text{ y j}$$

$$u_i\ y\ v_j \text{ No restringidas}$$

En donde

$a_i =$ Cantidad de oferta en la fuente i
$b_j =$ Cantidad de demandas en el destino j
$c_{ij} =$ Costo unitario de transporte entre la fuente i y el destino j
$u_i =$ Variable dual de la restricción asociada con la fuente i
$v_j =$ Variable dual de la restricción asociada con el destino j

El modelo de transporte por consiguiente se convierte en un tema indispensable de estudio para la resolución de problemas de transporte. Sabiendo que la necesidad del transporte surge por el hecho de que los bienes de consumo y los factores de producción tienen generalmente una localización diferente, al tiempo que desde el punto de vista social aumentan las posibilidades culturales y sociales de los individuos, ya que sin el transporte las relaciones sociales se encuentran más restringidas.

[36] Ibid. p. 195.
*Problema Dual: es una programación lineal definida en forma directa y sistemática a partir del modelo original (o primal) de programación lineal.

1.5 HERRAMIENTAS TECNOLÓGICAS PARA LA CONSTRUCCIÓN DE SISNOVA

La construcción de SISNOVA se hizo posible gracias a la investigación de nuevas tecnologías. Su desarrollo se implementa en diferentes plataformas y lenguajes de programación, lo que hace, que sea un sistema robusto y escalable.

A continuación se da una descripción general de las tecnologías usadas para el desarrollo de SISNOVA y que hacen que sea un sistema de gran apoyo para los diferentes desarrolladores de software, ya que es multiplataforma, multilenguaje, y además cuenta con grandes capacidades de cálculo y de aspectos visuales muy detallados y llamativos, que hacen del proceso de resolución de problemas como los de transportes, una experiencia muy fácil de manejar.

1.5.1 Construcción del servicio Web y la biblioteca de controles de usuario.

Se encuentran en un momento especial en la industria de computación. Se está en el inicio de una nueva manera de hacer y de integrar las aplicaciones, gracias al apoyo de tecnologías como las nombradas a continuación.

1.5.1.1 Microsoft .Net. Algunos expertos de la industria de la computación predicen que las nuevas tecnologías en su función de hacer e integrar las aplicaciones traerán cambios que serán equivalentes al que produjo la introducción de la PC, la interface visual o al surgimiento mismo de Internet. Dispositivos móviles como celulares, Tablet, hasta televisores u otros dispositivos hogareños estarán conectados entre sí, con servidores y distintas aplicaciones. El elemento integrador será Internet. Estar ahora en el inicio de la tercera generación de Internet, con el uso de tecnologías como Visual Studio .NET se permite ser protagonistas del cambio.

El tema de fondo es romper barreras. Barreras entre distintas aplicaciones que tienen información, barreras entre sistemas, barreras entre los sistemas y la gente que los utiliza, barreras entre las organizaciones, en fin llegar a no tener límite de nada.

¿Qué es Microsoft .Net?. Es un conjunto de tecnologías dispersas, que en muchos casos ya existían, que Microsoft ha integrado en una plataforma común con el objetivo de facilitar el desarrollo de los actuales servicios[37]. Estos son los pilares de esta nueva plataforma:

- **Integración**: Proporcionar mecanismos para que una empresa pueda ofrecer servicios a otras empresas o clientes de una forma sencilla y rápida.

[37] ORALLO, Enrique Hernández, Introducción a Microsoft.NET. En < http://WWW.acta.es/ > [13 de diciembre de 2012]

- **Nuevos dispositivos**: La forma más común de acceso a Internet hasta ahora ha sido el ordenador personal con sus limitaciones de movilidad. Pero recientemente han ido apareciendo una serie de dispositivos que permiten el acceso a servicios Internet de forma rápida y directa, como por ejemplo Tablet, teléfonos móviles, videoconsolas, etc. Esto supone un cambio radical en la forma de acceder a este tipo de servicios.

Con estos objetivos, Microsoft .NET, es una plataforma para construir, ejecutar y experimentar la tercera generación de aplicaciones distribuidas, que consiste en los siguientes elementos:

- Un modelo de programación basado en XML.
- Un conjunto de servicios Web XML, como Microsoft .NET My Services para facilitar a los desarrolladores integrar estos servicios.
- Un conjunto de servidores que permiten ejecutar estos servicios (como .NET Enterprise Servers).
- Software en el cliente para poder utilizar estos servicios.
- Herramientas para el desarrollo como: Visual Studio.NET.

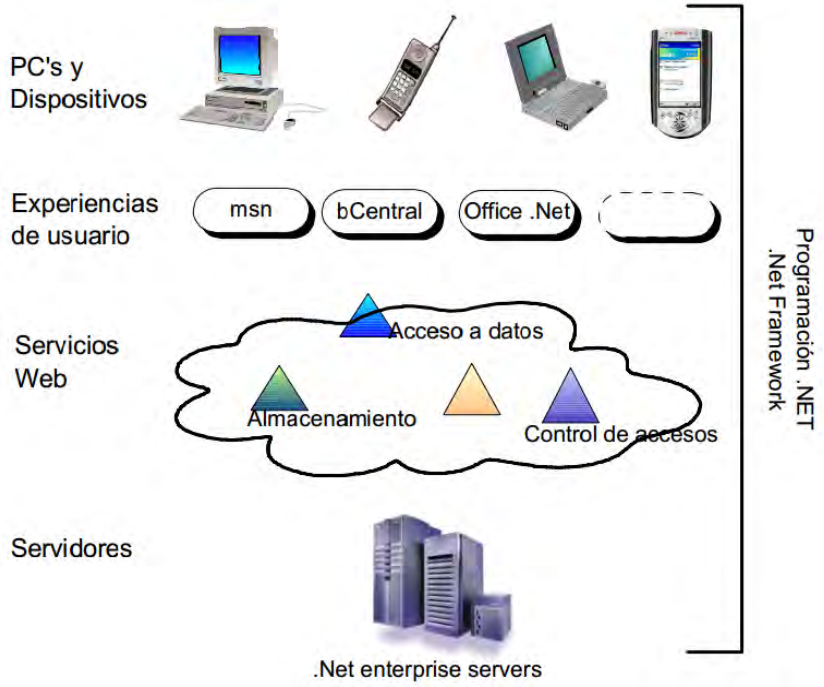
.NET no sólo se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, sino también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación. Para esto hace un uso extensivo de numerosos estándares globales que son de uso extensivo en la industria. Algunos ejemplos de estos estándares son los ya nombrados XML, HTTP, SOAP, y WSDL.

La plataforma de ejecución intermedia se muestra en la figura 12.

Figura 12. Plataforma de ejecución intermedia.

Una parte importante de esta plataforma es el software de los dispositivos clientes y servidores, que ha sido el mercado habitual de Microsoft. Para los dispositivos clientes, Microsoft planea integrar .NET en cualquier dispositivo imaginable. Esto supone para las empresas aumentar el número de potenciales clientes que puedan utilizar sus servicios sin limitarse al PC. En la figura 13, se muestra los elementos de la plataforma .Net[38].

Figura 13. Elementos de la plataforma .NET



Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado.

Además .NET intenta ofrecer una manera rápida y económica pero a la vez segura y robusta de desarrollar aplicaciones o como la misma plataforma las denomina, soluciones permitiendo a su vez una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

El “framework” o marco de trabajo, constituye la base de la plataforma .NET y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

[38] Microsoft .Net, <http://es.wikipedia.org/wiki/Microsoft> [noviembre 2013].

Framework 4. Es el modelo de programación completo y coherente de Microsoft para compilar aplicaciones que ofrezcan una sensacional experiencia visual del usuario, comunicación perfecta y segura, y la capacidad de modelar una amplia gama de procesos empresariales[39].

.NET Framework 4 funciona en paralelo con versiones anteriores de .NET Framework. Las aplicaciones basadas en versiones anteriores de .NET Framework continuarán ejecutándose en la versión que tienen definida como destino de forma predeterminada.

Microsoft .NET Framework 4 proporciona las siguientes mejoras y características nuevas:

- **Mejoras en el entorno en tiempo de ejecución de lenguaje común (CLR) y en la biblioteca de clases base (BCL)**
  - Mejora en el rendimiento, incluida una mayor compatibilidad con equipos multinúcleo, recolección de elementos no utilizados en segundo plano y asociación del generador de perfiles en el servidor.
  - Nuevos tipos numéricos y archivo asignado en memoria.

- **Innovaciones en los lenguajes Visual Basic y C#;** por ejemplo, lambdas de instrucciones, continuaciones de línea implícitas y distribución dinámica.

- **Mejoras en el acceso a datos y el modelado**
  - Entity Framework permite a los desarrolladores programar con bases de datos relacionales usando objetos .NET y LINQ.
  - Servicios de datos de WCF es un componente de .NET Framework que permite crear servicios y aplicaciones basados en REST que usen OData* para exponer y usar datos a través de la Web.

- **Mejoras en ASP.NET**
  - Más control sobre HTML, identificadores de elemento y hojas CSS personalizadas que facilitan enormemente la creación de formularios Web Forms que admiten optimización del motor de búsqueda y son conformes a los estándares.
  - **Nuevas características de datos dinámicos**, incluidos nuevos filtros de consulta, plantillas de entidad, mayor compatibilidad con Entity Framework 4 y características de validación y creación de plantillas que se pueden aplicar fácilmente a formularios Web Forms existentes.

[39] MICROSOFT, Microsoft .NET Framework 4 (instalador Web), Detalles rápidos, Versión: 4, <http://WWW.microsoft.com/es-co/download/details.aspx?id=17851> [Publicado 21 febrero de 2011]

*OData: es un protocolo de acceso de datos para la Web.

- **Mejoras en Windows Communication Foundation (WCF)**, en la compatibilidad con Servicios de flujos de trabajo de WCF, que permiten programas con actividades de mensajería y correlación.

- **Nuevas características de programación en paralelo**, como la compatibilidad con bucles en paralelo, LINQ paralelo (PLINQ) y estructuras de datos de coordinación que permiten a los desarrolladores aprovechar la eficacia de procesadores multinúcleo.

Sin quitar importancia del uso de los servicios Web, ya que toda la industria de software está enfocada en ello. La gran competencia es por proveer de las mejores herramientas basadas en estándares y las más fáciles y más productivas herramientas para construir las aplicaciones. La plataforma .NET es la infraestructura, los servicios y productos que Microsoft ofrece para esto.

Antes de la adopción del modelo de Servicios Web basados en XML los datos eran 'islas' que se encontraban dentro de las aplicaciones en las empresas. Era muy difícil y costoso implementar soluciones para acceder a la información desde afuera de la aplicación y la empresa. Las aplicaciones pueden ahora, comunicarse entre sí y con los sistemas de sus socios, proveedores y clientes gracias a los Servicios Web y XML.

1.5.1.2 Windows Communication Foundation– WCF. Las colaboraciones entre empresas se han vuelto cada vez más comunes, y las diferentes organizaciones empresariales usan software diferente como su implementación profunda. Por lo tanto, se requiere un marco de programación de comunicación, independiente de la plataforma e independiente del lenguaje, para facilitar la comunicación. WCF, presenta estas características.

¿Qué es WCF?. Es un conjunto de bibliotecas que provee Microsoft en el Framework .NET para la construcción de aplicaciones conectadas a través de un nuevo modelo de programación orientado a servicios [40] .El mecanismo de operación de WCF, consiste en que los clientes pueden invocar y consumir servicios múltiples, y un único servicio puede ser consumido por varios clientes.

WCF permite describir, publicar, implementar y consumir servicios, no solo con la interoperabilidad de los Web Services entre plataformas servidor y cliente, sino también utilizando diferentes plataformas de transporte de forma transparente al resto de la arquitectura.

[40] GUZMÁN, Hernan, “Diferencias entre Web Services y WCF (Windows Communication Foundation)” En: .Net wcf Webservices, 2011.

Características de WCF. Windows Communication Foundation permite un amplio control sobre las funciones de mensajería de una aplicación [41]. Se diseñó prácticamente para ofrecer un enfoque manejable para la creación de servicios Web y clientes de servicios Web. A continuación se incluyen el conjunto de características que posee:

- **Orientación a servicios**: WCF permite crear aplicaciones orientadas a servicios. Los servicios tienen la ventaja de estar débilmente acoplados entre una aplicación y otra en lugar de incluidos en el código. Esto implica que cualquier cliente creado en cualquier plataforma se puede conectar con cualquier servicio siempre y cuando se cumplan los contratos esenciales.

- **Interoperabilidad**: WCF se crea para interoperar con los servicios Web que admite un conjunto de especificación ya conocidas y que son estándar en los servicios Web.

- **Varios modelos de mensajes:** toda comunicación con un servicio de WCF se produce a través de los extremos del servicio, mediante distintos modelos. El más común es el de solicitud/respuesta donde un extremo solicita datos y el otro extremo le contesta, existe otros como el de mensaje unidireccional donde un extremo envía un mensaje sin esperar respuesta y uno más complejo es el dúplex donde dos extremos envían datos.

- **Metadatos de servicios:** WCF admite la publicación de metadatos de servicios utilizando los formatos como WSDL, esquemas XML. Estos metadatos pueden utilizarse para generar y configurar automáticamente clientes para el acceso a los servicios de WCF.

- **Contratos de datos**: dado que WCF se basa en .NET Framework, también incluye métodos con código sencillo para proporcionar los contratos que se desea aplicar. WCF utiliza un motor de la serialización llamado Serializador de contrato de datos de forma predeterminada para serializar y deserializar los datos, es decir convertirlos de y a XML. Es decir, para comunicarse, el cliente y el servicio no tienen que compartir los mismos tipos, solo los contratos de datos.

- **Seguridad:** la seguridad de WCF se basa en conceptos ya en uso e implementados en varias infraestructuras de seguridad. WCF admite algunas de esas infraestructuras, como Secure Sockets Layer (SSL). Sin embargo, va más allá de admitir las infraestructuras de seguridad existentes implementando las nuevas normas de seguridad interoperables en mensajes codificados por SOAP, lo que garantiza el consumo del servicio de SISNOVA por SOAP.

[41] MEDINA Juan Carlos, Windows Communication Foundation (WCF). En: Un poco de Sistemas Distribuidos, febrero 2013.

- **Varios transportes y codificaciones:** Los mensajes pueden enviarse con cualquiera de los protocolos y codificaciones integrados. La combinación más frecuente de protocolo y codificación consiste en enviar mensajes SOAP codificados de texto utilizando el Protocolo de transferencia de hipertexto (HTTP) usado en la WWW. Estos mensajes pueden codificarse como texto o utilizando un formato binario optimizado. Los datos binarios pueden enviarse de manera eficaz utilizando el estándar MTOM.

- **Mensajes confiables y en cola**: Las colas y sesiones de confianza son las características de WCF que implementan la mensajería de confianza, que se define como la manera en que un origen de mensajería de confianza transfiere mensajes de manera fiable a un destino de mensajería de confianza. La mensajería de confianza tiene los siguientes aspectos clave: 1. Transfiere garantías para los mensajes enviados desde un origen a un destino sin tener en cuenta los errores de transporte o de transferencia de los mensajes, 2. Separación del origen y el destino, que proporciona errores y recuperaciones independientes del origen y el destino, así como una transferencia y entrega de mensajes confiable, incluso cuando el origen o el destino no están disponibles.

- **Mensajes duraderos**: Un mensaje duradero es aquel que nunca se pierde debido a una interrupción de la comunicación. Los mensajes que forman parte de un modelo de mensajes duraderos siempre se guardan en una base de datos. Si se produce una interrupción, la base de datos le permite reanudar el intercambio de mensajes cuando se restablezca la conexión.

- **Transacciones:** WCF proporciona un amplio conjunto de características que permiten crear transacciones distribuidas a las aplicaciones de servicio Web. WCF implementa la compatibilidad con el protocolo WS-AtomicTransaction (WS-AT) que permite que las aplicaciones de WCF hagan fluir transacciones a aplicaciones interoperables, como servicios Web interoperables compilados mediante tecnología de otro fabricante.

- **Compatibilidad con AJAX y REST:** WCF se puede configurar para procesar datos XML “sin formato” que no se ajustan en un sobre SOAP. WCF también se puede extender para admitir formatos XML concretos, como ATOM (un estándar popular de RSS), e incluso formatos no XML, como notación de objetos JavaScript (JSON).

- **Extensibilidad:** WCF permite modificar y extender los componentes en tiempo de ejecución para controlar y extender precisamente las aplicaciones basadas en servicio. La arquitectura de WCF tiene varios puntos de extensibilidad. Si se necesita una función adicional, existen una serie de puntos de entrada que permiten personalizar el comportamiento de un servicio.

Las ventajas de WCF

- Uno de los grandes atractivos de WCF es la facilidad con la que permite al desarrollador la creación de servicios Web interoperables y aplicaciones que hagan uso de los mismos.

- WCF puede ser configurado para trabajar de forma independiente de SOAP y usar RSS en su lugar.

- WCF es una de las tecnologías de comunicación más rápidas y ofrece un rendimiento excelente en comparación con otras especificaciones de Microsoft.

- Para mejorar la comunicación y la velocidad de transmisión necesita ser optimizado. Esto se logra mediante la transmisión de datos XML codificados en binario en lugar de texto sin formato para disminuir la latencia.

- Gestión del ciclo de vida del objeto y administración de transacciones distribuidas son aplicables en cualquier aplicación desarrollada utilizando WCF.

- Utiliza las especificaciones Web Services para proporcionar fiabilidad, seguridad y gestión de transacciones.

- Los mensajes pueden estar en cola con la persistencia de colas. Como resultado, no se producen retrasos, incluso bajo condiciones de alto tráfico.

- WCF está diseñado para comunicarse con otras aplicaciones, además de los diversos sucesores y predecesores de la tecnología de Microsoft.

Integración de WCF. La integración de WCF con otras tecnologías Microsoft es posible gracias a la flexibilidad extrema con la que cuenta la plataforma WCF[42]. Los flujos de trabajo simplifican el desarrollo de aplicaciones encapsulando los pasos del flujo de trabajo como “actividades”.

En la primera versión de Windows Workflow Foundation, un desarrollador tenía que crear un host para el flujo de trabajo. La versión siguiente de Windows Workflow Foundation se integró con WCF. Esto permitió hospedar cualquier flujo de trabajo fácilmente en un servicio de WCF.

Microsoft .NET Services es una iniciativa de computación en nube que utiliza WCF para la creación de aplicaciones habilitadas para Internet. En esta investigación se hizo uso de .NET Services para crear servicios WCF.

[42] PACHECO, Juan Carlos Ruiz, Integración de WCF con otras tecnologías de Microsoft. En: ¿Qué es Windows Communication Foundation? Microsoft, 2013.

Arquitectura WCF. WCF admite muchos estilos de desarrollo de aplicaciones distribuidas proporcionando una arquitectura superpuesta. En su base, la arquitectura de canal de WCF proporciona primitivos asíncronos de paso de aprobación de mensajes sin tipo.

Generados sobre esta base están las funciones de protocolos para un intercambio de datos de transacción seguro y fiable, así como una amplia variedad de opciones de codificación y transporte.

El modelo de programación tipificada (llamado modelo de servicio) está diseñado para facilitar el desarrollo de aplicaciones distribuidas y proporcionar a los desarrolladores pericia en servicios Web y comunicación remota .NET Framework, así como a aquellos que llegan a WCF con cierta experiencia en desarrollo. En la figura 14, se muestra las capas principales de la arquitectura WCF.

Figura 14. Arquitectura de WCF

Contratos y descripciones: los contratos definen varios aspectos del sistema de mensajes. El contrato de datos describe cada parámetro que constituye cada mensaje que un servicio puede crear o utilizar. El contrato del mensaje define partes específicas del mensaje utilizando los protocolos SOAP y permite el control más fino sobre las partes del mensaje, cuando la interoperabilidad exige tal precisión.

Tiempo de ejecución de servicio: la capa del tiempo de ejecución del servicio contiene los comportamientos que solo se producen durante la operación actual del servicio, es decir, los comportamientos en tiempo de ejecución del servicio.

Un comportamiento de error especifica lo que sucede cuando se produce un error interno en el servicio, por ejemplo, controlando qué información se comunica al cliente y que demasiada información puede dar ventaja a un usuario malintencionado para organizar un ataque.

Mensajería: La capa de la mensajería se crea de canales. Un canal es un componente que procesa un mensaje de alguna manera, por ejemplo, autenticando un mensaje. Los canales funcionan en los mensajes y encabezados del mensaje. Esto es diferente de la capa en tiempo de ejecución del servicio, que se ocupa principalmente de procesar el contenido de los cuerpos de los mensajes.

Hay dos tipos de canales: canales de transporte y canales de protocolo:

- **Los canales de transporte**: leen y escriben mensajes de la red. Algunos transportes utilizan un codificador para convertir los mensajes que se representan como conjuntos de información XML, hacia y desde la representación de la secuencia de bytes utilizada por la red.

- **Los canales de protocolo**: implementan protocolos de procesamiento de mensajes, a menudo leyendo o escribiendo encabezados adicionales en el mensaje.

Alojamiento y activación: En su forma final, un servicio es un programa. Como otros programas, un servicio se debe ejecutar en un ejecutable. Esto se conoce como un servicio con host propio.

Los servicios también se pueden hospedar o ejecutar en un ejecutable administrado por un agente externo, como IIS. Un servicio también se puede ejecutar automáticamente como un servicio de Windows[43].

Enlaces proporcionados por el sistema. Los enlaces especifican el mecanismo de comunicación para dirigirse a un punto final e indicar como unirse a este punto. Las uniones consisten en que los elementos que se definen como canales WCF están acodados hasta proporcionar los rasgos de comunicación requeridos[44].
Las uniones que son embarcadas con WCF, se muestran en la tabla 2. Enlaces proporcionados por el sistema.

[43] MICROSOFT, Arquitectura de Windows Communication Foundation. En: Windows Communication Foundation <http://msdn.microsoft.com/> [2013].
[44] Ibid. Configuración de enlaces proporcionados por el sistema.

Tabla 2. Tipos de enlaces

Enlaces (Binding)	Elemento de configuración	Descripción
BasicHttpBinding	<basicHttpBinding>	Un enlace que es adecuado para la comunicación con los servicios Web conformes perfil WS-Basic. Este enlace utiliza HTTP como el transporte y el texto / XML como la codificación de mensajes predeterminado.
WSHttpBinding	<wsHttpBinding>	Un enlace seguro e interoperable que es adecuado para los contratos de servicios que no son dúplex.
WS2007HttpBinding	<ws2007HttpBinding>	Una encuadernación segura e interoperable, proporciona el apoyo a las versiones correctas de la Seguridad
WSDualHttpBinding	<wsDualHttpBinding>	Una encuadernación segura e interoperable conveniente para contratos de servicio duplex o comunicación por intermediarios de SOAP
WSFederationHttpBinding	<wsFederationHttpBinding>	Una encuadernación segura e interoperable que apoya el protocolo WS-FEDERACIÓN permitiendo a las organizaciones que están en una federación para de manera eficiente autenticar y autorizar a usuarios.
WS2007FederationHttpBinding	<ws2007FederationHttpBinding>	Una encuadernación segura e interoperable que proviene de WS2007HttpBinding y apoya la seguridad federada.
NetTcpBinding	<netTcpBinding>	Una encuadernación segura y optimizada conveniente para comunicación de máquina entre usos de aplicaciones WCF.

NetNamedPipeBinding	<netNamedPipeBinding>	Una encuadernación segura, confiable, optimizada que es conveniente para la comunicación sobre máquina entre usos WCF.
NetMsmqBinding	<netMsmqBinding>	Una encuadernación colocada en fila que es conveniente para la comunicación de máquina entre usos de aplicaciones WCF.
NetPeerTcpBinding	<netPeerTcpBinding>	Una encuadernación segura que permite, la comunicación de multimáquina.
WebHttpBinding	<WebHttpBinding>	Una encuadernación que configura puntos finales para los servicios WCF alojados en la Web que son expuestos por HTTP en lugar de mensajes SOAP.
MsmqIntegrationBinding	<msmqIntegrationBinding>	Una encuadernación que es conveniente para la comunicación entre máquinas WCF y la Formación de una cola de espera de Mensajes existentes.

Bibliotecas de clases de .Net. .NET Framework incluye clases, interfaces y tipos de valor que agilizan y optimizan el proceso de desarrollo y proporcionan acceso a las funciones del sistema. Para facilitar la interoperabilidad entre lenguajes, la mayoría de los tipos de .NET Framework son conformes a CLS* y, por tanto, se pueden utilizar en todos los lenguajes de programación cuyo compilador satisfaga los requisitos de CLS. Los tipos de .NET Framework son la base sobre la que se compilan aplicaciones, componentes y controles de .NET.

.NET Framework incluye tipos que realizan las funciones siguientes:

- Representar tipos de datos base y excepciones.
- Encapsular estructuras de datos.
- Realizar E/S.

*CLS: Especificación de Lenguaje Común, que es un conjunto de características de lenguaje básicas requeridas por la mayoría de las aplicaciones.

- Obtener acceso a información sobre tipos cargados.
- Invocar las comprobaciones de seguridad de .NET Framework.
- Proporcionar: acceso a datos, interfaz gráfica para el usuario (GUI) independiente de cliente e interfaz GUI de cliente controlada por el servidor.

.NET Framework proporciona un conjunto completo de interfaces, así como clases abstractas y concretas (no abstractas). Se pueden utilizar las clases concretas tal como están o, en muchos casos, derivar las clases propias de ellas. Para utilizar la funcionalidad de una interfaz se puede crear una clase que implemente la interfaz o derivar una clase de una de las clases de .NET Framework que implementa la interfaz.

1.5.1.3 Visual Studio (IDE). Visual Studio es un potente Entorno de Desarrollo Integrado (IDE) que asegura código de calidad durante todo el ciclo de vida de la aplicación, desde el diseño hasta la implementación[45]. Visual Studio es su solución "todo en uno", para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET.

Visual Studio, permite a los desarrolladores crear aplicaciones, sitios y aplicaciones Web, así como servicios Web en cualquier entorno que soporte la plataforma .NET. Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas Web y dispositivos móviles.

Actualmente Visual Studio, provee las herramientas necesarias para desarrollar aplicaciones que necesitan estar siempre conectadas, aplicaciones para desarrolladores independientes, o para entorno empresarial[46].

Entre estas herramientas se tiene:

- **Interfaz mejorada**. La interfaz ha sido rediseñada para que el flujo de trabajo sea más sencillo. Se ha optimizado el orden de las pestañas y las herramientas, para mejorar al máximo la navegación en la aplicación.

- **Desarrollo para Windows 8**. En Visual Studio 2012 se encuentran las herramientas necesarias para el desarrollo de aplicaciones en Windows 8, todo para adaptarse a la nueva interfaz propuesta por Microsoft en el sistema operativo. Tiene la opción de distribución y venta en la Windows Store, lo cual permite llegar a clientes en todo el mundo.

[45] GIBELLI, Monique, Visual Studio, Microsoft Latinoamérica. <http://WWW.visualstudio.com/> 2012.
[46] SOMASEGAR, S., "Announcing the Visual Studio 2013 Release Candidate", En: Somasegar`s blog. [Publicado: Septiembre de 2013].

- **Desarrollo con Web Dev mejorado**. Se han incluido nuevas plantillas, herramientas más eficientes de publicación y soporte para los nuevos estándares HTML5, CSS3, ASP.NET. Todo ello permite el desarrollo Web en escritorio y en entornos móviles para facilitar un mejor Responsive Design*.

- **Desarrollo en la nube**. Con las herramientas nuevas de Visual Studio 2012 se pueden desarrollar aplicaciones enfocadas a la nube mediante servidores virtuales ilimitados, para después migrar a servidores reales. Esto permite la ejecución y simulación de las aplicaciones desarrolladas, y mejor distribución del almacenamiento y los recursos consumidos.

- **Mejoras en Sharepoint**. Se ha mejorado con plantillas, opciones de implementación y nuevos diseñadores para aprovechar mejor todas las posibilidades de Sharepoint. Incluye mejoras en las funciones ALM, y una nueva herramienta llamada LightSwitch.

- **Organización de proyectos**. Gracias a las importantes mejoras en ALM, el proceso de organización de proyectos para desarrollo de aplicaciones complejas es más sencillo. En la herramienta se han incluido formas de seguimiento de los que integran el proyecto. Estos pueden hacer comentarios y también interactuar para que el proyecto pueda ser monitorizado y revisado más ágilmente.

Visual Studio aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones empresariales.

1.5.1.4 Lenguaje C#. Es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores generar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Se puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más[47].

A pesar que el lenguaje C# forma parte de la plataforma .NET, que es una interfaz de programación de aplicaciones. C# es un lenguaje independiente que originariamente se creó para producir programas sobre esta plataforma .NET.

* Responsive Design, o diseño adaptativo: es una técnica de diseño y desarrollo Web que mediante el uso de estructuras e imágenes fluidas en la hoja de estilo CSS, consigue adaptar el sitio Web al entorno del usuario.

[47] MICROSOFT, Introducción al lenguaje C# y .NET Framework. En: Introducción a Visual C#, <http://msdn.microsoft.com/> [2013].

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave override como medio para evitar redefiniciones accidentales.

En C#, una estructura es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios en línea de documentación XML.
- Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

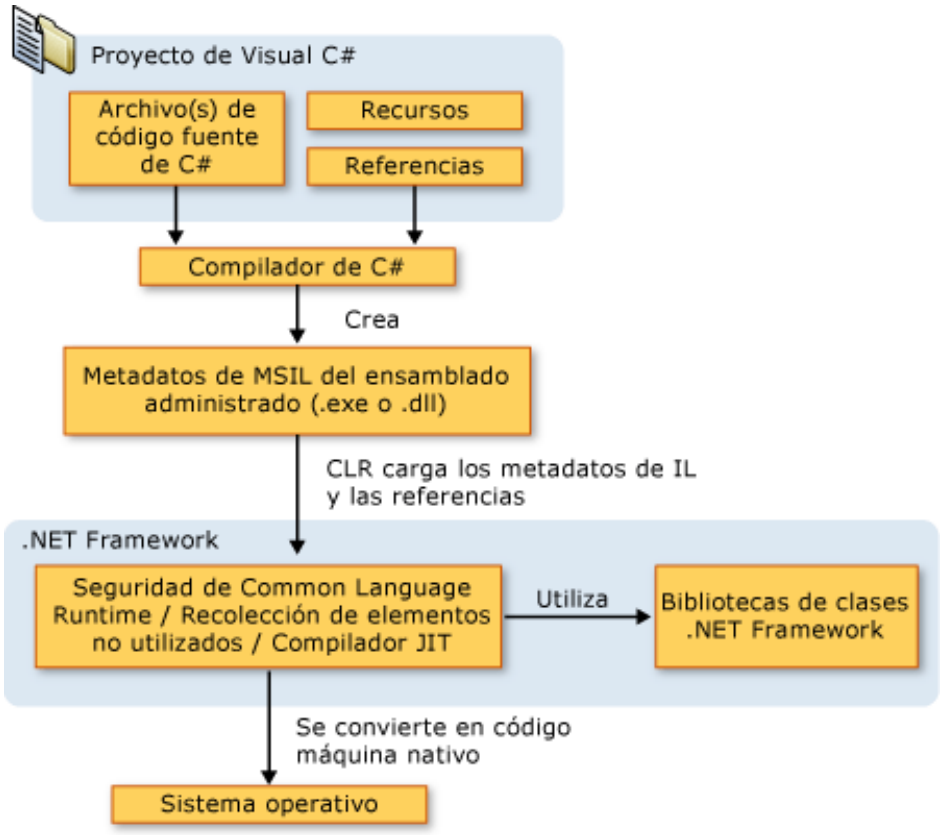
Arquitectura de la plataforma .NET framework. Los programas de C# se ejecutan en .NET Framework, un componente que forma parte de Windows y que incluye un sistema de ejecución virtual denominado CLR y un conjunto unificado de bibliotecas de clases. CLR es la implementación comercial de Microsoft de la Infraestructura de Lenguaje Común (CLI), norma internacional que constituye la base para crear entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin problemas.

Cuando se ejecuta un programa de C#, el ensamblado se carga en CLR, con lo que se pueden realizar diversas acciones en función de la información del manifiesto. El código ejecutado por CLR se denomina algunas veces "código administrado", en contraposición al "código no administrado" que se compila en lenguaje máquina nativo destinado a un sistema específico.

En la siguiente figura 15 se muestran las relaciones en tiempo de compilación y tiempo de ejecución de los archivos de código fuente de C#, las bibliotecas de clases de .NET Framework, los ensamblados y CLR[48].

[48] Ibid.

Figura 15. Desde el código fuente de C# a la ejecución de la máquina.



- **Recursos del lenguaje C#.** El proceso de generación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos.

A continuación, se enumeran algunos recursos de C#:

- **Lenguaje integrado de consulta – LINQ.** Es un conjunto de características en Visual Studio 2008 que agrega eficaces capacidades de consulta a la sintaxis de los lenguajes C# y Visual Basic. LINQ incluye patrones estándar y de fácil aprendizaje para consultar y actualizar datos, y su tecnología se puede extender para utilizar potencialmente cualquier tipo de almacén de datos[49].

Tradicionalmente, las consultas con datos se expresan como cadenas sencillas, sin comprobación de tipos en tiempo de compilación ni compatibilidad. Además, es necesario aprender un lenguaje de consultas diferente para cada tipo de origen de

[49] Ibid. Language-Integrated Query (LINQ), noviembre 2007.

datos: bases de datos SQL, documentos XML, servicios Web diversos, etc. LINQ convierte una consulta en una construcción de lenguaje de primera clase en C# y Visual Basic. Las consultas se escriben para colecciones de objetos con establecimiento inflexible de tipos, utilizando palabras clave del lenguaje y operadores con los que se está familiarizado.

Puede utilizar consultas LINQ en proyectos nuevos o junto a consultas que no son LINQ en proyectos existentes. El único requisito es que el proyecto esté orientado a la versión 3.5 de .NET Framework.

- **Operador Lambda (= >).** El token => se denomina operador lambda. Una expresión lambda es una función anónima que puede contener expresiones e instrucciones y se puede utilizar para crear delegados o tipos de árboles de expresión. Las expresiones lambda son expresiones insertadas similares a los métodos anónimos, pero más flexibles; se utilizan mucho en las consultas de LINQ que se expresan en la sintaxis del método[50]**.**

El operador => se lee como "va a". El lado izquierdo del operador lambda especifica los parámetros de entrada (si existe alguno), mientras que el lado derecho contiene el bloque de expresiones o instrucciones. La expresión lambda x => x * x se lee "x va a x veces x". Tiene la misma prioridad que el operador de asignación (=) y es asociativo por la derecha.

Puede especificar explícitamente el tipo de la variable de entrada o permitir que el compilador la infiera; en cualquier caso, la variable tiene establecimiento inflexible de tipos en tiempo de compilación. Al especificar un tipo, debe incluir entre paréntesis el nombre del tipo y el nombre de variable, tal y como se muestra en el ejemplo siguiente:

```
int shortestWord = words.Min((string w) => w.Length;
```

Como ya se mencionó las expresiones lambda se utilizan en consultas de LINQ, consultas basadas en métodos como argumentos para métodos de operador de consulta estándar, tales como Where.

Cuando se utiliza la sintaxis de método para llamar al método Where, el parámetro es un tipo delegado System.Func<T, TResult>[*]. Una expresión lambda constituye la manera más práctica de crear ese delegado. Las expresiones lambda permiten que las llamadas a Where tengan un aspecto similar, aunque, de hecho, el tipo de objeto creado desde la expresión lambda sea diferente.

[50] Ibid. Expresiones lambda (Guía de programación de C#), 2013.

[*]Func<T, TResult> (Delegado): Encapsula un método que tiene un parámetro y devuelve un valor del tipo especificado por el parámetro.

1.5.2 Tecnologías para prototipos de prueba. Para los prototipos de prueba de SISNOVA, se utilizaron diferentes tecnologías que permitieron que el desarrollo de los prototipos sea multiplataforma y multilenguaje.

A continuación, se describe dichas tecnologías dentro de las siguientes secciones:

- **Sistemas Operativos**
- **Entornos de Desarrollo Integrados (IDE)**
- **Lenguajes de Programación**
- **Máquinas Virtuales**
- **Servidores Web**
- **Hardware**

1.5.2.1 Sistemas operativos

"Así como el trabajo de las abejas melíferas, criaturas que, por una regla en la naturaleza, enseñan el acto de ordenar un reino poblado."[*]

Todos los sistemas operativos poseen ciertas características fundamentales en común: deben administrar la memoria, la capacidad de procesamiento, los dispositivos y periféricos, archivos y redes.

Un sistema de cómputo consta de software (programas) y hardware (la maquina física y sus componentes electrónicos). El Sistema Operativo es la parte fundamental del software, la porción del sistema de cómputo que gestiona todo el hardware y el software. Para ser más específicos, controla todos los archivos, todos los dispositivos, todas las secciones de la memoria principal y todos los nanosegundos de tiempo de procesamiento. Controla quién y cómo puede usar el sistema. En resumen, es el jefe[51].

Después de este breve concepto de sistema operativo, se procede a especificar los sistemas operativos utilizados en los prototipos de prueba.

- **Windows.** Desde la aparición del Microsoft© Windows hasta la actualidad se han desarrollado muchas versiones, provocando una gran revolución en la vida de las personas.

Windows 7. Es un sistema estable y seguro que aporta importantes novedades respecto a la interfaz de usuario, incrementa el rendimiento del equipo y es compatible con un gran número de dispositivos y aplicaciones.

[*] William Shakespear (1564 – 1616; en Enrique V).
[51] MCHOES, Ann Mclver, FLYNN, Ida M., Sistemas Operativos, 6º edición, Cengage Learning Editores, S.A., México, 2011. ISBN 978-607-481-485-9.

El desarrollo de este sistema operativo comenzó inmediatamente después del lanzamiento de Windows Vista. El 20 de julio de 2007 se reveló que ese sistema operativo era llamado internamente por Microsoft como la versión 7. Hasta ese momento la compañía había declarado que Windows 7 tendría soporte para plataformas de 32 bits y 64 bits. Con su variedad de ediciones y amplias características Windows 7 ha tenido gran acogida.

Windows Server. Es una marca que abarca una línea de productos servidor de Microsoft Corporation, consiste en un sistema operativo diseñado para servidores de Microsoft y una gama de productos dirigidos al mercado más amplio de negocios. Windows Server ofrece más control sobre la infraestructura de servidores y red, mejor hosting, protección del sistema operativo y el entorno de red, herramientas administrativas intuitivas, facilidad de consolidación, virtualización de servidores y aplicaciones.
Se describe como "listo para el centro de datos"[52], así mismo como "el movimiento hacia una mayor modularidad, automatización más fuerte y una mejor virtualización tienen mucho sentido en un mundo de nubes públicas y privadas", pero remarcó que "dicho esto, la capacidad de Windows para suministrar errores oscuros y el tiempo que consumen esos errores no han cambiado", y concluye que "no obstante, se trata de una fuerte mejora en general".

- **Linux.** El sistema operativo Linux, nació en 1991, gracias a un estudiante de la universidad de Helsinki[53]. El éxito de Linux se basa en una idea ingeniosa de su creador, L. Torvalds: en inscribir su proyecto bajo términos de la licencia GPL y proponer a todos los programadores y otros hackers[*] de Internet que le ayudaran.

El impulso de Linux proviene en gran parte del hueco que lleno en términos de núcleo en el proyecto GNU[**]. En conclusión Linux es un sistema libre completo.

- **Android.** Es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o Tablet.

El lanzamiento de Android como nueva plataforma para el desarrollo de aplicaciones móviles ha causado grandes expectativas y está teniendo una importante aceptación tanto por parte de los usuarios como de la industria.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK)[***]. Pero están disponibles otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para

[52] BISSON, Simon. En: ZDNet. <http://WWW.zdnet.com/> [22 de julio de 2013].
[53] Ibid. p. 499.
[*] El termino Hacker representa aquí los primeros programadores en los sistemas Unix.
[**] GNU: es un proyecto que consiste en reescribir completamente un sistema operativo libre.
[***] Android SDK: es el proceso por el cual se crean nuevas aplicaciones para el sistema operativo Android.

aplicaciones o extensiones en C o C++, Google App Inventor y un entorno visual para programadores novatos. El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o SDK.

Arquitectura de Android. En el caso de Android está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además, esta distribución permite acceder a las capas más bajas mediante el uso de bibliotecas para que así el desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware de los teléfonos. En la figura 16 se representa la Arquitectura.

Figura 16. Arquitectura Android

1.5.2.2 Entornos de Desarrollo Integrado (IDE). Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI)[54].

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Java, C#, Visual Basic, Python, Delphi, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Los entornos integrados desarrollo (IDE) utilizados en esta investigación son los siguientes.

[54] MARTIN, James, “Design of Real-Time Computer Systems”, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1967

- **Visual Studio.** Microsoft Visual Studio es un potente entorno de desarrollo integrado que presenta un conjunto de herramientas destinadas a ayudarle a escribir y modificar el código para los programas, así como a detectar y corregir errores en los programas. Más adelante se detalla más acerca de este tema.

- **NetBeans.** Es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Además es un producto libre y gratuito sin restricciones de uso, que facilita bastante el diseño gráfico asociado a aplicaciones Java.

- **Dreamweaver.** Es el programa incluido en el suite de Adobe, destinado a la creación y la gestión de sitios Web.

La gran ventaja de este editor es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en JavaScript-C, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino rutinas de JavaScript* que hace que sea un programa muy fluido, que todo ello hace, que programadores y editores Web hagan extensiones para su programa y lo ponga a su gusto.

- **Eclipse.** La plataforma Eclipse está diseñado para la creación de entornos de desarrollo integrados (IDE), que se puede utilizar para crear aplicaciones tan diversas como sitios Web, programas Java™ embebidos, programas C++ y Empresa JavaBeans. El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software.

*JavaScript: es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python. La arquitectura módulos permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java en el SDK de Eclipse. Y no tiene por qué ser usado únicamente con estos lenguajes, ya que soporta otros lenguajes de programación[55].

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

Ventajas de los IDEs.

- Es más ágil y óptimo para usuarios que no son expertos en manejo de consola.
- Formateo de código.
- Funciones para renombrar variables, funciones.
- Warnings y errores de sintaxis en pantalla de algo que no va a funcionar al interpretar o compilar.
- Poder crear proyectos para poder visualizar los archivos de manera gráfica.
- Herramientas de refactoring como por ejemplo seria extraer una
- porción de código a un método nuevo.
- No es recomendado pero posee un navegador Web interno por si se quiere probar las cosas dentro de la IDE.

De acuerdo a todo esto cabe resaltar que algunos IDEs no son gratuitos por el mismo motivo que el software son utilizados para trabajos mejorados, por tal motivo en los IDEs se puede implementar líneas de código donde se pueda resolver algún problema con base al compilador, este es el que permitirá modificar o corregir errores del programa.

Para esta investigación se tomó en cuenta que algunos de los IDEs necesitan muchas aplicaciones para poder sacar la aplicación propia, y en otros es muy fácil utilizarlos porque al momento de escribir las líneas de código corrige los problemas de la aplicación, de acuerdo a esto, se procedió implementar los IDEs anteriormente nombrados para los diferentes prototipos de aplicación que presenta SISNOVA.

1.5.2.3 Lenguajes de programación. Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué

[55] Object Technology International, Inc., “Eclipse Platform Technical Overview”, originally published July 2001, <http://eclipse.org/whitepapers/eclipse-overview.pdf>, [Febrero 2003].

ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML**.**

Para los prototipos de aplicación de SISNOVA, se utilizaron los siguientes lenguajes de aplicaciones.

- **C#.** Es un lenguaje de propósito general orientado a objetos creado por Microsoft para su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes. C# fue diseñado para combinar el control a bajo nivel de lenguajes como C y la velocidad de programación de lenguajes como Visual Basic[56].

Este lenguaje es una parte esencial en la programación de SISNOVA. Ya que C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic, etc. De hecho, su creador Anders Heljsberg[57] fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los programadores de C/C++/Java sea lo más inmediata posible.

- **Visual Basic.** Es un lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para Microsoft.

Al desarrollar un programa en Visual Basic, se hace una distinción clara entre la interfaz con el usuario y el trabajo propio del programa. Los compiladores de Visual Basic generan código que requiere una o más bibliotecas de enlace dinámico para que funcionen, conocidas comúnmente como DLL (sigla en inglés de dynamic-link library). Estas bibliotecas DLL proveen las funciones básicas implementadas en el lenguaje, conteniendo rutinas en código ejecutable que son cargadas bajo demanda en tiempo de ejecución. Además de las esenciales, existe un gran número de bibliotecas del tipo DLL con variedad de funciones, tales como las que facilitan el acceso a la mayoría de las funciones del sistema operativo o las que proveen medios para la integración con otras aplicaciones.

Así como bibliotecas DLL, hay numerosas aplicaciones desarrolladas por terceros que permiten disponer de variadas y múltiples funciones, incluso mejoras para el propio Visual Basic; las hay también para el empaquetado y distribución, y hasta para otorgar mayor funcionalidad al entorno de programación (IDE).

[56] CEBALLOS Javier, Enciclopedia de Microsoft Visual C#. Alfaomega. 2006.
[57] HELJSBERG, Anders, “The C# Programming Language”, Pearson Education, Inc., Boston, 2004. ISBN 0-321-15491-6.

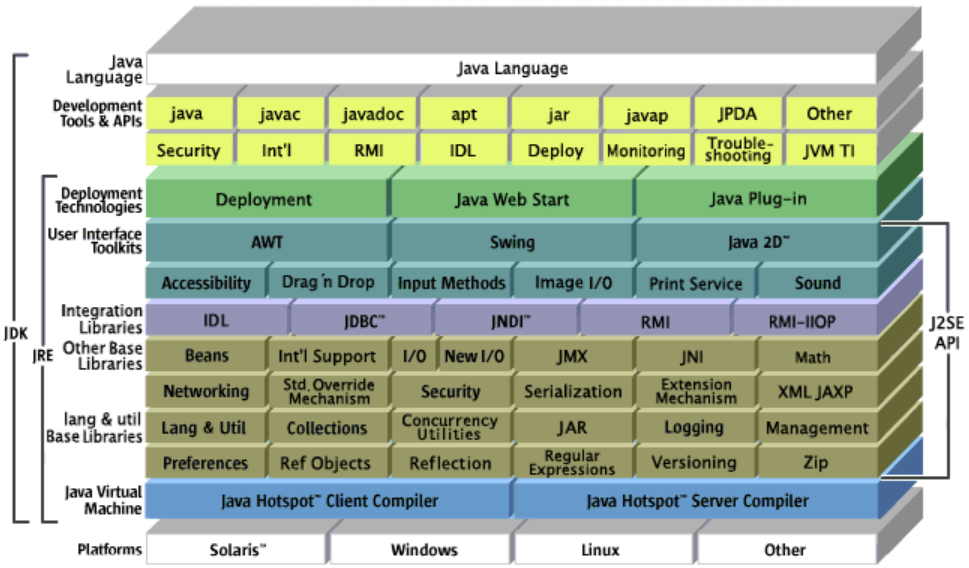
- **JAVA.** El lenguaje de programación Java fue originalmente desarrollado por James Gosling* de Sun Microsystems y publicado en el 1995 como un componente fundamental de la plataforma Java. Su sintaxis deriva mucho de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos.

La creación de este lenguaje y plataforma se inspiró en las funcionalidades interesantes propuestas por otros lenguajes tales como C++, Eiffel, SmallTalk, Objective C, Cedar/Mesa, Ada Perl. El resultado es una plataforma y un lenguaje idóneo para el desarrollo de aplicaciones seguras, distribuidas y portables en numerosos periféricos y sistemas transportables interconectadas en red pero también en Internet (clientes ligeros) y en estaciones de trabajo (clientes pesados)[58].

La plataforma Java. Por definición, una plataforma es un entorno de hardware o de software en la cual se puede ejecutar un programa. La mayoría de las plataformas actuales son la combinación de una máquina y de un sistema operativo (ej.: PC + Windows). La plataforma Java se distingue por el hecho de que solo se compone de una parte de software que se ejecuta en numerosas plataformas físicas y diferentes sistemas operativos.

La figura siguiente, sobre el lenguaje Java, muestra los diferentes componentes que conforman a la plataforma.

Figura 17. Plataforma Java



*James Gosling, es un famoso científico de la computación conocido como el padre del lenguaje de programación Java.

[58] GROUSSARD, Thierry. Java 7: Los fundamentos del lenguaje Java. Ediciones ENI. Barcelona 2012. p. 12.

Como muestra la figura 17, se compone de los elementos siguientes:

- La máquina virtual. Java (JVM)
- La interfaz de programación de aplicación Java (API Java), repartida en tres categorías (APIs básicas, APIs de acceso a los daros y de integración con lo existente, APIs de gestión de la interfaz de las aplicaciones con el usuario).
- Las herramientas de despliegue de las aplicaciones.
- Las herramientas de ayuda al desarrollo.

- **PHP.** Usa una mezcla entre interpretación y compilación para intentar ofrecer a los programadores la mejor mezcla entre rendimiento y flexibilidad.

PHP compila para el código propio en una serie de instrucciones siempre que estas son accedidas. Estas instrucciones son entonces ejecutadas una por una hasta que el script termina. PHP es recompilado cada vez que se solicita un script.

Una ventaja importante de interpretar el código es que toda la memoria usada por tu código es manejada por PHP, y el lenguaje automáticamente vacía esta memoria cuando el script finaliza. Esto significa que no hay que preocuparse de las conexiones a la base de datos, porque PHP lo hace por uno.

Es un lenguaje multiplataforma; los programas funcionan igual sobre diferentes plataformas, trabajando sobre la mayoría de servidores Web.

Para probar los prototipos de aplicación se hace uso de otras tecnologías, como la plataforma JAVA, que se utilizó para la creación de la aplicación de escritorio y la aplicación móvil**.**

En conclusión Java es una plataforma, que permitir ejecutar programas sin tener relativamente en cuenta el hardware final, sin volver a reescribir todo el código del programa y consiste en tres grandes bloques, el lenguaje Java, una máquina virtual y un programa de aplicación de interfaz o API.

1.5.2.4 Máquinas virtuales. Una máquina virtual es un software que simula a una computadora y puede ejecutar programas como si fuese una computadora real. Para las pruebas pertinentes de los prototipos de aplicación de SISNOVA, se tomaron las siguientes máquinas virtuales.

- **CLR.** El Entorno en tiempo de ejecución de lenguaje común es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET.

Este entorno virtual se encarga de aspectos importantes para una aplicación como la gestión de la memoria, la vida de los objetos, la seguridad y la gestión de subprocesos. Todos estos servicios unidos a su independencia respecto a

arquitecturas computacionales convierten la CLR en una herramienta muy útil ya que, en teoría, cualquier aplicación escrita para funcionar según la CLI puede ejecutarse en cualquier tipo de arquitectura de hardware[59].

- **JVM.** La máquina virtual de Java (JVM) es un programa ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial, el cual es generado por el compilador del lenguaje Java[60].

Esta JVM es la que le permite a Java ser multiplataforma debido a que cuando se compila el código de Java, este queda compilado en bytecode[*] que es lo que reconoce la JVM instalada en cualquier plataforma Windows, Linux o MacOS.

Para cada dispositivo debe haber una JVM específica, ya sea un teléfono móvil, un PC con Windows XP o un microondas. En cualquier caso, cada máquina virtual conoce el conjunto de instrucciones de la plataforma destino, y traduce un código escrito en lenguaje Java (común para todas) al código nativo que es capaz de entender el hardware de la plataforma. Esto permitió el desarrollo de los prototipos de aplicación de escritorio y móvil para SISNOVA.

- **Dalvik.** Es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. Ejecuta archivos en el formato Dalvik Executable (*.dex), un formato optimizado para el almacenamiento eficiente y ejecución mapeable en memoria. Su objetivo fundamental es el mismo que cualquier máquina virtual, permite que el código sea compilado a un bytecode independiente de la máquina en la que se va a ejecutar, y la máquina virtual interpreta este bytecode a la hora de ejecutar el programa[61].

Dalvik está basada en registros y puede ejecutar clases compiladas por un compilador Java y que posteriormente han sido convertidas al formato nativo usando la herramienta “dx”. El hecho de que corra sobre un kernel Linux le permite delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel. El uso de Dalvik para los prototipos de aplicación de SISNOVA, permite reducir bastante el tamaño del programa buscando información duplicada en las diversas clases y reutilizándola.

1.5.2.5 Servidores Web. La principal función de un servidor Web es almacenar los archivos de un sitio y emitirlos por Internet para poder ser visitado por los usuarios. Básicamente, un servidor Web es una gran computadora que guarda y transmite datos vía Internet. Cuando un usuario entra en una página de

[59] KATCHEROFF, Pablo, Desarrollador .Net, Sevagraf Longseller Argentina, p. 16. ISBN 978-987-1347-74-2.
[60] DEITEL, Harvey M., DEITEL, Paul J., Cómo programar en Java, Pearson Educación, 2004.
[*] Bytecode: es un código intermedio más abstracto que el código máquina.
[61] GIRONÉS, Jesús Tomás. El gran libro de Android, Tercera edición. Barcelona, 2013.

Internet su navegador se comunica con el servidor enviando y recibiendo datos que determinan qué es lo que ve en la pantalla. Por eso se dice que los servidores Web están para almacenar y transmitir datos de un sitio según lo que pida el navegador de un visitante.

Es importante contar con un servidor estable para alojar las aplicaciones, para lo cual en esta investigación se contó con uno que permitió alojar el prototipo de aplicación Web.

Sin los servidores Web la Internet tal como se conoce, no existiría. Los servidores son como la columna vertebral de la estructura de Internet. La industria del Web hosting es simplemente la forma de alquilar esos espacios de memoria y administración de datos.

- **Apache.** Es el servidor de páginas Web, que permite acceder a páginas Web alojadas en un ordenador[62]. Es el más utilizado seguido de Microsoft Information Services. En esta investigación se emplea el servidor Apache WampServer 5, por múltiples razones como disponibilidad, facilidad de instalación, pocos recursos necesarios, precio, disponibilidad del código fuente. Existen muchos otros aunque suelen estar especializados en nichos concretos de mercado.

Ventajas: Modular, Código abierto, Multi-plataforma, Extensible y Popular.

Además, Apache permite configurar un Hosting Virtual basado en IPs o en nombres, es decir, tener varios sitios Web en un mismo equipo (por ejemplo: nombreWeb1.com, nombreWeb2.com,….).

Características de Apache

- Soporte para los lenguajes perl, python, tcl y PHP.
- Módulos de autenticación: mod_access, mod_auth y mod_digest.
- Soporte para SSL y TLS.
- Permite la configuración de mensajes de errores personalizados y negociación de contenido.
- Permite autenticación de base de datos basada en SGBD.

La arquitectura característica de un servidor que funciona con Apache es modular. Esto quiere decir que está compuesto por partes o módulos que se utiliza de acuerdo con las necesidades que se presentan.

- **IIS 7.** Internet Information Services o IIS es un servidor Web y un conjunto de servicios para el sistema operativo Microsoft Windows.

[62] Macy, Ed, Apache, HarperCollins E-Books, 2008. ISBN 978-0-00-730747-0.

Este servicio convierte a una PC en un servidor Web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas Web tanto local como remotamente.

1.5.2.6 Hardware. El hardware es un término genérico utilizado para designar a todos los elementos físicos que componen un computador. Para las pruebas en los prototipos de aplicación de SISNOVA se hicieron uso de los siguientes elementos: Computador de Escritorio, Servidor, Teléfono Móvil y Tablet.

2 SISNOVA: SISTEMA DE ELEMENTOS DE SOFTWARE REUTILIZABLES, COMO APOYO A LA CONSTRUCCIÓN DE APLICACIONES ESCALABLES Y ROBUSTAS QUE PERMITAN LA RESOLUCIÓN DE PROBLEMAS DE TRANSPORTES

SISNOVA, nace de la necesidad de desarrollar sistemas robustos y escalables que aporten funcionalidades que den solución a problemáticas actuales. Si bien, los programadores cuentan con el acceso a Internet, en muchos casos no disponen de soluciones óptimas y adaptables a sus aplicaciones, de esta manera SISNOVA propone dos componentes: un servicio Web y una biblioteca de controles de usuario que están enfocados en la resolución de problemas de transportes.

2.1 DESARROLLO SISNOVA

SISNOVA consta de 2 productos enfocados hacia la reutilización de software en los tiempos de codificación y ejecución, orientados a resolver los problemas de transporte que se plantean en Investigación de operaciones. Estos 2 productos son en esencia un conjunto de componentes software y en adelante a estos productos se los enunciara como:

Transport_Service y Transport_LibraryControls, en la tabla 3 se especifica la arquitectura, capas y objetivo principal de cada producto.

Tabla 3. Especificación de productos SISNOVA

Producto	Arquitectura Implementación	Capas que implementa	Objetivo Principal
Transport_Service	SOA	Lógica de negocios, acceso a datos.	Presentar funcionalidades para la resolución de problemas de transporte.
Transport_LibraryControls	CBCD	Aplicación, lógica de negocios.	Presentar interfaces gráficas configurables, para visualizar planteamientos de problemas y soluciones del modelo de transportes.

En la tabla 3, la columna arquitectura implementación hace referencia a la arquitectura desde el punto de vista del usuario del componente, las columna capas se refiere a la arquitectura intrínseca en el componente. Esto hace apreciar el porqué de la separación de SISNOVA en dos productos, de esta forma se dota a cada producto con la permeabilidad de integrarse entre ellos u otros componentes externos.

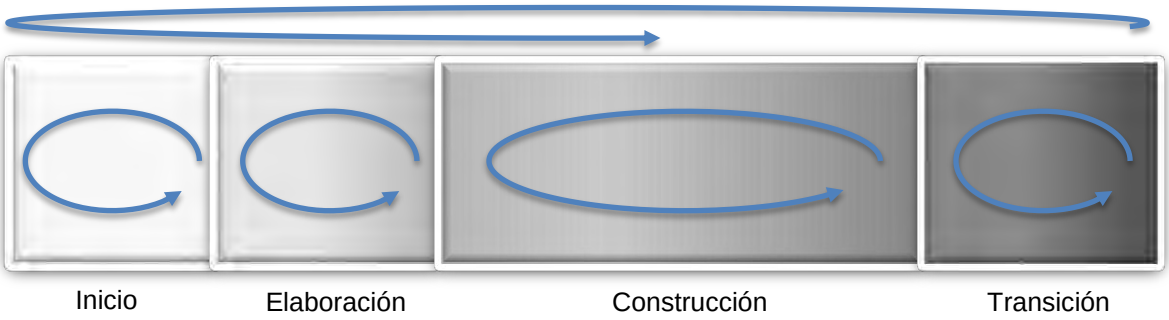
Los productos anteriormente dichos se han construido bajo la metodología RUP (metodología estudiada en el Capítulo 1, sección 1.3.5.2), por ser una metodología hibrida permeable, orientada bajo un esquema de retroalimentación iterativa incremental y por llevar una separación explicita de los flujos de trabajo de proceso y de soporte.

Debido a que RUP es una metodología iterativa incremental por fase y actividades se presenta la última versión, y de ella únicamente los artefactos software más relevantes. En principio se muestra el plan iterativo de SISNOVA, las entidades que interactúan con SISNOVA, las interacciones de los actores con el sistema, la arquitectura del sistema, el comportamiento del sistema y algunas especificaciones de implementación.

2.2 PLAN ITERATIVO RUP SEGUIDO PARA SISNOVA

SISNOVA se desarrolla en dos iteraciones de proceso* para el producto Transport_Service y tres iteraciones de proceso para el producto denominado Transport_LibraryControls. A continuación se especifica en más detalle los planes de iteraciones seguidos para cada producto, donde se especifican las iteraciones de fase por proceso, en la figura 18, las flechas circulares indican la estrategia de iteración de RUP.

Figura 18. Iteraciones de fase por proceso



2.2.1 Plan iterativo Transport_Service. Como se mencionó en el apartado introductorio, Transport_Service se desarrolla en 2 iteraciones de proceso: La Primera iteración orientada a la construcción de Transport_Service con funcionalidades de consumo bajo el protocolo SOAP (Capitulo 1, sección 1.3.3.4), la segunda iteración dedicada a proporcionar compatibilidad extra al servicio para el consumo mediante peticiones REST (Capitulo 1, sección 1.3.3.5) con envíos y

* Una iteración de proceso se refiere a la culminación de un objetivo propuesto para un sistema, en donde se ha seguido un respectivo orden de fase. Si aparece nuevos requerimientos al final de la fase del objetivo i se plantea un objetivo i+1 y se repite el proceso por fase. En la figura 17, la flecha circular de mayor tamaño encima de todas las fases, representa las iteraciones de proceso.

recepciones en los formato XML (Capitulo 1, sección 1.3.3.6) y JSON (Capitulo 1, sección 1.3.3.7). En la tabla 4, se aprecia el plan de iteración por fase seguido para cada iteración de proceso.

Tabla 4. Plan de iteración por fase de iteración de proceso para Transport_Service.

Fase	Iteraciones proceso 1	Iteraciones proceso 2
Inicio	1	1
Elaboración	2	1
Construcción	4	2
Transición	0	1

2.2.2 Plan iterativo Transport_LibraryControls. Transport_LibraryControls se construye en 3 iteraciones de proceso: En la primera fase solo se contempla un control de usuario con la capacidad de acoplarse en Visual Studio (Capitulo 1, sección 1.5.1.3) con una interface gráfica amigable con edición de planteamiento del problema y visualización de resultados en forma de grafos*, en la segunda iteración se adiciona una visualización grafica extra al control en forma matricial y en la tercera iteración se adicionan a la biblioteca, 2 controles de usuario complementarios al control principal, un control de detalles para ver la información de los elementos del control principal y otro control de navegación para proporcionar una conexión de enlace sobre las principales funcionalidades del control principal. En la tabla 5, se aprecia el plan de iteración por fase seguido para cada iteración de proceso.

Tabla 5. Plan de iteración por fase iteración de proceso para Transport_LibraryControls.

Fase	Iteraciones proceso 1	Iteraciones proceso 2	Iteraciones proceso 3
Inicio	1	1	1
Elaboración	1	1	2
Construcción	3	1	2
Transición	0	0	0

*Grafo, es una estructura de datos, en concreto un tipo abstracto de datos, que consiste en un conjunto de nodos y un conjunto de arcos que establecen relaciones entre los nodos.

2.3 ENTIDADES QUE INTERACTUARÁN CON SISNOVA

Partiendo del hecho de que una entidad*, representa una “cosa” u "objeto" del mundo real con existencia independiente, es decir, se diferencia unívocamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad y teniendo en cuenta que estas entidades son participantes y perceptibles de un sistema animado. Se contempla ha SISNOVA como un sistema perceptor, así como también a unas entidades externas con la capacidad de producir estímulos que en adelante se denominaran actores.

Estos actores demandaran funcionalidades para cada uno de los componentes de SISNOVA y se los clasifica en, actores humanos y actores máquina teniendo en cuenta el sentido cognitivo que requieren las actividades de cada uno dentro del sistema.

Los actores humanos se han dividido en actores expertos y en actores no expertos, con el fin de separar los roles de usuarios que cada uno desempeña desde la perspectiva de SISNOVA. Entonces un actor experto será aquel con la capacidad de configurar y adaptar algún componente de SISNOVA en un entorno conocido y posible de manera que produzcan una solución hacia un usuario común.

Estos usuarios comunes son los actores no expertos, y son quienes interactúan con SISNOVA a través de los software que produzcan los usuarios expertos o componentes gráficos que el sistema proporcione.

Los actores máquina son dispositivos hardware o software donde residen las aplicaciones de los actores expertos y son los encargados de gestionar la comunicación con SISNOVA a través de un canal receptor/emisor.

A continuación se presentan un detalle más específico de los actores que interactúan con SISNOVA.

2.3.1 Actores Transport_Service. Para el componente Transport_Service se han identificado dos actores un actor humano experto identificado como Desarrollador y un actor maquina software identificado como aplicación.

Los cuadros siguientes describen con mayor detalle a estos actores. En principio se describen los actores que interactúan con Transport_Service, seguido de los que interactúan con Transport_ LibraryControls.

En la tabla 6, se muestra la descripción del actor desarrollador, siendo este el actor principal del sistema.

*Entidad: nombre de una "cosa” u objeto de importancia para una organización o negocio, ya sea real o imaginaria, cuya información se necesita conocer o mantener.

Tabla 6. Descripción actor desarrollador para Transport_Service.

Nombre:	Desarrollador
Descripción	
Es el actor principal, debe contar con conocimientos sobre el tratamiento de servicios Web desde un lenguaje o lenguajes de programación. Este actor juega el papel de enlazador de consumo del Servicio y la plataforma* de desarrollo. Para Transport_Service este actor además de los conocimientos genéricos enunciados en el anterior párrafo, debe contar con algunos conocimientos sobre el modelo de transportes. Para este actor, Transport_Service debe ser tratado de forma similar a como el reutilizaría bibliotecas de clases o librerías de enlace dinámico.	
Objetivos	
– Preparar la plataforma de desarrollo para el consumo de servicios Web (Capitulo 1, sección 1.5.2.5). – Utilizar el servicio en base a los requerimientos preestablecidos. – Codificar los objetos necesarios para el envío y recepción de información. – Codificar las peticiones para el consumo de funcionalidades o recursos expuestos por el servicio. – Escoger Protocolo de consumo SISNOVA.	

En la tabla siguiente, se describe al actor aplicación, que puede ser cualquier aplicación que utiliza Transport_Service en su desarrollo.

Tabla 7. Descripción actor aplicación para Transport_Service.

Nombre:	Aplicación
Descripción	
Este actor hace referencia al software creado por un actor desarrollador para enlazar Transport_Service a una plataforma específica. Como consecuencia, este actor es abstracto** debido a la gran cantidad de posibilidades de aplicación que puedan presentarse. Para Transport_Service este actor es quien permite a un dispositivo*** comunicarse con el servidor.	

* Una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

** Esta abstracción hace referencia es que no se conoce las dimensiones especificas del actor (solo se conocen dimensiones generales) ya que esto depende de lo que se vaya a construir.

*** En el contexto referenciado es un hardware con la capacidad de conectarse a un servidor, por lo tanto un dispositivo puede ser un computador, una Tablet, un Smartphone etc

Objetivos
– Implementar los protocolos de comunicación y de información necesarios para el consumo del servicio. – Enviar solicitudes de funcionalidades o recursos con sus correspondientes insumos. – Recibir respuestas del servicio. – Presentar una interface entre el usuario de la aplicación y el servicio Web.

2.3.2 Actores Transport_ LibraryControls. Para Transport_ LibraryControls se han identificado dos actores humanos.

Tabla 8. Descripción actor diseñador Transport_ LibraryControls.

Nombre:	Diseñador
Descripción	
Este actor es quien se encargara de implementar las interface gráficas para Windows Forms en tiempo de diseño. Conoce las herramientas graficas que le proporciona el IDE* y como integrar nuevas herramientas a este. Para Transport_Library este actor es el encargado de integrar los controles de este componente a una aplicación Windows_Forms y configurar las propiedades graficas de cada uno.	
Objetivos	
– Integrar a Transport_Library a la caja de herramientas graficas del IDE. – Adaptar los controles a la aplicación Windows Forms. – Configurar propiedades gráficas. – Proporcionar eventos vacíos a los controles.	

Tabla 9. Descripción actor desarrollador Transport_LibraryControls.

Nombre:	Desarrollador
Descripción	
Este actor especializa al actor diseñador, con implementación y configuración de interfaces graficas en tiempo de codificación.	
Objetivos	
– Enlazar los controles de Transport_Library – Codificar acciones en los eventos de los controles. – Utilizar las funcionalidades y propiedades internas de cada control.	

*IDE: entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

2.4 INTERACCIONES DE LOS ACTORES CON EL SISTEMA

2.4.1 Diagrama de casos de uso Transport_Service. A continuación, se presentan los diagramas de casos de uso para los actores que intervienen en el componente Transport_Service.

Figura 19. Diagrama de casos de uso del actor desarrollador Transport_Service

En el diagrama anterior, se aprecia las interacciones que un actor Desarrollador tiene con el componente Transport_Service limitado en el recuadro de borde azul, se puede observar que los casos de uso que son ejecutados directamente por el actor son dependientes, en este sentido el caso de uso “codificar funcionalidad del servicio a consumir” depende del caso de uso “codificar clases e instanciaciones de objetos para el envío y recepción de información” que su vez depende de “codificar conexión al servicio” y este de “preparar plataforma de desarrollo para el consumo del servicio” quien está fuera de la frontera del sistema indicando que es una funcionalidad externa al servicio. Esta dependencia entre casos no significa que el desarrollador no pueda ejecutar el caso, significa que el desarrollador puede ejecutar el caso pero su correcto funcionamiento depende de otro u otros casos.

El caso de uso “preparar plataforma de desarrollo para el consumo del servicio” especifica que el desarrollador debe “escoger el protocolo de consumo SISNOVA” y según esto “agregar las bibliotecas” necesarias para consumir por SOAP o REST.

El caso de uso “codificar clases e instanciaciones de objetos para el envío y recepción de información” puede extenderse mediante la generación de un proxy* a través de las herramientas que el desarrollador tenga a disposición, siempre y cuando el desarrollador lo requiriera.

Para que un desarrollador ejecute el caso de uso “codificar funcionalidad del servicio a consumir” debe haber escogido previamente, una funcionalidad del servicio.

Figura 20. Diagrama de casos de use del actor aplicación Transport_Service.

En el figura 20, al igual que en el figura 19, se distinguen los casos de uso del sistema por el límite dado por el recuadro de borde azul, del mismo modo hay una dependencia entre casos. Los casos de envío y recepción dependen de la

*Un proxy, es un programa o dispositivo que realiza una acción en representación de otro. Para los servicios Web un proxy es quien importa los metadatos con los que genera código que se pueda usar para la invocación de los recursos o funcionalidades del servicio.

implementación de los protocolos de comunicación e información y este de la conexión del servicio. La implementación de protocolos necesariamente debe valerse de bibliotecas o librerías SOAP o REST y la conexión del servicio puede extenderse de forma síncrona o asíncrona. El caso de uso externo se refiere a una interface abierta o cerrada para que un usuario pueda interactuar con la aplicación.

2.4.2 Diagrama de casos de uso Transport_ LibraryControls

Figura 21. Diagrama de casos de uso para los actores diseñador y programador de Transport_LibraryControls.

En la figura 21, se observa que un programador es un desarrollador extendido es decir que tiene unas funcionalidades extra dentro del sistema. Debido a esta separación de funcionalidades se aprecia una frontera del sistema para el desarrollador contenida dentro de la frontera del programador, entonces las funcionalidades del desarrollador son un subconjunto de las funcionalidades del programador. Lo que destaca a estos actores se puede notar a través de sus casos de uso, ya que todas las funcionalidades que ellos representan están orientadas a tareas de configuración de cada uno de los controles de Transport_LibraryControls.

Figura 22. Diagrama de casos de uso del actor usuario de Transport_LibraryControls.

En la figura 22, se muestra la interactividad de un actor usuario con el sistema, cada uno de los casos de uso está orientado al funcionamiento general de cada control de usuario en tiempo de ejecución.

2.5 ARQUITECTURA DEL SISTEMA

2.5.1 Diagrama de capas y componentes

Figura 23. Diagrama de capas de SISNOVA.

En la figura 23, se ve la separación de capas para SISNOVA, las dependencias fuertes entre capas de componentes (flechas de color oscuro) y las dependencias débiles (flechas de color claro). Estas dependencias indican el grado de acoplamiento entre capas, es por eso que se ve ideal la dependencia débil entre las capas componente para indicar que cada componente puede complementarse entre sí u otros componentes externos a SISNOVA.

Figura 24. Diagrama de componentes de SISNOVA.

Estos componentes ofrecen interfaces para poder acoplarse, que puede ser de entrada o salida.

El componente Transport_Service resume sus interfaces de entrada mediante un paso de mensajes de entrada llamados “Input parameters” ver figura 24 y sus correspondientes salidas por medio de funcionalidades para el consumo mediante SOAP o REST.

En el diagrama se ve la composición del componente Library_Controls en sus tres componentes que son: CtrlSisnova, NavigateTransport y ViewElements con sus correspondientes interfaces, cada una de las cuales sigue un patrón de entrada mediante el estímulo de ejecución de funcionalidades, desencadenamiento de eventos y peticiones de edición, como interface de salida cada componente esta guiado a mostrar formularios, cambios de vista y ejecución de definiciones de eventos.

Es imprescindible destacar la posibilidad de conexión de los dos componentes mediante sus interfaces output input correspondientes.

2.5.2 Diagrama de clases. Con los anteriores diagramas se ha obtenido una visión general de las funcionalidades y estructura de SISNOVA, a continuación se presenta un grado más detallado de la estructura de dicho sistema.

Los diagramas siguientes muestran en un principio una vista general donde se pueden observar las clases más relevantes de cada componente y las relaciones entre ellas, le sigue una clasificación de clases adecuada a cada componente, con un grado de detalle para observar sus respectivos atributos, métodos, propiedades y eventos .

2.5.2.1 Diagrama de clases Transport_Service. El componente Transport_Service lo conforman dos interfaces principales para SOAP y REST como se mencionaba en la descripción de la figura 24, estas interfaces[*] estructuralmente son implementadas por la clase de servicio Transport_Service (ver figura 25) y son las que posibilitan la comunicación hacia el exterior[**], la implementación de las funcionalidades de estas interfaces es proporcionada por las clases Solution_ByIterationMethods y Start_SolutionMethods clases de lógica de negocio que utilizan la encapsulación de las clases de acceso a datos Transport_Solution, Transport_Node, Indicator, Parameters y StartParameters y las enumeraciones DirectionType y StartSolutionType.

[*] Una interface en un diagrama representa un patrón de diseño para los métodos de una clase.
[**] Las interfaces además deben estar declaradas con el atribute Datacontract y configuradas en el archivo de configuración del servicio.

Figura 25. Diagrama de clases de Transport_Service.

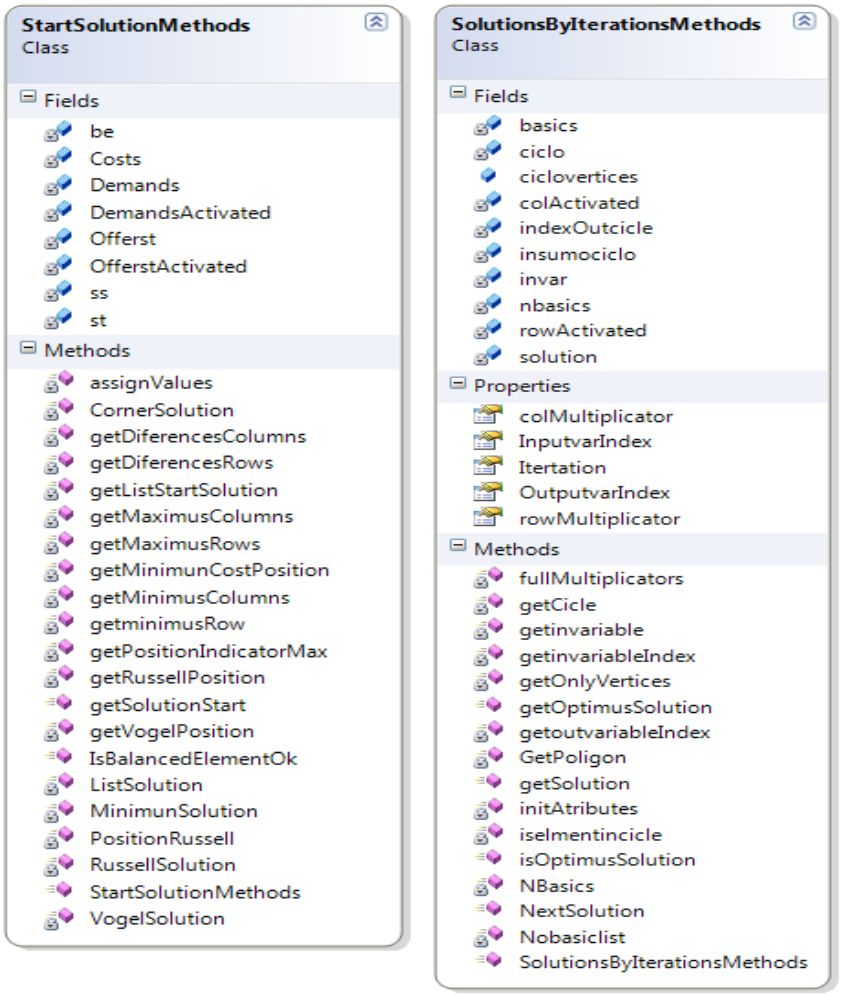
Una instancia de Tranport_Solution está compuesta por varias de Transport_Node y la clase StartParameters especializa a Parameters. A continuación se muestra en detalle los miembros de las clases anteriores.

Definición de clases de lógica de negocios

Las siguientes clases han sido clasificadas como de lógica de negocios porque son ellas quien efectúan todas las operaciones para solucionar un problema de transportes.

La clase StartSolutionMethods es la encargada de proporcionar una solución básica factible inicial por los métodos de esquina noroeste, costo mínimo, Vogel y Russell, esto lo logra a través del uso de sus métodos privados y lo expone al exterior mediante el método publico getSolutionStart, también implementa una funcionalidad publica para saber si los parámetros de entrada están balanceados.

Figura 26. Detalle de miembros de clases de lógica de negocios para Transport_Service.

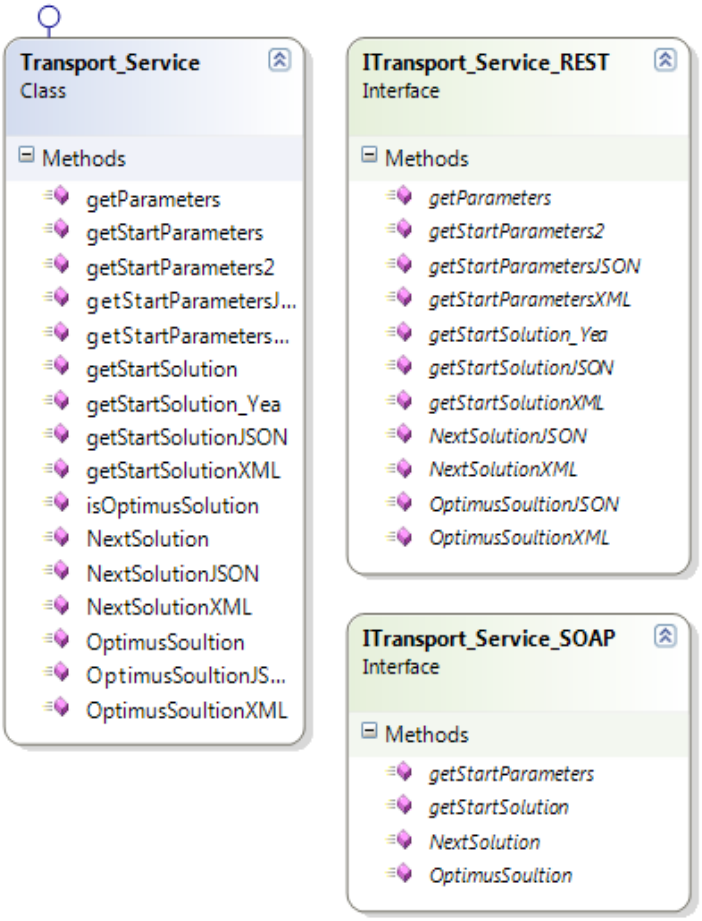


La clase SolutionByIterationMethods es la encargada de mejorar una solución factible mediante el método de multiplicadores para ello hace uso de métodos privados usados por los métodos públicos getOptimusSolution que obtiene la solución óptima al problema en un solo paso y por NextSolution que mejora la solución factible actual. Esta clase también proporciona el método público isOptimusSolution para determinar si la solución es óptima o no. Cada mejora de solución por el método NextSolution modifica el estado del atributo ciclovertices y las propiedades colMultiplicator, rowMultiplicator, OutputVarindex, InputVarIndex e iteration. Las propiedades rowMultiplicator y colMultiplicator son vectores que informan los valores de los multiplicadores fila y columna respectivamente. Las propiedades OutputVarindex e InputVarindex son índices que indican la posición de la variable que sale y la variable que entran en las listas de soluciones básicas y no básicas según corresponda. La propiedad iteración informa sobre el número de iteraciones que se han hecho para mejorar la solución. El atributo ciclovertices

es una lista que indica cuales son los vértices del polígono de asignación cerrado para el ajuste de la próxima solución.

Clases e interfaces para la exposición del servicio

Figura 27. Detalle de miembros de clase de exposición del servicio para Transport_Service.



Como se comentó en un apartado anterior Transport_Service es la clase principal del Servicio junto con las interfaces que implementa, estas son las que permiten publicar las funcionalidades o recursos a una red. Los métodos más relevantes son:

- getStartParameters[JSON|XML]
- getStartSolution[JSON|XML]
- NextSolution[JSON|XML]
- OptimusSolution[JSON|XML]*

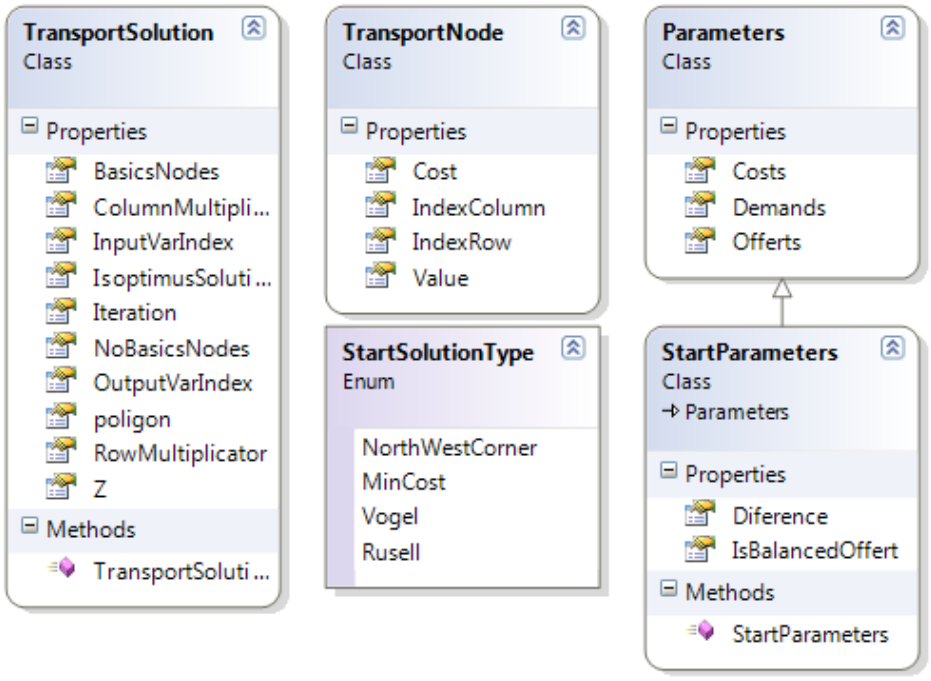
* La notación entre corchetes es una notación gramatical que indica que la frase o frases dentro de los corchetes es opcional. El símbolo | expresa que puede ir o una u otra frase pero no ambas. Por ejemplo getStartParameters[JSON|XML] expresa al conjunto de posibilidades siguientes

Los métodos getStartParameters[JSON|XML] encapsulan los parámetros iniciales Parameters (Ofertas, Demandas, Costos) en un objeto balanceado de la clase StartParameters. Los métodos getStartSolution[JSON|XML] reciben un objeto de la clase Starparameters y el tipo de método de solución de los literales de la enumeración StartSolutionType, estos métodos obtienen la solución factible inicial en un objeto de la clase TransportSolution. Los métodos NextSolution[JSON|XML] reciben un objeto TransportSolution y lo mejoran en un nuevo objeto TransportSolution. Los métodos OptimusSolution[JSON|XML] obtienen la solución óptima en un paso usando la misma lógica del método anterior.

Definición de clases de la capa de acceso a datos

Clases y enumeraciones con contrato de datos

Figura 28. Detalle miembros de clase con contrato de datos para Transport_Service.



Las anteriores clases y enumeraciones son las clases de datos de comunicación, se dice que son de contratos de datos porque estas estructuras son conocidas tanto por el servicio como por los usuarios del servicio. Y son las únicas estructuras complejas que se pueden enviar a través del canal de comunicación. La clase TransportSolution cuenta con listas de objetos TransportNode para expresar la solución de variables básicas (BasicsNodes), y una lista de solución

---

{getStartParameters, getStartParametersJSON, getStartParametersXML}. Los métodos que tienen postfijo JSON o XML se refieren al formato de envió y/o recepción de datos, de no tener posfijo los datos se envían y reciben en un sobre SOAP.

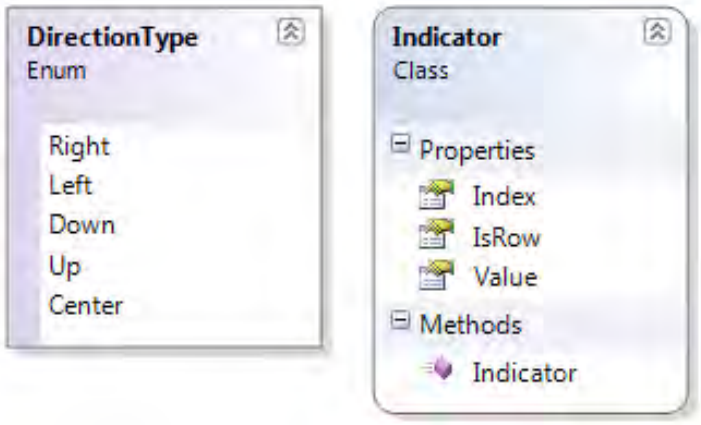
de variables no básicas (NoBasicNodes); listas de multiplicadores fila y columna (ColumnMultiplicators,RowMultiplicators), lista de vértices de variables donadoras y receptoras (polígon), el valor de transporte de la solución básica actual, índices de variables de salida y entrada (OutputvarIndex, InputVarIndex) y el número de iteración de la solución (Iteration). La clase TransportNode encapsula el costo (Cost) de transporte de un origen a un destino (IndexRow, IndexColumn) y el valor a transportar por este nodo (Value). La propiedad IsBalancedOffert determina si el balanceo fue hacia las ofertas o hacia los destinos.

La clase Parameters encapsula los insumos de un problema de transportes, estos son las ofertas e que son la producción de una fuente de origen (Offerst), las demandas que son las solicitudes de productos por un destino (Demands) y los costos de envío de un producto desde los orígenes hasta los destinos. La anterior clase es extendida por la clase StartParameters que adiciona dos propiedades extras para determinar si se adiciono una fuente ficticia por medio de la premisa de que si la diferencia (Diference) entre la sumatoria de ofertas y la sumatoria de las demandas es diferente de 0 entonces los parámetros han sido balanceados*.

Finalmente la enumeración StartSolutionType tiene los literales de los métodos para la obtención de la solución inicial, esto ayuda en gran medida a evitar las ambigüedades de los nombres de los métodos y hace más intuitivo él envió de información para los programadores.

Clases y enumeraciones de datos auxiliares

Figura 29. Detalle de miembros de clase auxiliares para Transport_Service.



Las enumeración DirecionType (ver figura 29) indica el estado de las direcciones de movimiento para conseguir el polígono cerrado de la clase TransportSolution figura 28.

* Balanceados se refiere a que los parámetros se los lleva a un equilibrio, es decir tienen que tener la misma cantidad de elementos (fuentes, destinos).

La clase Indicator es utilizada por la clase StartSolutionByIterationMethods al utilizar el método de Vogel para determinar la mayor penalización (Capitulo 1, sección 1.4.5.3).

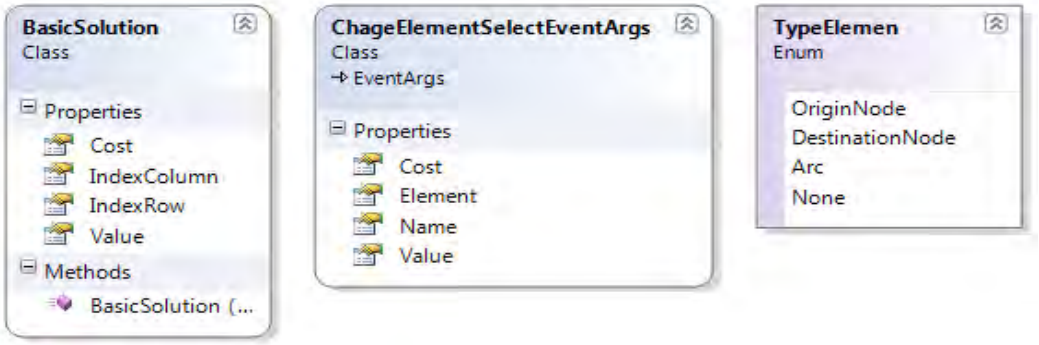
2.5.2.2 Diagrama de clases Transport_ LibraryControls. Como se puede apreciar en la figura 24, Transport_Library LibraryControls está compuesto por tres componentes, por esta razón a continuación, se muestra la estructura para cada uno de estos componentes.

Figura 30. Diagrama de clases de CtrlSisnova

En el diagrama anterior, se observa la estructura del control principal de Transport_Library, la clase principal de este control es CtrlSisnova que hereda de la clase UserControl indicando con ello que es una clase de interface gráfica. Las demás clases son clases para el almacenamiento en memoria de los parámetros iniciales, declaraciones de delegados para la definición de eventos y enumeraciones para indicar el estado y el tipo de elemento para el delegado ChangeElementSelectEventHandler quien en su firma utiliza a la clase ChangeElementSelectEventArgs para proporcionar información sobre los elementos de CtrlSisnova.

Definición clases y enumeraciones de acceso a datos

Figura 31. Detalle de miembros de clases de acceso a datos para CtrlSisnova.



En la anterior figura, se detalla las siguientes clases: BasicSolution y ChangeElementSelectEventArgs, BasicSolution es implementada por la clase CtrlSisnova para requerirla como colección en el método que actualiza el estado actual de la solución. La clase ChangeElementSelectEventArgs hereda de la clase EventArgs y es utilizada para mostrar la información de un elemento seleccionado en la vista de grafo de CtrlSisnova. La enumeración TypeElement tiene literales que son referenciados a través de la propiedad Element de la clase ChangeElementSelectEventArgs.

Delegados para la definición de eventos

Figura 32. Detalle de delegados para CtrlSisnova.

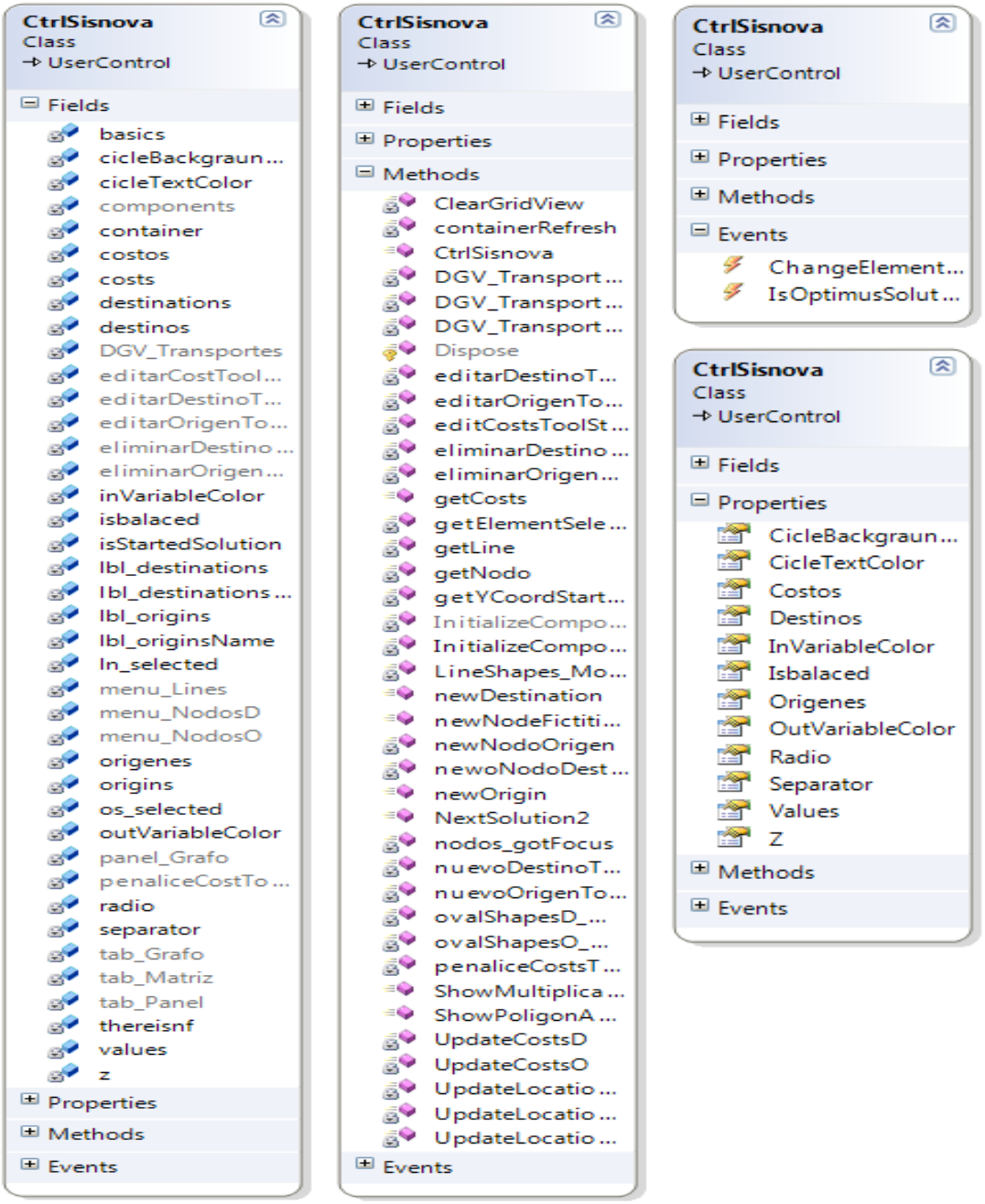
En el diagrama anterior, se presentan los dos delegados que la clase CtrlsSisnova utiliza para la definición de los eventos ChangeElementSelect y IsOptimusSolutión respectivamente y son lanzados cuando ocurre un cambio en la selección de un elemento de grafo (ChangeElementSelect) y cuando se ha llegado a la solución óptima (IsOptimusSolutión). La notación de firmas de delegado es la que implementa .NET para sus propios eventos[*].
ChangeElementSelectEventHandler el sender corresponden a un arco, o nodo de la vista grafo del control y el atributo *e* es un miembro de la clase ChangeElementSelecteventArgs.

[*]Sender por lo general es un parámetro de tipo Object y proporciona al control generador del evento, e es un objeto de un tipo de argumento especificado para cada control cuya función es proporcionar información relacionada con el evento y el control.

Para IsOptimusSolutioneventHandler el sender es la instancia de CtrlSisnova y el atributo *e* es un miembro de la clase EventArgs.

Definición clase CtrlSisnova manejadora de capa de presentación y lógica de negocios

Figura 33. Detalle de miembros de clases manejadoras de la capa de presentación y logica de negocios para CtrlSisnova.



La clase CtlSisnova muestra los atributos que la componen figura 33 (izquierda), sus metodos (centro) y eventos y propiedades (derecha). Los atributos de color más claros como DGV_Transportes o menu_Lines representan a los componentes graficos que lo componen y los atributos de color oscuro los datos que maneja el control en su mayoria son accesidos a traves de sus respectivas propiedades. Los métodos privados son los que el control utiliza internamente para dar completitud a los métodos públicos.

Tabla 10. Descripción miembros públicos de clase de CtrlSisnova.

Tipo miembro de clase	Nombre	Descripción
Metodo	getCosts	Trae una matriz de costos.
Metodo	NewDestination	Proporciona un formulario para la edición de un nuevo destino.
Metodo	NewNodeFictitius	Crea un nodo ficticio origen o destino dependiendo del valor de envio, true es para origen ficticio.
Metodo	NewOrigen	Proporciona un formulario para la edición de un nuevo origen.
Metodo	NextSolution2	Obtiene una nueva solución al enviarle la nueva lista de variables basicas.
Metodo	ShowMultiplicators	Muestra los multiplicadores en la vista matriz del control.
Metodo	ShowPoligonAndInOutVariables	Muestra el poligono y las variables de de entrada y de salida.
Propiedad	CicleBackground	Establece u obtiene el color de los arcos y celdas que conformen el ciclo de variables donadoras y receptoras.
Propiedad	CicleTextColor	Establece u obtiene el color del texto de los arcos y celdas que conformen el ciclo de variables donadoras y receptoras.
Propiedad	Costos	Obtiene una lista de lista de costos.
Propiedad	Destinos	Obtiene una lista de destinos.
Propiedad	InvariableColor	Establece u obtiene el color de la celda o arco de la variable no basica entrante.
Propiedad	IsBalanced	Establece el estado de balanceo del control.
Propiedad	OutVariableColor	Establece u obtiene el color de la celda o arco de la variable basica saliente.
Propiedad	Radio	Establece el radio de los nodos del grafo.
Propiedad	Separator	Establece u obtiene el espacio entre nodos del grafo.
Propiedad	Values	Establece u obtiene los valores asignados a los arcos y celdas del control.
Propiedad	Z	Establece u obtiene el costo de transporte total de envio.

La tabla anterior, resume las utilidades de cada propiedad y métodos públicos del control.

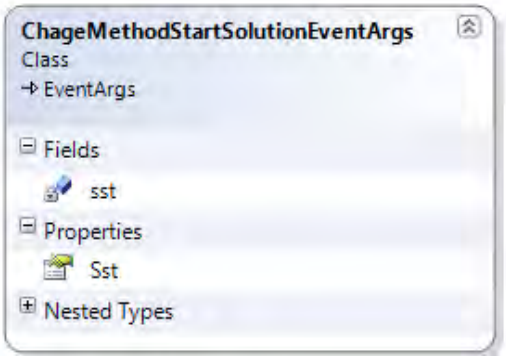
- **Diagrama de clases NavigateTransport**

Figura 34. Diagrama de clases de NavigateTransport.

La clase principal del anterior diagrama es NavigateTransport quien hereda de Usercontrol y se relaciona únicamente con dos delegados de eventos NavigatorEventHandler y ChangeMethodStartSolutionEventHandler. Este control está ideado para complementar al control CtrlSisnova.

Definición de clases de acceso a datos

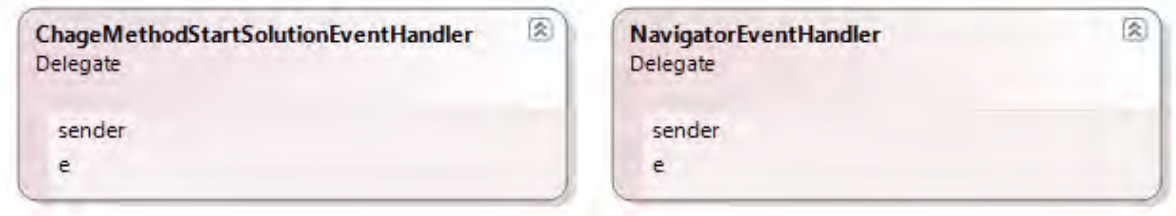
Figura 35. Detalle de miembros de clases de acceso a datos para NavigateTransport.



Esta clase es usada como el atributo de evento e para el delegado ChangeMethodStartSolutionEventHandler y está diseñada para proporcionar el tipo de método para la obtención de la solución inicial factible.

Delegados para la definición de eventos

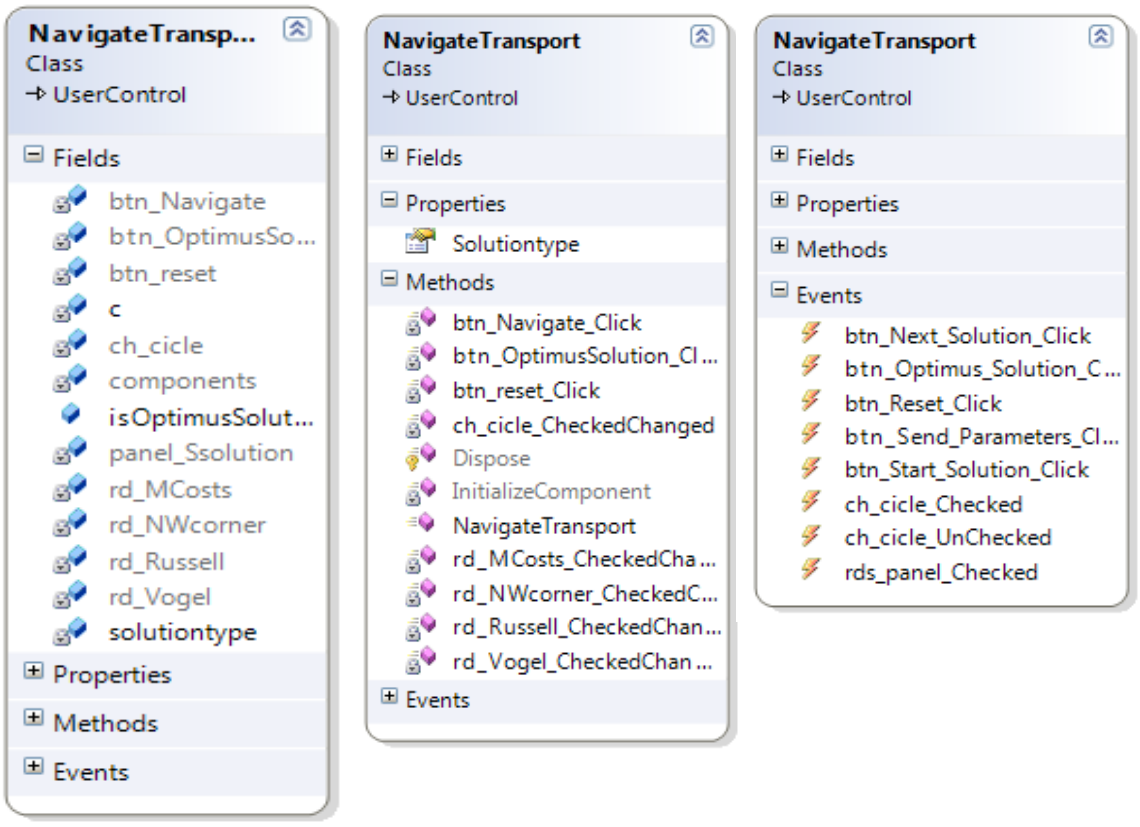
Figura 36. Detalle de delegados para NavigateTransport



El delegado ChangeMethodStartSolutionEventHandler está diseñado para proporcionar un evento que se ejecute cuando el usuario escoja un nuevo método de solución inicial, y NavigatorEventHandler se usa para lanzar eventos sobre el resto de controles que componen a NavigateTransport.

Definición clase NavigateTransport manejadora de capa de presentación y lógica de negocios

Figura 37. Detalle de miembros de clases manejadoras de la capa de presentación y lógica de negocios para NavigateTransport.



El diagrama anterior, sigue la misma especificación de la descripción de la figura 33**,** a continuación se presenta una tabla con los miembros de clase de NavigateTransport.

Tabla 11. Descripción miembros públicos de clase NavigateTransport.

Tipo miembro de clase	Nombre	Descripción
Propiedad	SolutionType	Obtiene el tipo de solución escogida por un usuario.
Evento	btn_NextSolution_Click	Evento de tipo NavigatorEventHandler, enviado al hacer clic sobre el botón NextSolution.
Evento	btn_OptimusSolution_Click	Evento de tipo NavigatorEventHandler, enviado al hacer clic sobre el botón NextSolution.
Evento	btn_Reset_Click	Evento de tipo NavigatorEventHandler, enviado al hacer clic sobre el botón NextSolution.
Evento	btn_Send_Parameters_Click	Evento de tipo NavigatorEventHandler, enviado al hacer clic sobre el botón Send_Parameters.
Evento	btn_Start_Solution_Click	Evento de tipo NavigatorEventHandler, enviado al hacer clic sobre el botón Start_Solution.
Evento	ch_cicle_Checked	Evento de tipo NavigatorEventHandler, enviado al cambiar a verdadero la selección de la caja cicle.
Evento	ch_cicle_UnChecked	Evento de tipo NavigatorEventHandler, enviado al cambiar a falso la selección de la caja cicle.
Evento	rds_panel_Checked	Evento de tipo ChageMethodStartSolutionEventHandler, enviado al cambiar la selección de una caja de radio.

Los miembros mostrados en la tabla 11, son los miembros de clase más relevantes de NavigateTransport.

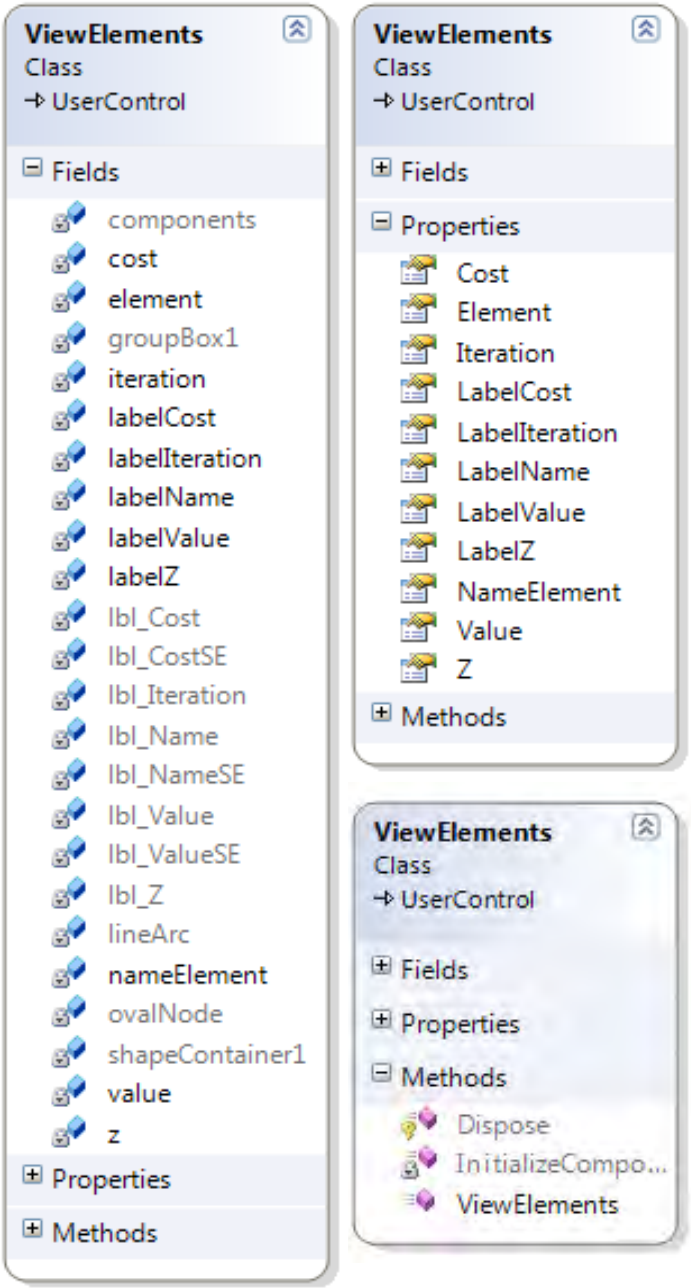
- **Diagrama de clases ViewElements**

Figura 38. Diagrama de clases de ViewElements

En el anterior diagrama, se observa que la clase principal ViewElements especializa a UserControl y usa a la enumeración TypeElement para saber el tipo de elemento a mostrar.

Definición clase ViewElements manejadora de capa de presentación y lógica de negocios

Figura 39. Detalle de miembros de clase manejadora de capad de presentación y lógica de negocios para ViewElements.



Siguiendo la convención de descripción de las figuras 33 y 37, se detalla los miembros de la clase ViewElements en la siguiente tabla.

Tabla 12. Descripción miembros públicos de la clase ViewElements.

Tipo miembro de clase	Nombre	Descripción
Propiedad	Cost	Establece, muestra y obtiene el Costo del elemento seleccionado.
Propiedad	Element	Establece, muestra y obtiene el elemento seleccionado.
Propiedad	Iteration	Establece, muestra y obtiene la iteración actual.
Propiedad	LabelCost	Establece u obtiene la etiqueta de Costo.
Propiedad	LabelIteration	Establece u obtiene la etiqueta de Iteration.
Propiedad	LabelName	Establece u obtiene la etiqueta Nombre.
Propiedad	LabelValue	Establece u obtiene la etiqueta de Valor.
Propiedad	LabelZ	Establece u obtiene la etiqueta de Costo total de envió.
Propiedad	NameElement	Obtiene la etiqueta nombre de elemento.
Propiedad	Value	Establece u obtiene el valor de envió.
Propiedad	Z	Establece u obtiene el costo total de envío.

2.6 COMPORTAMIENTO DEL SISTEMA

A continuación, se presentan los diagramas de actividades y de secuencia para cada uno de los componentes de SISNOVA. En el presente proyecto los diagramas de actividades se han orientado al orden de visión general sobre las actividades de los actores principales de cada componente y los diagramas de secuencia se han orientado a dar una visión de detalle sobre la secuencia de ejecución de funcionalidades de manera que se dé una idea de los pasos que sigue cada funcionalidad y la colaboración entre objetos de las clases presentadas en la sección arquitectura del sistema del presente documento.

2.6.1 Diagramas de actividades

2.6.1.1 Diagrama de actividades Transport _ Service. El diagrama de actividades de la figura 40, da un flujo lógico en cuanto al uso del servicio. En un inicio se llama a la actividad elegir tipo de conexión para que dependiendo de la conexión se instancie para SOAP o REST. Una vez se tiene una instancia de conexión al servicio, el usuario debe elegir la funcionalidad o el recurso a consumir, se envían los parámetros requeridos por la funcionalidad o recurso y se hace una llamada de ejecución.

Figura 40. Diagrama de actividades Transport_Service

Si los parámetros requeridos por la funcionalidad escogida no están instanciados el usuario es quien debe hacer la instanciación correspondiente de los objetos de acceso a datos correspondientes, cada vez que se hace una petición de llamada hacia el servicio este valida si él envió de estos datos por parte del cliente, es correcto. Lo anterior indica que el servicio da la permeabilidad de armar la información en un objeto u obtenerla desde el propio servicio.

Si se desea continuar se puede escoger nuevamente una funcionalidad, de este modo si un usuario quiere saber el resultado de una solución bajo un método diferente al que escogió en un principio este solo tendría que cambiar el método de solución y realizar el llamado y evitar la generación de la instancia de StartParameters, para mejorar una solución simplemente se vuelve a llamar a la función encargada de esta tarea sin ninguna modificación, y la solución óptima se puede obtener en cualquier estado de solución en el que se encuentre la instancia a TransportSolution.

2.6.1.2 Diagrama de actividades Transport_LibraryControls. En el siguiente diagrama, se nota la secuencia de pasos que debe seguirse para el uso de los componentes de la biblioteca de controles de usuario

.

Figura 41. Diagrama de actividades Transport_LibraryControls.

La biblioteca debe ser integrada a la caja de herramientas de Windows Forms, luego se puede disponer de los controles del mismo modo ha como se usan los controle tradicionales, arrastrando los controles al formulario y editando sus propiedades desde la vista diseño, o incorporando el control a través de código. En el caso de que el usuario sea un programador este proporciona las definiciones a los eventos que lanza el control.

2.6.2 Diagramas de secuencia. A continuación se presentan los diagramas de secuencia de las funcionalidades más relevantes de cada componente. Para el componente Transport_Service se enfatiza en visualizar los procedimientos internos de los métodos ejes para la solución del problema de transportes estos son getStartParameters, getStartSolution, NextSolution y getOptimusSolution.

Para Transport_LibraryControls únicamente CtrlSisnova presenta métodos, por esta razón los diagramas de secuencia para este componente son sobre estas operaciones y de ella solo se enmarca las más destacables y cuyo funcionamiento interno lleva intrínseco una moderada complejidad.

2.6.2.1 Diagramas de secuencia Transport_Service

Figura 42. Diagrama de secuencia para el método getStartParameters de Transport_Service.

En el anterior diagrama, la aplicación debe hacer una llamada al servicio solicitando la obtención de la encapsulación y balanceo de los parámetros iniciales (ofertas, demandas, costos) de un planteamiento de problema para dar solución

por medio del método de transportes. Estos parámetros son enviados al servicio, una vez en dentro del servicio (service ver figura 42), este hace una revisión de protección de errores de ejecución[*], mira que las ofertas y demandas no estén vacías y pide colaboración de balanceo a la clase StartParameters instanciándola (ver figura 42) replicando él envió de los parámetros iniciales de la aplicación a esta clase quien hace el respectivo procesamiento y retornar la respuesta al servicio que a su vez retorna a la aplicación.

En la figura 43, se muestra que debe hacerse una llamada al servicio por parte de la aplicación para solicitar la obtención de la solución inicial factible de transporte, este envía petición anteriormente dicha con una instancia a la clase StartParameters (ver figura 28), en la figura 43, esta instancia lleva el nombre sp, esta instancia puede obtenerse por medio del servicio como se explicó en el apartado anterior o manualmente, en cuyo caso será codificada por el desarrollador. Además el desarrollador envía el método por el cual quiere obtener la solución inicial, haciendo uso de los literales[**] de la enumeración StartSolutionType (ver figura 28).

Cuando el servicio recibe las instancias anteriores crea una instancia de la clase StartSolutionMethods con el nombre sss (ver figura 43), con los parámetros enviados por la aplicación, luego el servicio solicita a sss la obtención de la solución inicial en la cual en primera instancia valida que los parámetros recibidos estén correctos, crea una instancia de TransportSolution con el nombre ss, seguido a ello realiza un llamado al procesamiento de obtención de la solución inicial según sea el método escogido para encargarse de esta tarea y retorna el resultado a la instancia sss quien recibe a ss. Empero la respuesta obtenida de esta forma solo llena únicamente las soluciones básicas y no básicas (propiedades BasicNodes y NBasicNodes de TransportSolution, figura 28), para completar las propiedades faltantes, sss instancia un objeto de la clase SolutionByIterationMethods con el nombre sbm, con las soluciones básicas y no básicas anteriormente mencionadas y solicita a sbm el método para obtener la solución (getsolution) quien se encarga de llenar las propiedades faltantes.

Los pasos generales del método getSolution de la clase:

SolutionByIterationMethods se representan en la figura 47.

En la siguiente figura, es notorio que el método getStartSolution invoca a los distintos métodos de solución inicial dependiendo de la solicitud de la aplicación.

---

[*] Esta protección se refiere al control de Excepciones, para el caso en mención puede ocurrir errores en el tratamiento de los arreglos de ofertas, demandas o costos.

[**] Para una enumeración un literal es una palabra que representa un valor numérico constante, para la enumeración StartSolutionType estos valores son NorthWestCorner, MinimunCost, Vogel y Russell.

Figura 43. Diagrama de secuencia para el método getStartSolution de Transport_Service.

En la figura 44, la aplicación debe solicitar al servicio la siguiente solución enviando una instancia a la solución actual, el servicio recibe la petición y comprueba si la instancia de solución enviada es óptima en cuyo caso retorna a la aplicación la misma instancia de envió, caso contrario el servicio crea una instancia de SolutionByiterationMethods con el nombre s, enviándole las listas de solución básicas y no básicas actuales, luego el servicio solicita de la instancia s la siguiente solución este la mejora y obtiene la nueva solución factible.

Figura 44. Diagrama de secuencia para el método NextSolution de Transport_Service.

Pero este método únicamente modifica las listas de variables básicas y no básicas en base a la variable básica de salida y la variable no básica de entrada de la solución actual, esto es la variable básica de salida debe salir de la lista de variables básicas y pasar a la lista de variables no básicas, y la variable no básica de entrada sustituye el espacio de la variable básica de salida (ver figura 48), pero el resto de propiedades aún siguen iguales. Esta situación es similar a la que se presentaba en la figura 43, por ello se procede de igual forma una llamada al método obtener solución de (getSolution), se crea una instancia de recepción de la clase TransportSolution en que se almacena la respuesta de dicho método y se retorna la respuesta a la aplicación.

Figura 45. Diagrama de secuencia para el método OptimusSolution de Transport_Service.

Las pautas para solicitar la llamada a OptimusSolution por parte de la aplicación son las mismas que el método NextSolution planteado en la figura 44, la diferencia entre los dos está en la implementación del método en la clase SolutionByIterationMethods, en ese punto se hace una llamada cíclica al método NextSolution hasta que la solución sea optima como se muestra en la figura 46.

El ciclo de la figura 46, además de obtener la Solución óptima puede servir como recurso a ser implementado por una aplicación, para ver la resolución del problema de transportes paso a paso.

Figura 46. Diagrama de secuencia para el método getOptimusSolution de la clase SolutionByIterationMethods.de Transport_Service

Los siguientes diagramas dan un grado de detalle más profundo sobre los métodos más relevantes enunciados en los anteriores diagramas de secuencia, todos ellos son métodos de la clase SolutionByIterationMethods.

Figura 47. Diagrama de secuencia para el método getsolution de la clase SolutionByIterationMethods.de Transport_Service.

En la anterior figura, se aprecia la interacción entre objetos de las clases: TransportSolution y TransportNode, con el objetivo de proveer el armazón de una solución de transportes. Amparándose en un método clave esquematizado en la siguiente figura.

Figura 48. Diagrama de secuencia para el método InitAtributes SolutionByIterationMethods.de Transport_Service.

En la figura 48, se ve el proceso para inicializar los atributos principales del método de transportes, como son: variables básicas, variables no básicas, multiplicadores, ciclo cerrado y las respectivas variables de entrada y salida.

Figura 49. Diagrama de secuencia para el método NextSolution de la clase SolutionByIterationMethods.de Transport_Service.

2.6.2.2 Diagramas de secuencia Transport_LibraryControls

Figura 50. Diagrama de secuencia métodos newOrigen,newDestination, newNodeFicticius de CtrlSisnova.

En la figura anterior, se observa 3 diagramas de secuencia todos con el objetivo de crear nuevos nodos. Los diagramas de la izquierda y centro modelan el comportamiento de CtrlSisnova cuando se solicita crear un nuevo nodo* origen o destino, para realizar esta tarea se realizan unas validaciones previas para saber el estado de CtrlSisnova, si el control está en o sobre la etapa de solución factible inicial y no ha sido balanceado entonces permite realizar las siguientes acciones.

Si el control pasa las validaciones iniciales instancia y muestra un dialogo de edición, en este dialogo de edición se editan los valores de oferta o demanda, y nombres del origen o destino. Si se pulsa sobre los botones ok y cancelar el dialogo devuelve el foco a CtrlSisnova, si el botón pulsado es ok se obtiene una instancia a un nodo (origen en el caso del diagrama izquierda, destino en el caso diagrama centro), se recalcula la posición de todos los nodos origen o destino incluidos hasta el momento, se actualizan las localizaciones de los arcos, y se refresca las vistas del control.

El diagrama de secuencia de la figura 50, situado a la derecha, debe hacer las validaciones enunciadas en el párrafo anterior, valida si el nodo a insertar es origen o destino y luego sigue los mismos pasos anteriores sin mostrar diálogos de edición ya que el valor de balanceo es calculado a través de las ofertas y demandas vigentes.

En la figura 51, muestra los pasos que debe realizar el control CtrlSisnova para mostrar una solución básica factible, en principio recorre todos los arcos en forma matricial, si el elemento de recorrido (i,j) coincide con una posición de la lista de soluciones básicas entonces genera un arco para esa posición, caso contrario coloca un arco nulo.

Al finalizar el recorrido se llama al método clearGridView quien conoce a la matriz anterior por ser atributo de clase, y limpia los valores básicos de celda. Inmediatamente después de refrescar la vista matriz, se refresca el contenedor para poder visualizar los resultados de la solución para el grafo y la matriz del control.

* Un nodo representa a un origen o destino, este nodo puede ser representado como un círculo en la vista grafo del control o una nueva fila o columna en la vista matriz.

Figura 51. Diagrama de secuencia método NextSolution de CtrlSisnova

Es importante destacar que en la figura 51, los dos bucles anidados iniciales permiten dibujar la siguiente solución factible, en forma de grafo y matriz.

Figura 52. Diagrama de secuencia método ShowPoligon de CtrlSisnova

En la figura anterior, se mira los pasos que debe seguir el control para mostrar las indicaciones de las variables de entrada, variables de salida y el polígono en el control CtrlSisnova. Para ello se debe hacer un recorrido del polígono y comparar con la matriz de arcos[*], cuando el dato en la posición de recorrido es no nulo se cambia el color del arco del grafo y la celda de la matriz[**] , en este punto puede ocurrir una doble coincidencia ya que el arco puede ser un arco para la variable que sale en cuyo caso el arco y la celda cambiaran al color de variable que sale definido en las propiedades del control. Si el arco encontrado es null en esta posición se debe agregar un nuevo arco (LineShape ver figura 52) y la respectiva modificación de color tanto al arco nuevo como a la celda en esa posición. Se hace una limpieza al contenedor del grafo (container.Shapes.clear() figura 52) y se refresca con los nuevos cambios.

[*] Recuerde que en la descripción de la figura 34 se especificó que las variables no básicas se colocaban como datos null en la matriz (costs).

[**] Esta matriz hace referencia al control gridView de la vista matriz de CtrlSisnova.

2.7 ESPECIFICACIONES DE IMPLEMENTACIÓN DE SISNOVA

2.7.1 Especificaciones de implementación Transport _Service

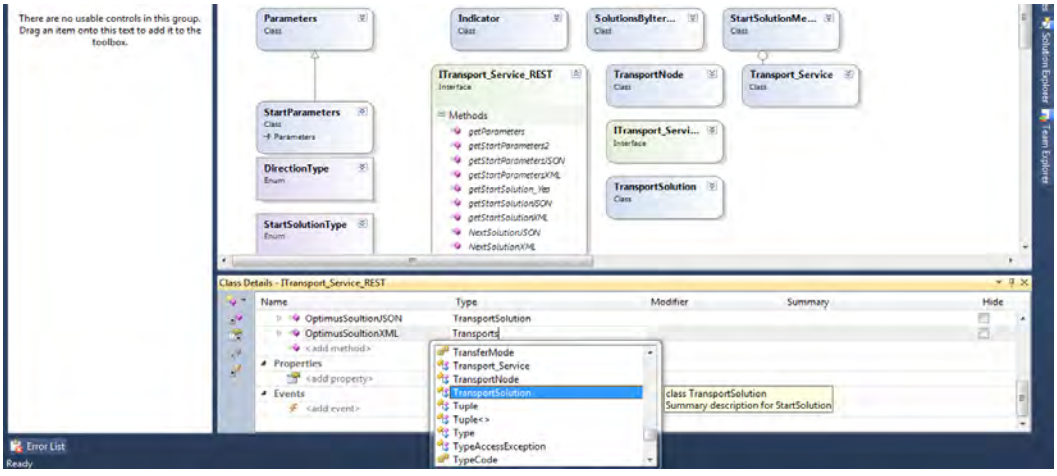
Teniendo en cuenta la discriminación de clases en el modelo estructural* especificado en la arquitectura (Capitulo 2, sección 2.6) y la tecnología escogida para el desarrollo de Transport_Service sigue un plan de codificación de definición de código siguiendo los siguientes pasos:

1. Actualizar el código en base al último documento de arquitectura.
2. Colocar los atributes* correspondientes a las estructuras de datos según discriminación.
3. Generar los métodos de implementación en la clase de servicio (Declaración).
4. Realizar las acciones correspondientes en el archivo de configuración del servicio.
5. Realizar los algoritmos que cumplan con el objetivo de los métodos (Definición).

A continuación, se muestra los resultados de la última iteración de codificación siguiendo el orden establecido anteriormente.

2.7.1.1 Actualización de código en base al último documento de arquitectura

Figura 53. Edición de la interface ITransport_Service_Rest en tiempo de diseño en Visual Studio.



*En el modelo estructural las clases se clasificaron en clases de lógica de negocio, de exposición al servicio, con contratos de datos y auxiliares (figuras 26, 27, 28, 29).

*Un atribute en WCF es el encabezado entre corchetes que se coloca a una clase, interface, enumeración, excepción, evento, atributo de clase o propiedad de clase. Para suscribir los metadatos del servicio, es decir la información que se va exponer al cliente.

La figura 53, muestra la implementación de la estructura arquitectónica en tiempo de diseño*, esto es una buena práctica ya que es más intuitivo reflejar diagrama a diagrama que diagrama código.

Figura 54. Implementación de la arquitectura de Transport_Service.

En la última iteración la interface que esta expandida es la única modificación de código a realizar para cubrir el primer punto del plan de codificación, la figura 54, muestra el resultado final.

2.7.1.2 Colocar los atributes correspondientes a las estructuras de datos

Para entender mejor de que trata este punto, a continuación se muestra los atributes para la enumeración StartSolutionType, la clase TransportNode y ITranport_Service_SOAP ya que en la interface ITransport_Service_REST se realizan acciones adicionales y no se contemplan algunos atributes debido al tipo de comportamiento de exposición del servicio. Las siguientes tablas muestran la clasificación de los elementos estructurales de la arquitectura con los atributes que deben adicionarse.

* Visual Studio ofrece la posibilidad de generar código estructural en tiempo de diseño.

Tabla 13. Guía de diseño de elementos estructurales por tipo, clasificación y atribute.

Nombre del elemento estructural	Tipo elemento estructural	Clasificación	Atributes a especificar
StartSolutionType	Enumeración	Contratos de datos	DataContract
TransportNode	Clase	Contratos de datos	DataContract
ITranport_Service_SOAP	Clase	Exposición al servicio	ServiceContract
Itransport_Service_REST	Clase	Exposición al servicio	ServiceContract

Tabla 14. Guía de diseño de tipo de elemento estructural por clasificación, miembros y atributes.

Tipo elemento estructural	Clasificación	Tipo de miembro con atributes	Atributes a especificar
Enumeración	Contratos de datos	Literal	EnumMember
Clase	Contratos de datos	Atributo, Propiedad	DataMember
Clase	Exposición al servicio	Metodo	OpertationContract

En la siguiente figura, se muestra como seguir por codigo las especificaciones hechas en las tablas 13 y 14.

Figura 55. Código Fuente en C# de StartSolutionType, TransportNode y ITransport_Service_SOAP con sus respectivos atributes.

```
[DataContract]
public enum StartSolutionType
{
    [EnumMember]
    NorthWestCorner,
    [EnumMember]
    MinCost,
    [EnumMember]
    Vogel,
    [EnumMember]
    Rusell,
}

[DataContract]
public class TransportNode
{
    [DataMember]
    public int IndexRow { get; set; }
    [DataMember]
    public int IndexColumn { get; set; }
    [DataMember]
    public double Cost { get; set; }
    [DataMember]
    public double Value { get; set; }
}


[ServiceContract]
public interface ITransport_Service_SOAP
{

    [OperationContract]
     StartParameters getStartParameters(List<double> offerst, List<double>
     demands, double[][] costs);
    [OperationContract]
     TransportSolution getStartSolution(StartParameters startparameters,
     StartSolutionType type);
    [OperationContract]
    TransportSolution NextSolution(TransportSolution t);
    [OperationContract]
    TransportSolution OptimusSoultion(TransportSolution t);
}
```

Pero como se mencionó anteriormente, queda faltando la especificación en el codigo para Itransport_Service_REST ya que para la exposición de un servicio REST hay que tener en cuenta el diseño de URIs y el método HTTP de exposición del recurso.

La meta propuesta en ese punto es resolver la forma de exposición de las funcionalidades que necesitan enviar datos complejos (diferentes a cadena de caracteres) aprovechando la infraestructura construida hasta el momento para el consumo por SOAP y sin uso de almacenamiento de datos en el servidor cuando en la solución arquitectónica no hay contemplado almacenamiento de datos persistente.

La tabla siguiente, es la solución propuesta para este tipo de problemas.

Tabla 15. Método de Interface por método HTTP y URI.

Método de interface	Método http	URI
NextSolutionJSON	POST	/NextSolutionJSON
NextSolutionXML	POST	/NextSolutionXML
OptimusSoultionJSON	POST	/OptimusSoultionJSON
OptimusSoultionXML	POST	/OptimusSoultionXML
getParameters	GET	/Parameters
getStartParameters	GET	/StartParameters
getStartParametersJSON	POST	/StartParametersJSON
getStartParametersXML	POST	/StartParametersXML
getStartSolution	GET	/StartSolution?solutionmethod={TYPE}
getStartSolutionJSON	POST	/StartSolutionJSON?solutionmethod={TYPE}
getStartSolutionXML	POST	/StartSolutionXML?solutionmethod={TYPE}

En la tabla anterior, se especifica el diseño de URIS de exposición para los métodos de la interface Itransport_Service_REST. Los métodos que se exponen por GET hacen referencia a un problema inscrito con almacenamiento en el servidor.

Atendiendo al diseño de la tabla 15, le sigue adicionar a las tablas 13 y 14 atributos WebInvoke si es un método http no get, o WebGet si es un método http get. Luego determinar los tipos de formatos de los mensajes de envió respuesta.

La siguiente tabla, reúne las especificaciones de cada método.

Tabla 16* Guía de diseño para implementación de métodos de interface en REST.

Método de interface	Método http	***Atributo adicional***	***Formato envío***	***Formato recepción***
NextSolutionJSON	POST	WebInvoke	JSON	JSON
NextSolutionXML	POST	WebInvoke	XML	XML
OptimusSoultionJSON	POST	WebInvoke	JSON	JSON
OptimusSoultionXML	POST	WebInvoke	XML	XML
getParameters	GET	WebGet	JSON	XML
getStartParameters2	GET	WebGet	JSON	XML
getStartParametersJSON	POST	WebInvoke	JSON	JSON
getStartParametersXML	POST	WebInvoke	XML	XML
getStartSolution_Yea	GET	WebGet	JSON	JSON
getStartSolutionJSON	POST	WebInvoke	JSON	JSON
getStartSolutionXML	POST	WebInvoke	XML	XML

El atributo WebInvoke lleva la siguiente sintaxis:

[WebInVoke(Method = "Método http",
BodyStyle = WebMessageBodyStyle.Bare,
RequestFormat = WebMessageFormat.Formato envío,
ResponseFormat = WebMessageFormat.Formato recepción,
UriTemplate = "URI")]

El atributo WebGet lleva la siguiente sintaxis:

[WebGet(BodyStyle = WebMessageBodyStyle.Bare,
RequestFormat = WebMessageFormat.Formato envio,
ResponseFormat WebMessageFormat.Formato recepción,
UriTemplate = " URI ")]

Los elementos de la sintaxis que están en cursiva y subrayados tienen su correspondencia en los encabezados de las tablas No 15 y 16. Siguiendo todas las indicaciones de las tablas de la 13 a la 16, se obtienen los siguientes resultados.

Figura 56. Código fuente en C# de ITransport_ServiceREST con sus respectivos atributes de consumo para REST.

```
[ServiceContract]
public interface ITransport_Service_REST
{
    [OperationContract]
    [WebGet(BodyStyle=WebMessageBodyStyle.Bare,RequestFormat=WebMessageFormat.Json,
             ResponseFormat=WebMessageFormat.Xml,UriTemplate="/Parameters")]
    Parameters getParameters();
    [OperationContract]
    [WebGet(BodyStyle=WebMessageBodyStyle.Bare,RequestFormat=WebMessageFormat.Json,
             ResponseFormat=WebMessageFormat.Xml,UriTemplate="/StartParameters")]
    StartParameters getStartParameters2();
    [OperationContract]
    [WebGet(BodyStyle=WebMessageBodyStyle.Bare,RequestFormat=WebMessageFormat.Json,
             ResponseFormat=WebMessageFormat.Json,
             UriTemplate = "/StartSolution?solutionmethod={type}")]
    TransportSolution getStartSolution_Yea(String type);
    [OperationContract]
    [WebInvoke(Method="POST", BodyStyle=WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Json,ResponseFormat = WebMessageFormat.Json,
             UriTemplate = "/StartParametersJSON")]
    StartParameters getStartParametersJSON(Parameters param);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Json,ResponseFormat = WebMessageFormat.Json,
             UriTemplate = "/StartSolutionJSON?solutionmethod={type}")]
    TransportSolution getStartSolutionJSON(String type, StartParameters startparameters);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
             UriTemplate = "/NextSolutionJSON")]
    TransportSolution NextSolutionJSON(TransportSolution t);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat =   WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
             UriTemplate = "/OptimusSoultionJSON")]
    TransportSolution OptimusSoultionJSON(TransportSolution t);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Xml, ResponseFormat = WebMessageFormat.Xml,
             UriTemplate = "/StartParametersXML")]
    StartParameters getStartParametersXML(Parameters param);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Xml, ResponseFormat = WebMessageFormat.Xml,
             UriTemplate = "/StartSolutionXML?solutionmethod={type}")]
    TransportSolution getStartSolutionXML(String type, StartParameters startparameters);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Xml, ResponseFormat = WebMessageFormat.Xml,
             UriTemplate = "/NextSolutionXML")]
    TransportSolution NextSolutionXML(TransportSolution t);
    [OperationContract]
    [WebInvoke(Method = "POST", BodyStyle = WebMessageBodyStyle.Bare,
             RequestFormat = WebMessageFormat.Xml, ResponseFormat = WebMessageFormat.Xml,
             UriTemplate = "/OptimusSoultionXML")]
    TransportSolution OptimusSoultionXML(TransportSolution t);
}
```

2.7.1.3 Generar los métodos de implementación en la clase de servicio (Declaración). En la figura 27, se muestra como la clase principal Transport_Service implementa los métodos de las interfaces:

Itransport_Service_SOAP y Itransport_Service_REST, el resultado en el código se muestra a continuación.

Figura 57 Código fuente C# de implementación de las interfaces en la clase Transport_Service.

```
public class Transport_Service : ITransport_Service_SOAP,ITransport_Service_REST
{
    //Soap Methods
    public StartParameters getStartParameters(List<double> offerst, List<double> demands, double[][] costs)...
    public TransportSolution getStartSolution(StartParameters startparameters, StartSolutionType type)...
    public TransportSolution NextSolution(TransportSolution t)...
    public TransportSolution OptimusSoultion(TransportSolution t)...
    public bool isOptimusSolution(TransportSolution t)...
    //Rest Methods ...
    public StartParameters getStartParametersJSON(Parameters param)...
    public TransportSolution getStartSolutionJSON(string type, StartParameters startparameters)...
    public TransportSolution NextSolutionJSON(TransportSolution t)...
    public TransportSolution OptimusSoultionJSON(TransportSolution t)...
    //XML FORMATER
    public StartParameters getStartParametersXML(Parameters param)...
    public TransportSolution getStartSolutionXML(string type, StartParameters startparameters)...
    public TransportSolution NextSolutionXML(TransportSolution t)...
    public TransportSolution OptimusSoultionXML(TransportSolution t)...
    public Parameters getParameters()...
    public StartParameters getStartParameters2()...
    public TransportSolution getStartSolution_Yea(String type)...
}
```

2.7.1.4 Realizar las acciones correspondientes en el archivo de configuración del servicio. Para que las interfaces anteriores (SOAP, REST) se puedan exponer al tiempo, hay que realizar las siguientes configuraciones en el archivo de configuración del servicio (Web.config).

1. Crear 2 puntos de acceso (endpoints) para las peticiones REST y SOAP.
2. Asociación de comportamiento (Behavior) a los puntos de acceso (endpoints).

Tabla 17. Resultado de análisis para configuración de Transport_Service

Tipo de servicio	Encuadernación (binding)*	Nombre del endpoint	Nombre de interface	Enlace a comportamiento (behavior)
REST	WebHttpBinding	WCFRestEndpoint	ITransport_Service_REST	WCFRestEndpointBehavior
SOAP	basicHttpBinding	WCFSoapEndpoint	ITransport_Service_SOAP	WCFSoapEndpoint Behavior

* Los tipos de bindings se estudian en el Capítulo 1, sección 1.5.1.2

En la figura siguiente, se aplica el análisis hecho en la tabla 17 dentro del archivo de configuración del servicio Transport_Service.

Figura 58. Código fuente XML archivo de configuración Web.config de Transport_Service

```
<?xml version="1.0"?>
<configuration>
	<system.web>
		<compilation debug="true" targetFramework="4.0">
			<assemblies>
				<add assembly="mscorlib, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/></assemblies></compilation>
	</system.web>
	<system.serviceModel>
		<!--Configuración del servicio y sus interfaces-->
		<bindings/>
		<client/>
		<services>
			<service name="Transport_Service"
behaviorConfiguration="ServiceBehavior">
				<!--Endpoint (punto de acceso) para peticiones REST-->
				<endpoint address="rest" binding="webHttpBinding"
name="WCFRestEndpoint" contract="ITransport_Service_REST"
behaviorConfiguration="WCFRestEndpointBehavior"/>
				<!--Endpoint (punto de acceso) para peticiones SOAP-->
				<endpoint address="soap" binding="basicHttpBinding"
name="WCFSoapEndpoint" contract="ITransport_Service_SOAP"
behaviorConfiguration="WCFSoapEndpointBehavior"/>
			</service>
		</services>
		<behaviors>
			<!--Asociación de behaviors (comportamiento) a los endpoints
respectivos (soap rest)-->
			<endpointBehaviors>
				<!--Behavior para el Endpoint de peticiones REST-->
				<behavior name="WCFRestEndpointBehavior">
					<webHttp helpEnabled="true"/>
				</behavior>
				<!--Behavior para el Endpoint de peticiones SOAP-->
				<behavior name="WCFSoapEndpointBehavior"/>
			</endpointBehaviors>
			<serviceBehaviors>
				<behavior name="ServiceBehavior">
					<!-- To avoid disclosing metadata information, set
the value below to false and remove the metadata endpoint above before deployment --
>
					<serviceMetadata httpGetEnabled="true"/>
					<!-- To receive exception details in faults for
debugging purposes, set the value below to true.  Set to false before deployment to
avoid disclosing exception information -->
					<serviceDebug
includeExceptionDetailInFaults="false"/>
				</behavior>
			</serviceBehaviors>
		</behaviors>
		<serviceHostingEnvironment multipleSiteBindingsEnabled="true"/>
	</system.serviceModel>
	<system.webServer>
		<modules runAllManagedModulesForAllRequests="true"/>
	</system.webServer>
</configuration>
```

La configuración de los puntos de acceso que se muestra en la figura 58, sigue la siguiente sintaxis de correspondencia con base a los encabezados de la tabla 17.

Sintaxis para un punto de acceso:

<endpoint address="dirección"
binding=" *Encuadernación* "
name=" *Nombre del endpoint*"
contract=" *Nombre de interface*"
behaviorConfiguration=" *Enlace a comportamiento* "/>

El enlace a comportamiento como se observa en el archivo de configuración debe llevar el mismo nombre que el especificado en el punto de acceso.

2.7.1.5 Realizar los algoritmos que cumplan con el objetivo de los métodos (Definición). Para la definición de los métodos de las clases especificadas en la arquitectura se partió por codificar las clases de lógica de negocio SolutionsByIterationsMethods y StartSolutionMethods (ver figura 43). A partir de ellas se incorpora las funcionalidades establecida para el servicio expuesto por la clase Transport_Service.

Este paso se comprueba en un entorno funcional y por ello se han realizado prototipos de prueba, estos se ven con mayor profundidad más adelante. Empero se realizaron pruebas del servicio previas a la construcción de los prototipos y se muestran a continuación.

Pruebas del componente Transport _Service en exploradores Web

Prueba de hospedaje

Esta prueba tiene la finalidad de ver si el servicio está instalado en la Web. Para ello se debe colocar en la barra de direcciones del explorador la url del servicio Web. La URL del servicio es:

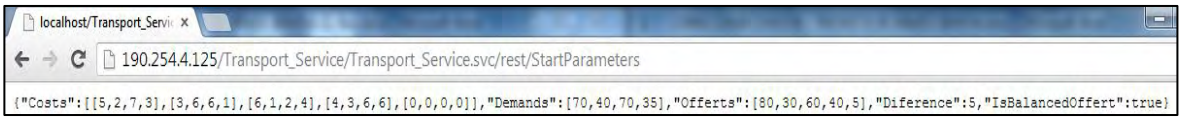
http://190.254.4.125/Transport_Service/Transport_Service.svc y en la figura siguiente, se muestra la imagen que debe aparecer en el explorador Web.

Figura 59 Comprobación de hospedaje del servicio Transport_Service en un explorador Web.

Prueba de funcionalidad

Solicitar los parámetros iniciales mediante REST en la URI *http://190.254.4.125/Transport_Service/Transport_Service.svc/rest/StartParameters* este es el resultado de la solicitud.

Figura 60. Resultado JSON de petición StartParametersJSON en un explorador Web.



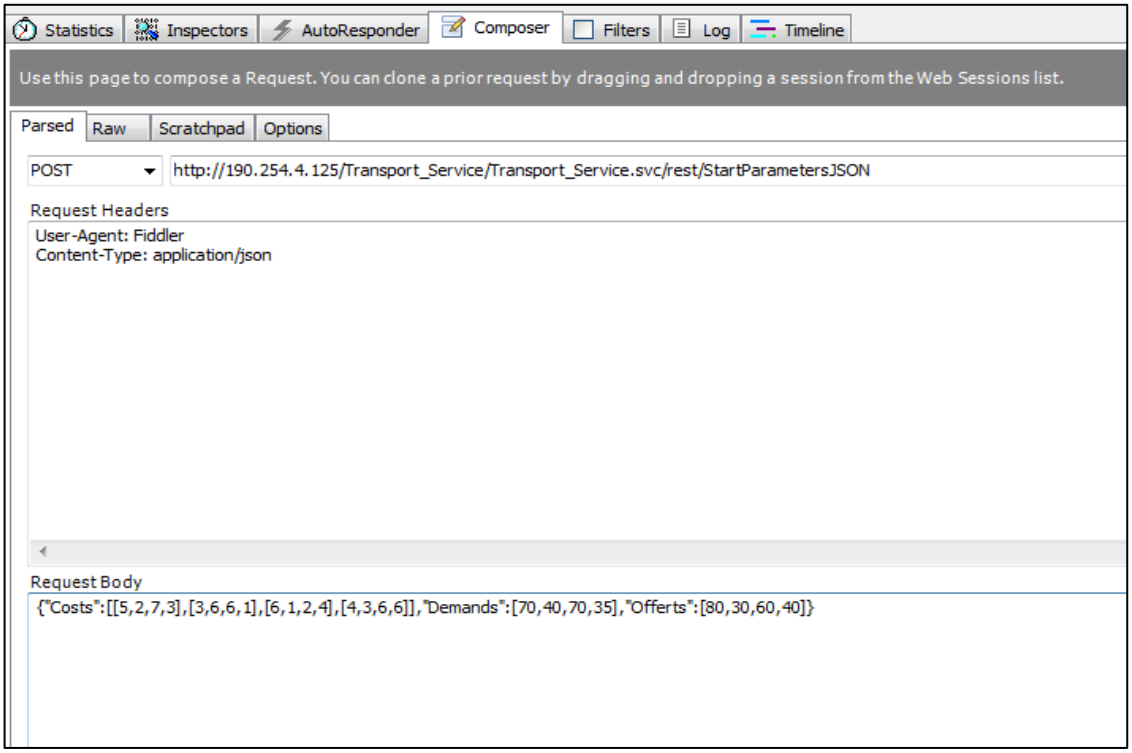
Pruebas del componente Service_Transport con fiddler2

Fiddler es una aplicación proxy o puente para depuración de programas que utilicen el protocolo HTTP. En ella solo se hace pruebas de funcionalidad.

Para realizar la prueba es necesario contar con la representación de los parámetros de un problema en formato JSON, como se muestra a continuación.

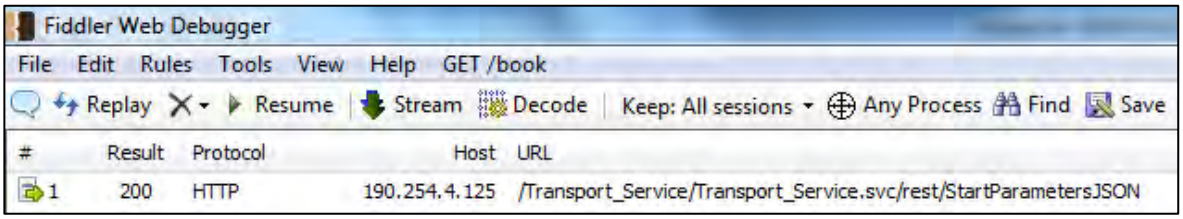
{"Costs":[[5,2,7,3],[3,6,6,1],[6,1,2,4],[4,3,6,6]],"Demands":[70,40,70,35],"Offerts":[80,30,60,40]}

Figura 61. Preparación de solicitud de recurso StartParametersJSON en fiddler



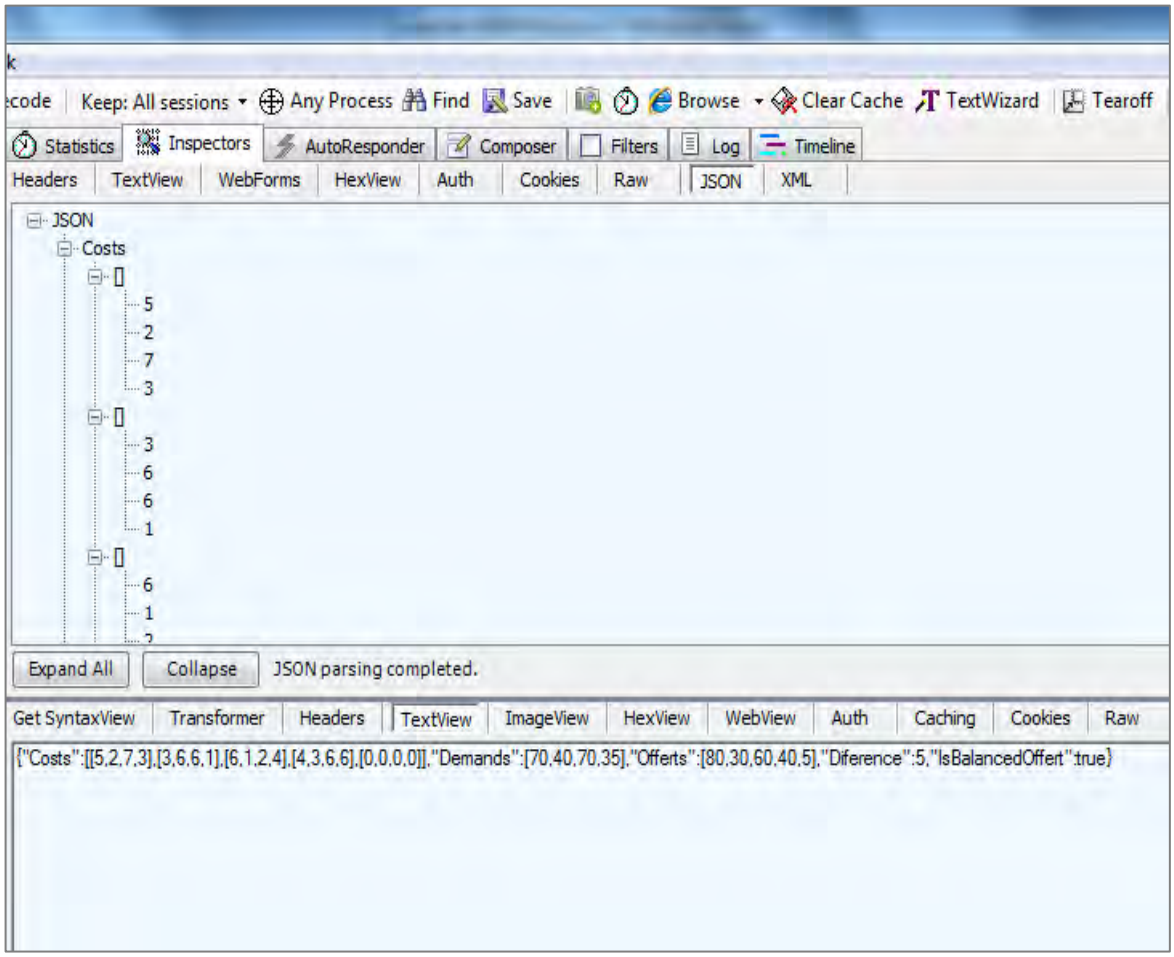
La figura 61, muestra cómo hacer la prueba del servicio con los parámetros iniciales, se observa que al lado izquierdo de la barra de direcciones esta una lista desplegable en la que se ha seleccionado como método http a el método POST, en la barra de direcciones esta la URI de prueba, y en el panel inferior (Request Body) el problema formateado en JSON.

Figura 62. Archivo de repuesta de la petición panel izquierdo de fiddler



Cuando se ejecuta la petición de la figura 61, en el panel izquierdo Fiddler muestra el formulario de respuesta en verde, significando esto, que no ha ocurrido errores. Si se da doble click en el icono mencionado fiddler da una vista de la respuesta del servicio.

Figura 63. Respuesta de la petición panel inferior de fiddler.



La figura 63, muestra la interface de respuesta que presenta fiddler, el panel inferior en este caso es de respuesta (Response) y en el se ve que la solución trae al objeto JSON siguiente.

{"Costs":[[5,2,7,3],[3,6,6,1],[6,1,2,4],[4,3,6,6],[0,0,0,0]],"Demands":[70,40,70,35],"Offerts":[80,30,60,40,5],"Diference":5,"IsBalancedOffert":true}.

En esta respuesta se puede apreciar que se ha agregado una oferta ficticia con un valor de 5 el cual corresponde a las diferencias de las sumas de las ofertas y demandas del planteamiento inicial. También se observa que en la matriz de costos hay una nueva fila de 0 concluyendo de esta forma que el resultado de la llamada al método es correcto.

2.7.2 Especificaciones de implementación Transport_LibraryControls. Al igual que para Service_Transport para la implementación de este componente se siguen una serie de pasos similares enumerados a continuación.

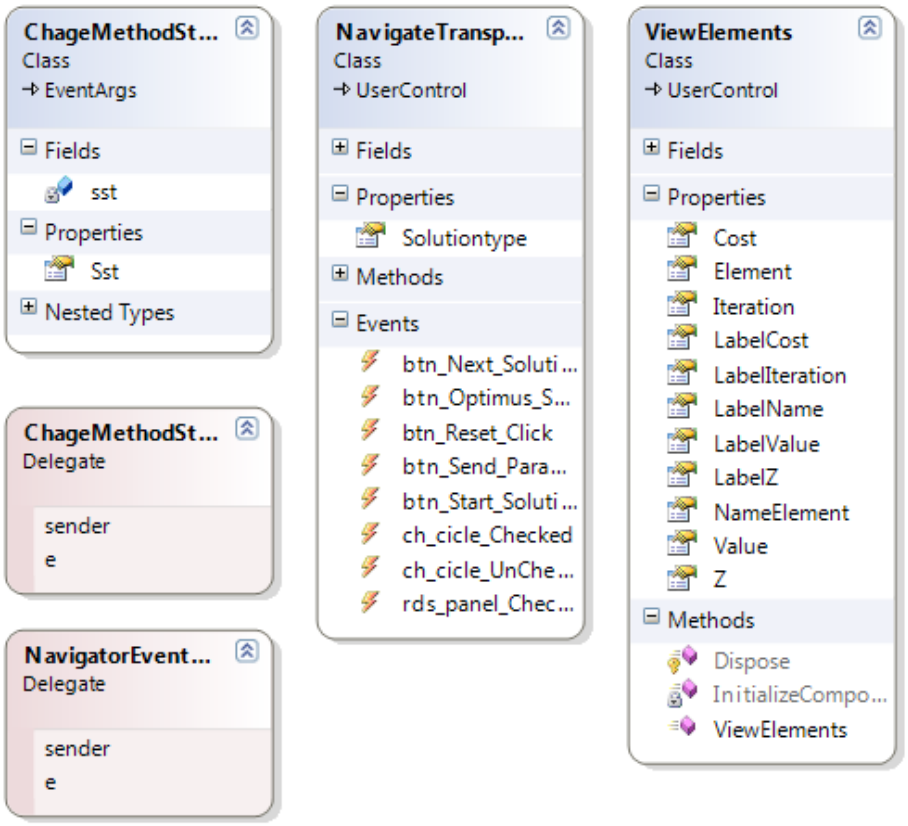
1. Actualizar el código en base al último documento de arquitectura.
2. Diseñar elementos de interfaz gráfica.
3. Realizar los algoritmos que cumplan con el objetivo de los métodos de cada control.

2.7.2.1 Actualizar el código en base al último documento de arquitectura

Figura 64. Implementación de la arquitectura de CtrlSisnova.

En la anterior figura, se ve reflejado el diagrama estructural figura 30 planteado para CtrlSisnova, los elementos aparecen colapsados debido a que el en la ultima iteración este componente ya esta terminado. En la siguiente figura, se ve que los elementos estan en implementación y coinciden con la estructura propuesta en las figuras 34 y 38.

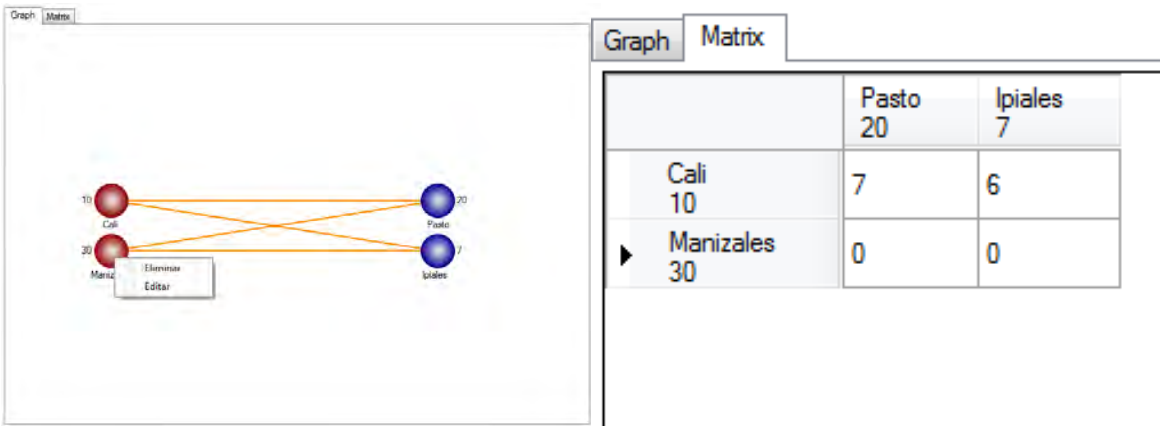
Figura 65. Implementación de la arquitectura de NavigateTransport y ViewElements.



2.7.2.2 Diseñar elementos de interfaz gráfica.

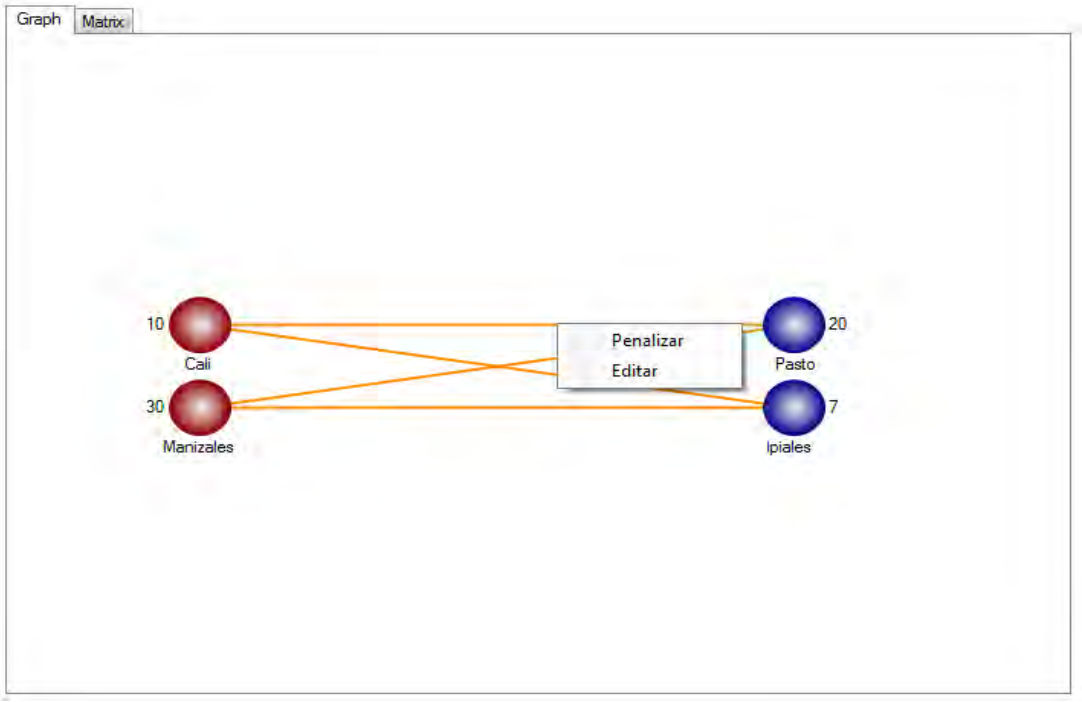
Interface gráfica para CtrlSisnova

Figura 66. GUI CtrlSisnova vistas del control y menú emergente de edición de nodos.



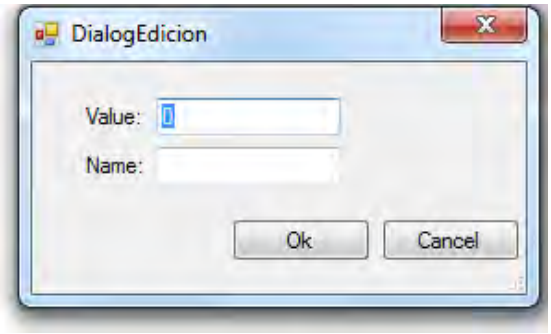
En la anterior figura, se observa el diseño de el control CtrlSisnova, proponiendo 2 vistas, una de grafo (Graph) y otra vista matriz (Matrix). En la vista grafo se ve el diseño de un menu emergente para editar o eliminar un nodo.

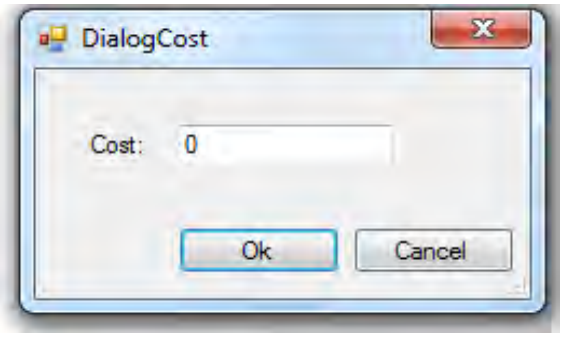
Figura 67. GUI CtrlSisnova vista de menu emergente de edición de arcos.



Igualmente, el control presenta un menu emergente para los arcos del grafo dando la opción de penalizar o editar figura 67.

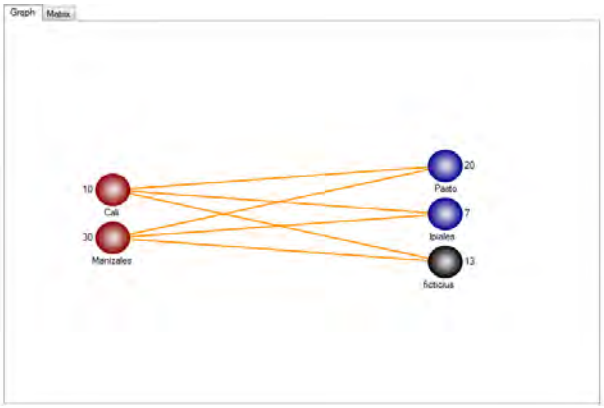
Figura 68. GUI CtrlSisnova vista de dialogos de edición ofertas,demandas y costos.





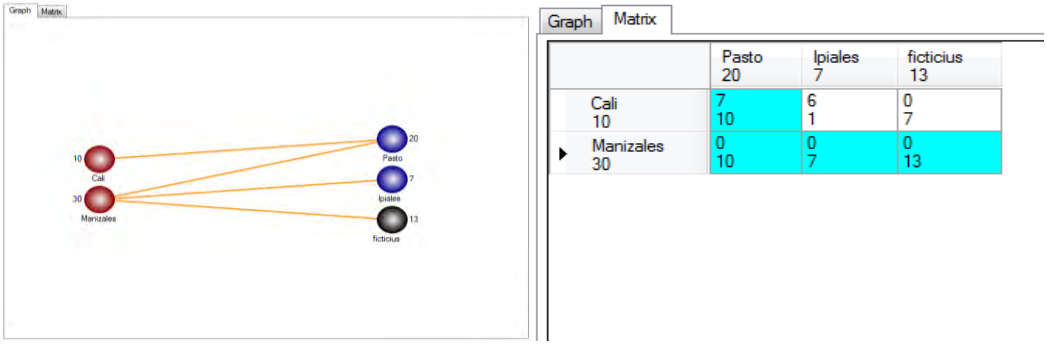
Los anteriores diálogos, están diseñados para la inserción y edición de ofertas y demandas (dialogo izquierda) y para la edición de costos (dialogo derecha).

Figura 69. GUI CtrlSisnova vista de nodo ficticio.



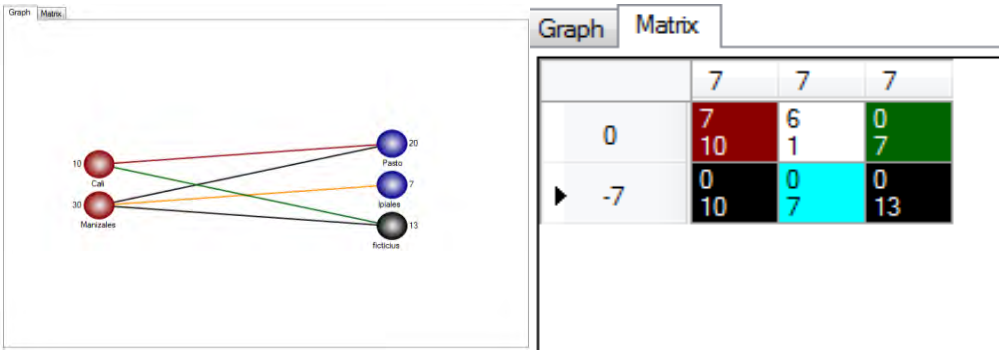
En la figura anterior, se muestra el diseño del control para representar un nodo (origen, destino) ficticio.

Figura 70. GUI CtrlSisnova vistas de obtención de soluciones grafo y matriz.



Para las vistas, grafo y matriz, una solución factible será representada como se manifiesta en la figura 70. El polígono y las variables que salen y entran se muestran como se indica en la figura 71.

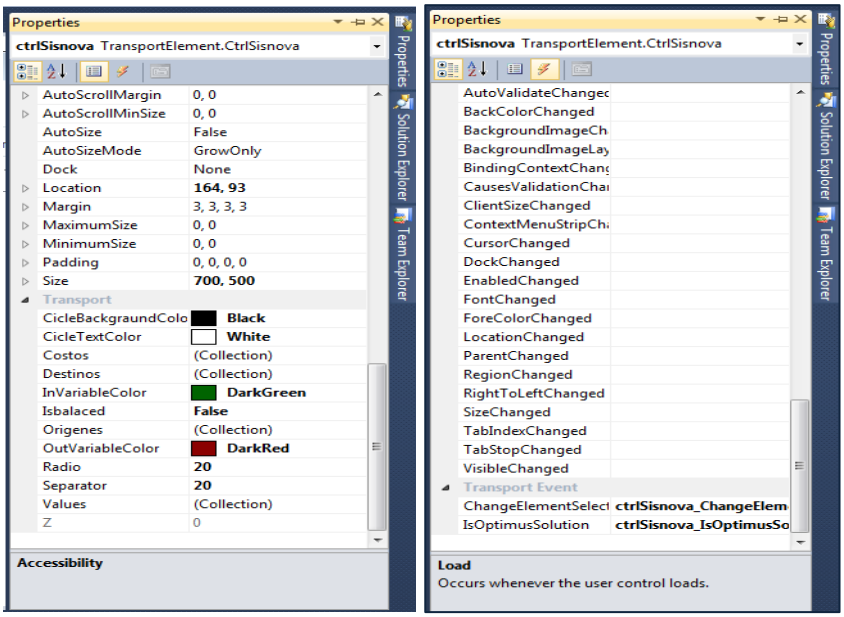
Figura 71. GUI CtrlSisnova vistas de obtención de ciclo y variables de entrada y salida grafo y matriz.



Interfaces a las propiedades y eventos en tiempo de diseño

CtrlSinova como se ha mencionado en secciones anteriores puede ser configurado por los actores desarrolladores o diseñadores. Las siguientes son extensibilidades al panel propiedades que se habilita en tiempo de diseño en Visual Studio.

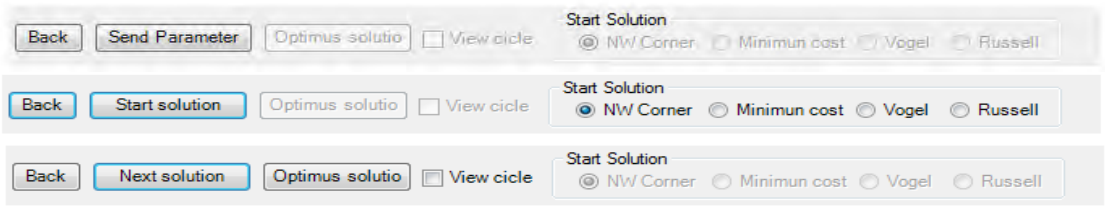
Figura 72. Panel propiedades para la edición de eventos y propiedades de CtrlSisnova.



En la figura 72, se observa la interface de configuración de CtrlSisnova para el actor diseñador, en la figura de la izquierda, el panel propiedades esta seleccionado en la opción propiedades y en él se visualizan las propiedades extendidas* del control ubicadas en la categoria “Transport”, en la figura de la derecha la selección esta en la opción eventos, en la categoria “Transport Event “ estan los eventos extendidos del control.

Interface gráfica para NavigateTransport

Figura 73. GUI NavigateTransport estados del control.

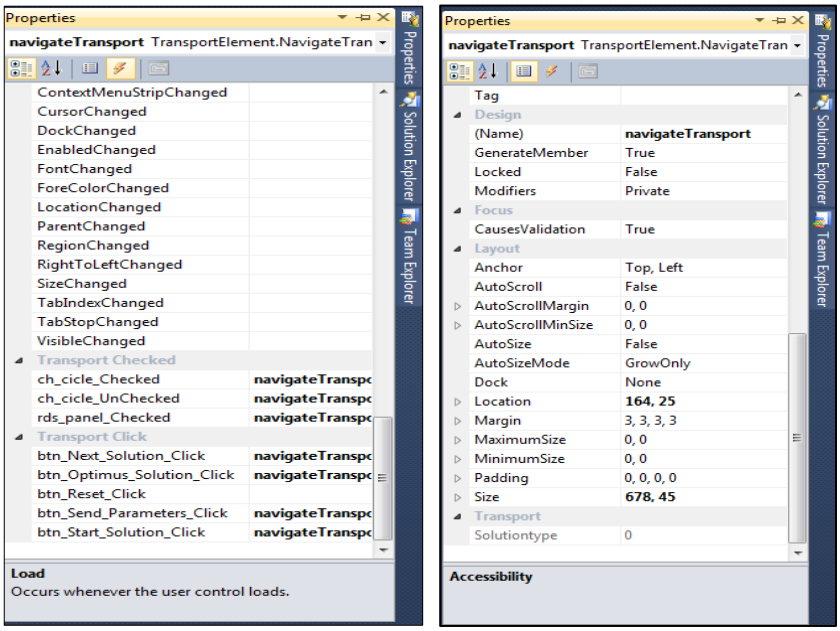


* Extender se refiere a dotar de nuevos elementos al artefacto software o mejorar los existentes.

La interface gráfica de NavigateTransport está diseñada para referenciar los posibles estados para empezar a resolver un problema de Transportes. El primer estado es enviar parámetros (Send Parameters) refiriéndose al envió del planteamiento del problema con las ofertas, demandas, y costos de transporte, una vez se presiona el botón “Send Parameter” (ver figura 73 arriba) el estado del control cambia al segundo estado que es Solución inicial (Start solution), en este estado el control habilita el grupo de opciones de métodos de solución inicial, si se vuelve a presionar sobre el botón esta vez con el nombre “Start solution” (ver figura 73 centro) el estado del control cambia al último estado siguiente solución (Next solution) en el que se deshabilita la selección de solución inicial, y se habilita el botón optima solución (Optimus solution) y la caja de chequeo ver ciclo (View cicle). El botón regresar (back) retorna al control al estado inmediatamente anterior.

Interfaces a las propiedades y eventos en tiempo de diseño

Figura 74. Panel propiedades para la edición de eventos y propiedades de NavigateTransport.

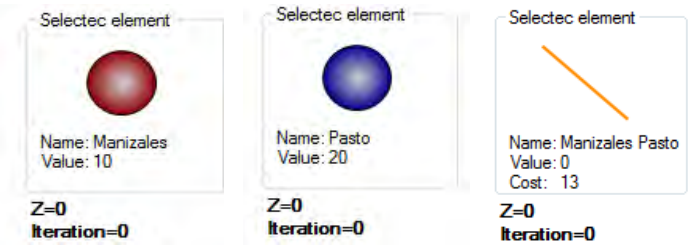


Como se muestra en la figura anterior, NavigateTransport tiene dos categorias extendidas de eventos “Transport Checked” para referenciar eventos en los controles de selección de metodo de obtencion de solución inicial y caja de chequeo, y la categoria Transport Click para los eventos sobre los botones del control (ver figura 74 izquierda). El control solo extiende para las propiedades de control a la propiedad Solutype que es de solo lectura y se encuentra en la categoria “Transport” (ver figura 74 derecha).

Interface gráfica para ViewElements

El control ViewElements está diseñado para mostrar la información de los controles (nodos y arcos) para la vista grafo del control CtrlSisnova y los resultados de la solución del problema de trasporte (valor de transporte e iteraciones). Como se muestra en la siguiente figura.

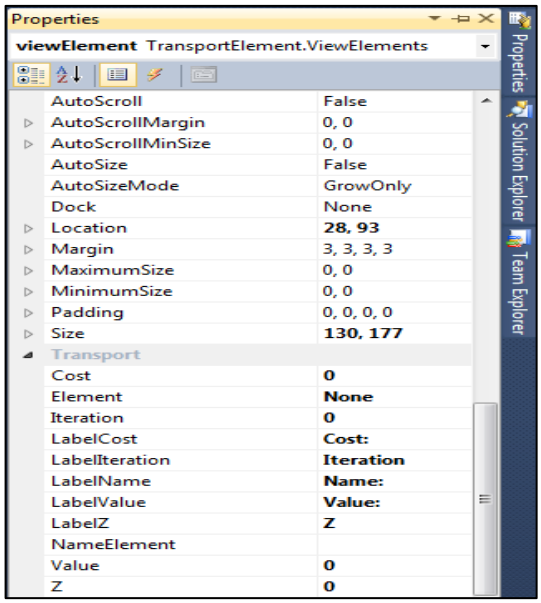
Figura 75. GUI Viewelements vistas de representación de elementos de grafo transporte.



En la figura anterior, se muestra como “ViewElements” muestra la información de los nodos y arcos.

Interfaces a las propiedades y eventos en tiempo de diseño

Figura 76. Panel propiedades para ViewElement.



La figura 76, se aprecia la extensibilidad en las propiedades del control “ViewElement” en la categoría “Transport”. Este control no extiende eventos como los anteriores controles.

3. PROTOTIPOS DE APLICACIÓN PARA EL DESARROLLO DE SISNOVA

En esta sección, se exponen los prototipos de aplicación utilizados para las pruebas de SISNOVA. A continuación se muestra las especificaciones que se tuvieron en cuenta para estos prototipos.

Tabla 18. Especificaciones de plataforma de desarrollo y ejecución para los prototipos.

Prototipo de Aplicación	Hardware	Sistema Operativo	IDE	Lenguaje	Máquina Virtual	Servidor Web
Escritorio	Computador de escritorio	Windows 7	Visual Studio	C#	CLR	
			Visual Studio	Visual Basic	CLR	
			NetBeans	Java	JVM	
Web	Servidor	Windows	Dreamweaver	PHP		Apache
		Linux				
Móvil	Celular	Android	Eclipse	Java	Dalvik	
	Tablet					

Tabla 19. Especificaciones adicionales de ejecución del servicio en los prototipos.

Prototipo de Aplicación	Acceso al servicio	Tipo de llamadas a funcionalidades
Escritorio	SOAP WSDL	Llamada síncronas
Web	SOAP	Llamada síncronas
Móvil	REST	Llamada asíncronas

3.1 AGREGACIÓN DE LOS COMPONENTES A LA PLATAFORMA DE DESARROLLO

Las pruebas de los prototipos de aplicaciones se muestran en tres entornos:

- Aplicación de Escritorio.
- Aplicación Web.
- Aplicación Móvil.

Las aplicaciones de escritorio se desarrollan en .Net, esto para ver la integración de los dos componentes de SISNOVA (servicio Web, biblioteca de controles de usuario), y se implementa en los lenguajes: C#, Visual Basic. Como se muestra en la tabla 18. Y una aplicación de escritorio en Java para probar la funcionalidad del servicio en un ambiente diferente al de .Net.

La aplicación Web se desarrolla en PHP hospedada bajo un servidor Apache para reafirmar el funcionamiento multilenguaje del servicio.

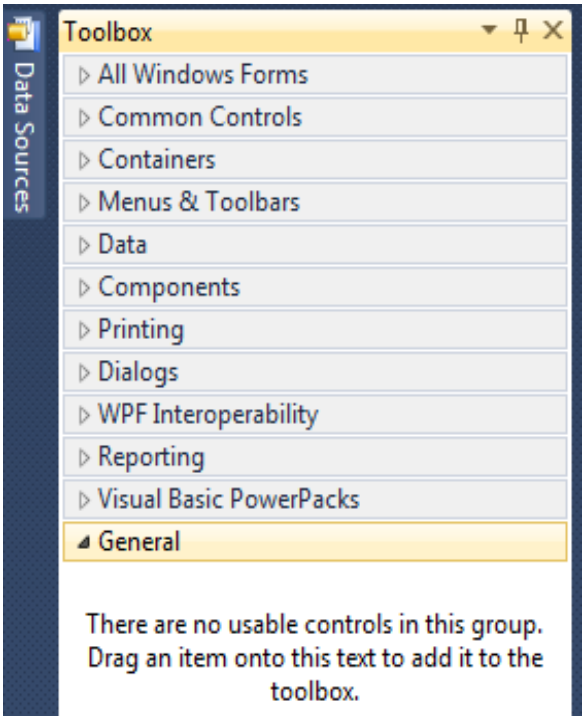
La aplicación móvil se desarrolla en lenguaje JAVA pero para el desarrollo de una aplicación Android para probar el funcionamiento bajo peticiones de recursos REST asíncronas.

3.1.1 Agregación de los componentes de SISNOVA a las aplicaciones de escritorio desarrolladas en .Net

Pasos para agregar la Biblioteca de controles de usuario a la caja de herramientas en una solución Windows Forms de .Net.

1. Se indica la caja de herramientas de Visual Studio para Windows Forms de .Net antes de ser añadidos los controles.

Figura 77. Caja de herramientas de Visual Studio para Windows Forms de .Net



2. En seguida, se muestra como se añade el componente: TransportsLibraryControls

Figura 78. Adicionar el componente Transports_LibraryControl a la caja de herramientas.

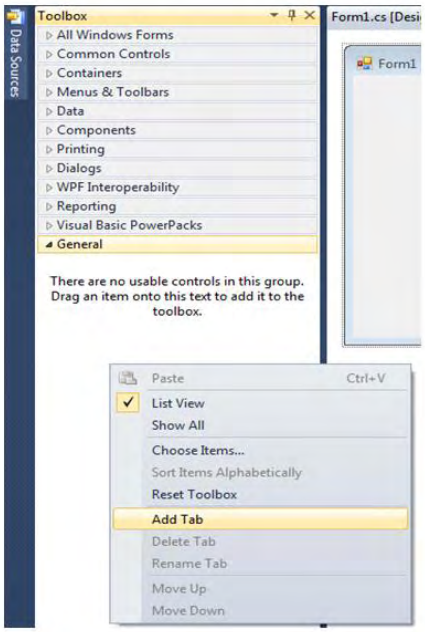


Figura 79. Definir un nombre para el componente Transports_LibraryControl.

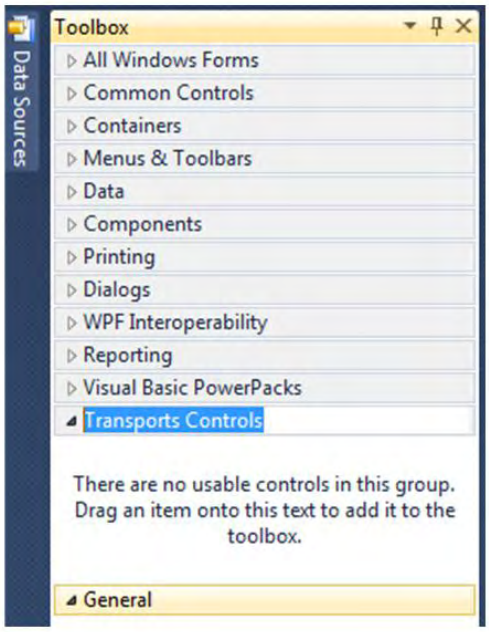


Figura 80. Escoger bibliotecas de enlace dinámico.

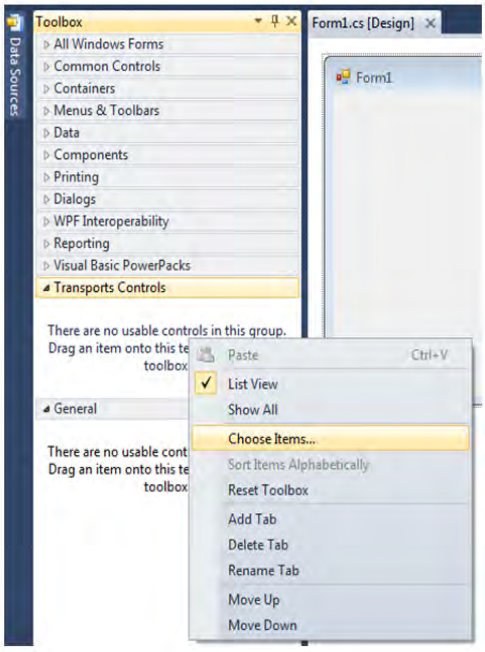


Figura 81. Lista de bibliotecas de enlace dinámico.

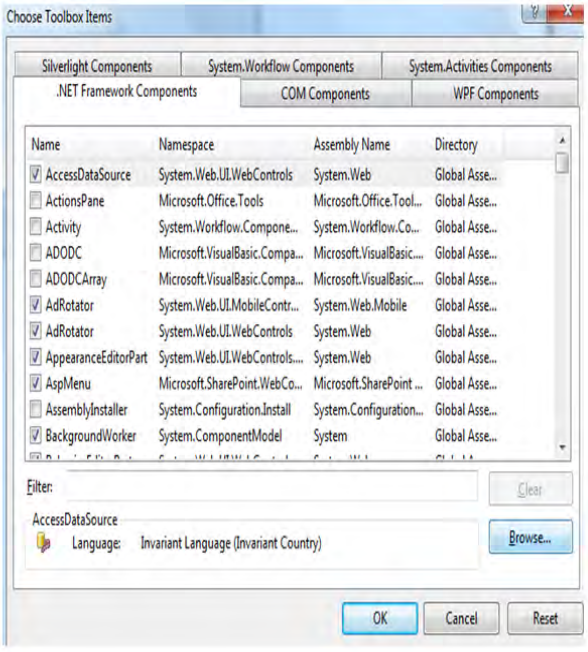


Figura 82. Añadir el componente Transports_LibraryControl.

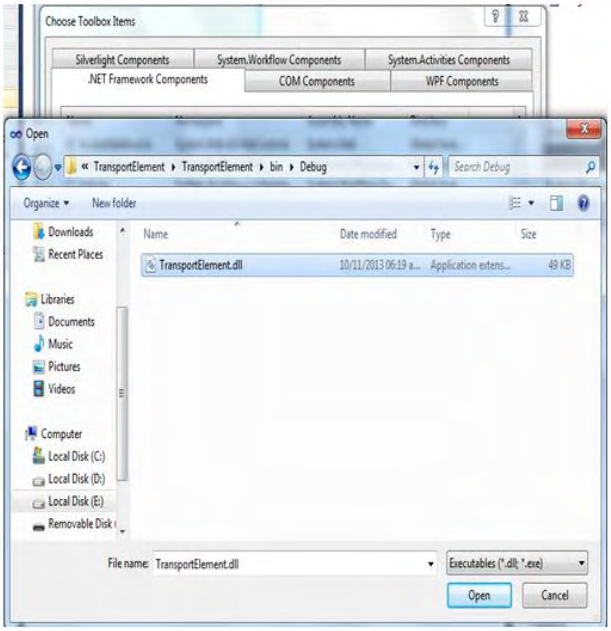
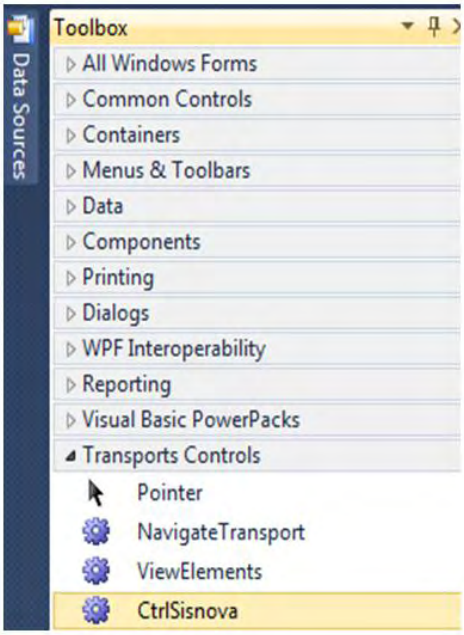
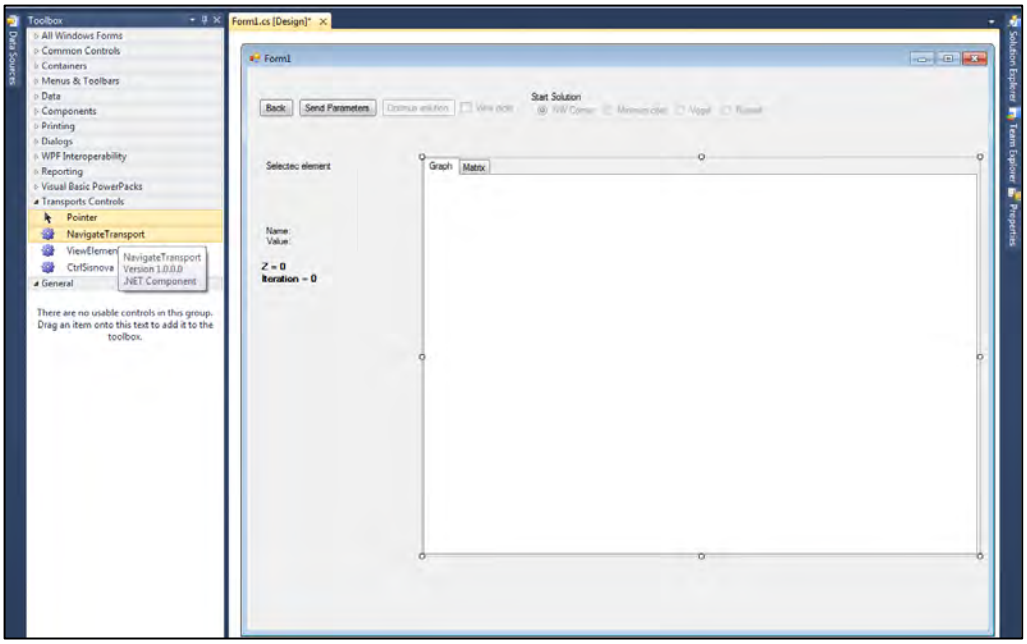


Figura 83. Controles del componente Transports_LibraryControl añadidos en la caja de herramientas de Visual Studio.



3. Con los controles disponibles en caja de herramientas de Visual Studio se procede a arrastran los controles hacia el formulario.

Figura 84. Adicionar los Controles del componente Transports_LibraryControl al formulario



Pasos para añadir el servicio

En el explorador de soluciones de Visual Studio se adiciona la referencia del servicio.

Figura 85. Adición el servicio Transport_Service al explorador de soluciones.

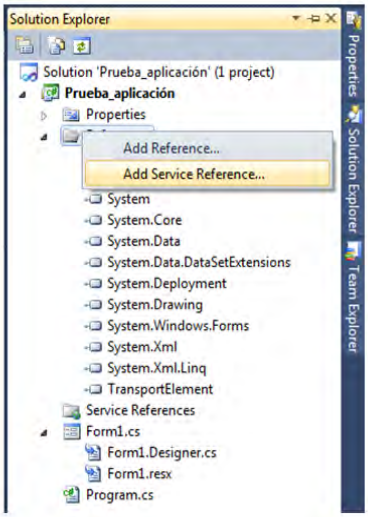


Figura 86. Añadir la dirección del servicio Transport_Service.

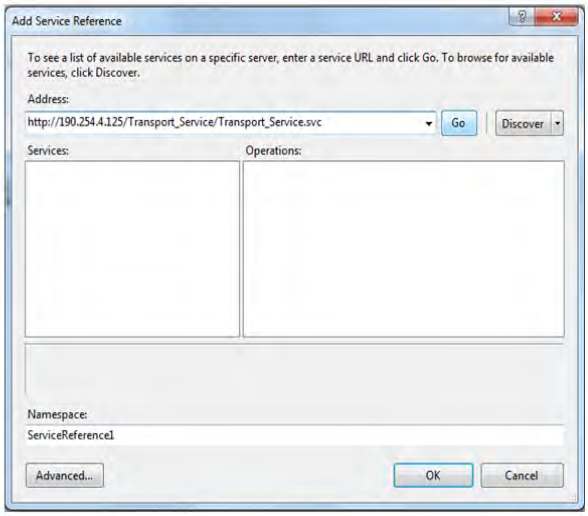


Figura 87. Descubrimiento del servicio Transport_Service.

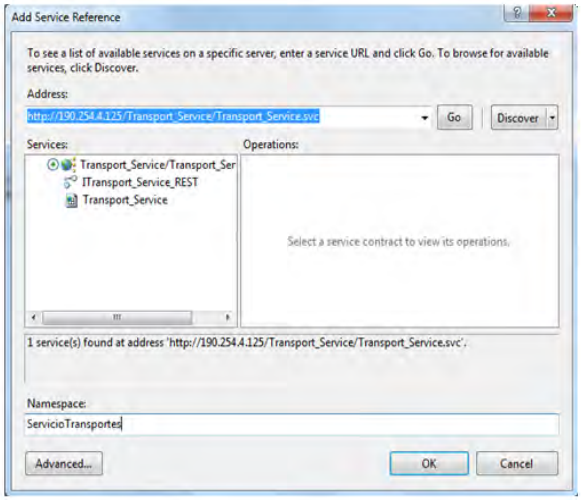
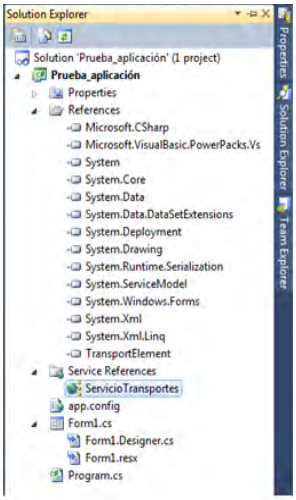


Figura 88. Servicio Transport_Service agregado en el explorador de solución.



La implementación de las funcionalidades del servicio en los lenguajes Visual Basic y C# se encuentra incluida en el cd que acompaña a este documento.

3.1.2 Agregación del servicio para la aplicación de escritorio en NetBeans (JAVA)

Figura 89. Agregación del servicio para la aplicación de escritorio en NetBeans (JAVA).

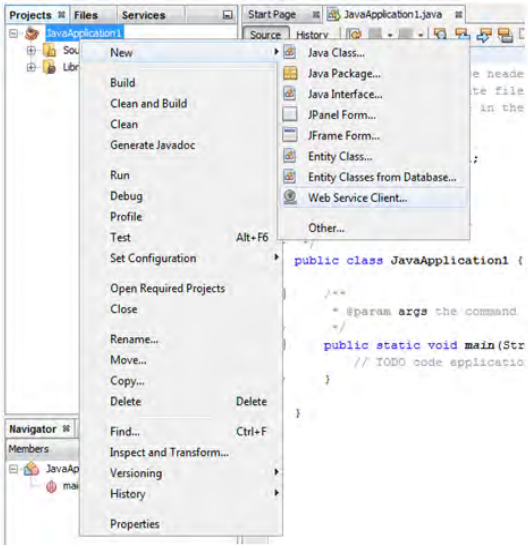


Figura 90. Selección de WSDL y agregación de la url del servicio en NetBeans (JAVA).

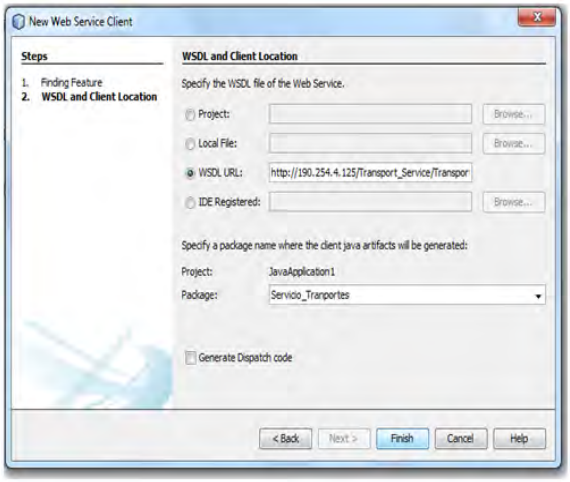
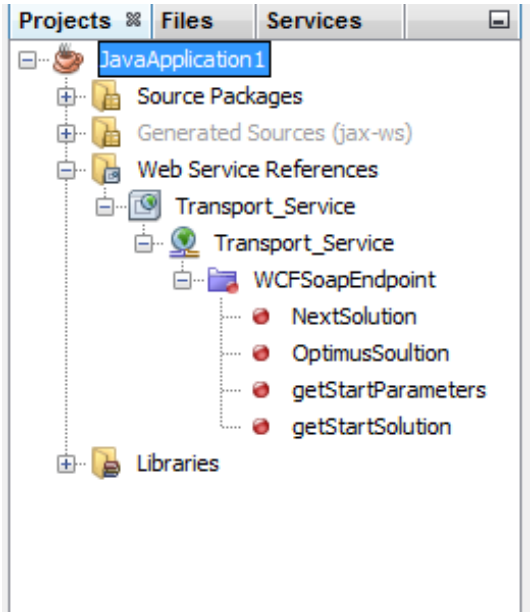
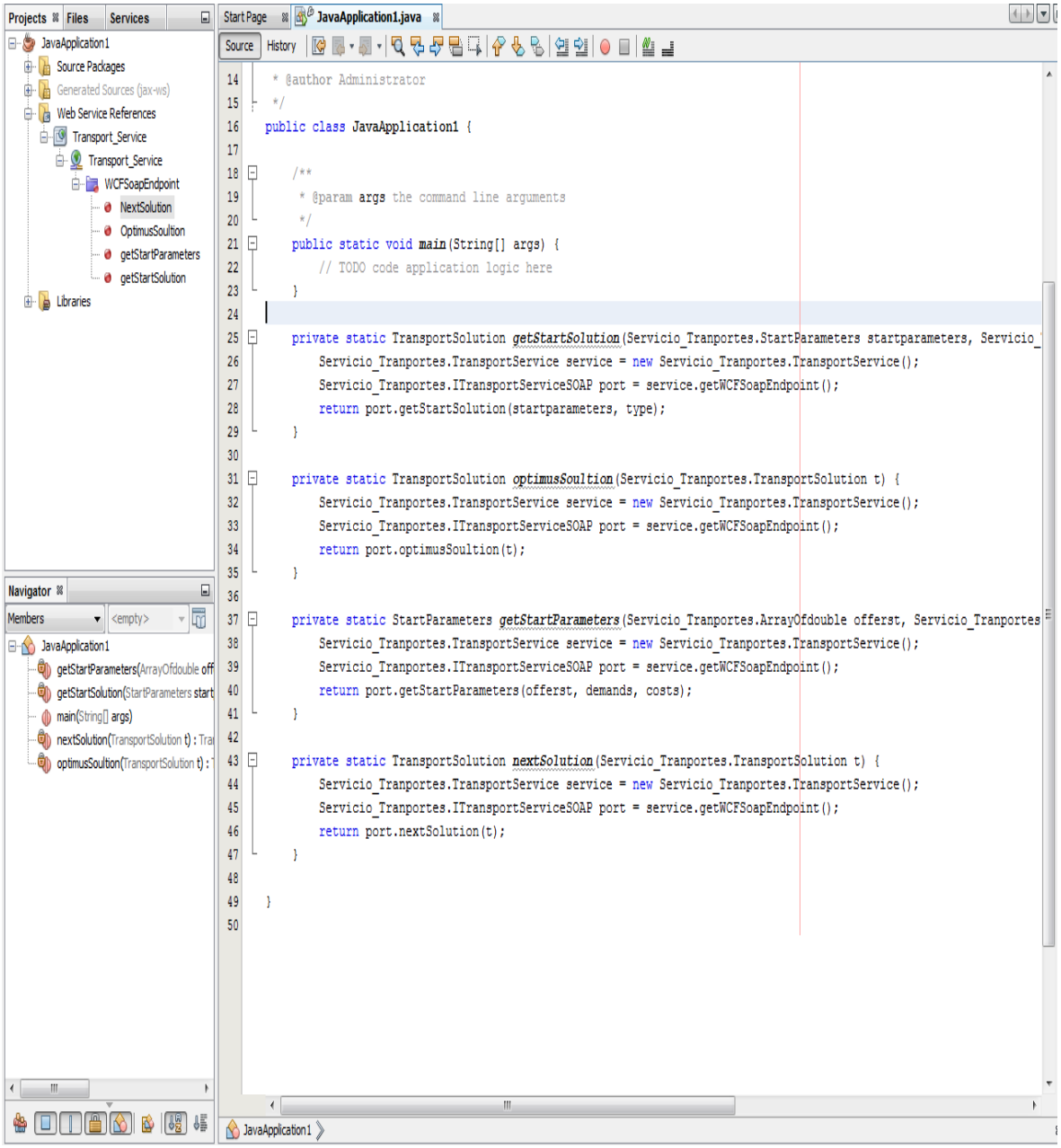


Figura 91. Servicio Transport_Service agregado en el panel de proyectos de NetBeans (JAVA)



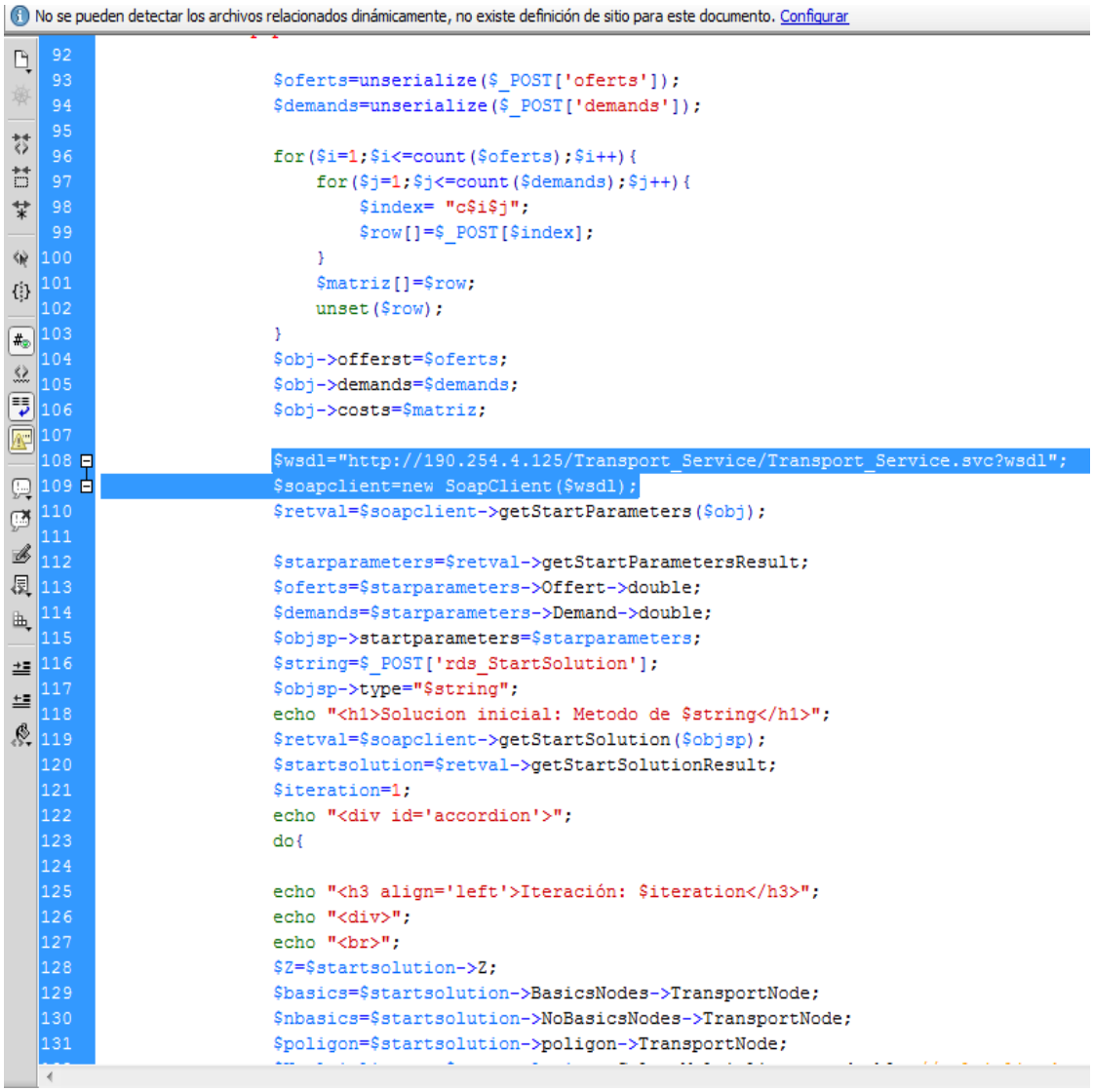
En la siguiente figura, se incorpora las funcionalidades del servicio en NetBeans (JAVA).

Figura 92. Incorporación de funcionalidades del servicio al código fuente en NetBeans (JAVA)



3.1.3 Agregación del servicio para la aplicación Web en PHP

Figura 93. Agregación del servicio para la aplicación Web en PHP.



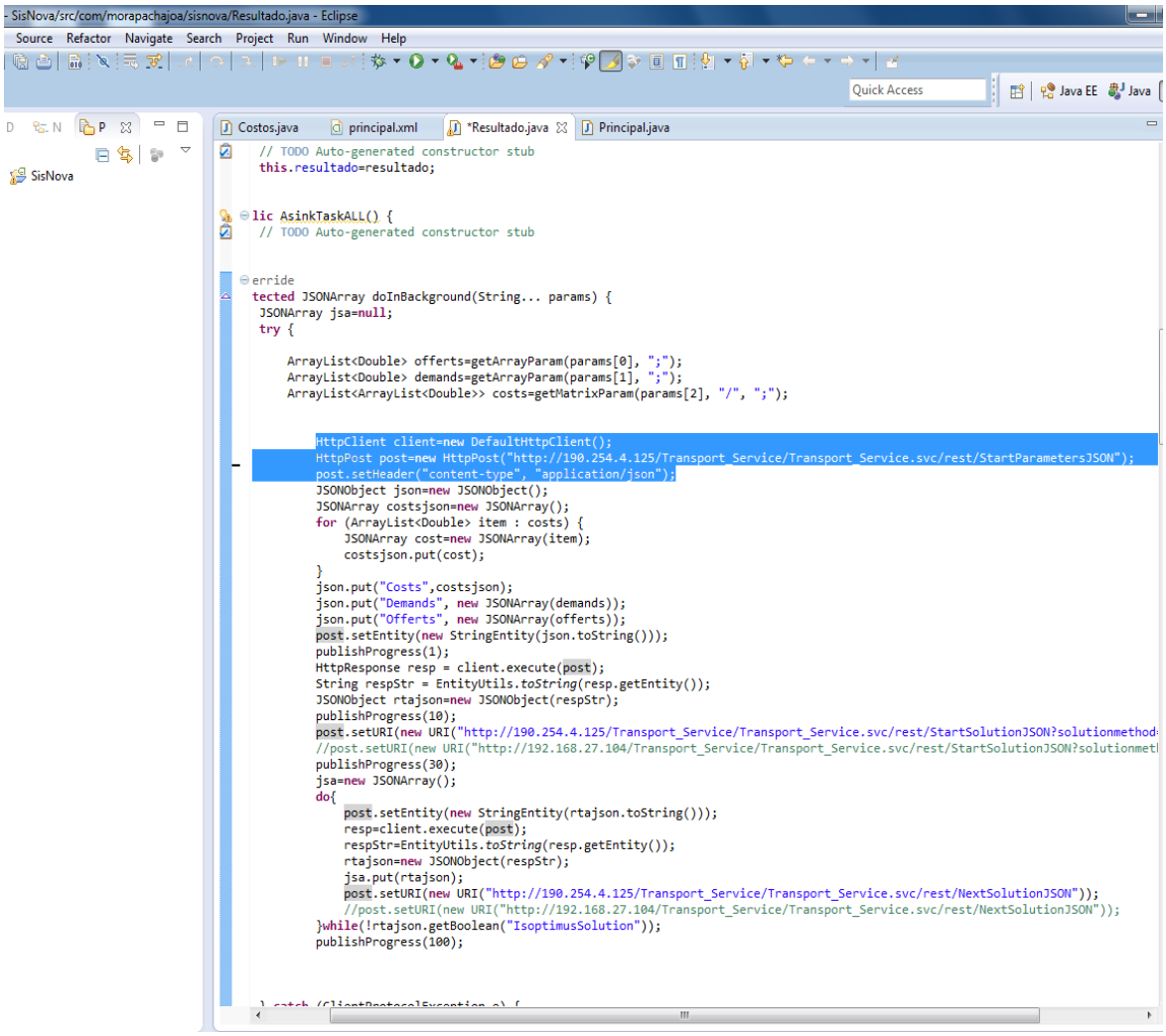
Las líneas resaltadas en la anterior figura, referencian al servicio denominado Transport_Service para consumirlo por SOAP, las líneas referidas se muestran con mayor claridad a continuación.

$wsdl="http://190.254.4.125/Transport_Service/Transport_Service.svc?wsdl";

$soapclient=new SoapClient($wsdl);

3.1.4 Agregación del servicio para la aplicación móvil en Android (JAVA)

Figura 94. Agregación del servicio para la aplicación móvil en Android (JAVA).



Las líneas resaltadas en la anterior figura, referencian al servicio Transport_Service para consumirlo por REST, las líneas referidas se muestran con mayor claridad a continuación.

*HttpClient client=**new** DefaultHttpClient();*

*HttpPost post=**new** HttpPost("http://190.254.4.125/Transport_Service/Transport_Service.svc/rest/StartParametersJSON");*

post.setHeader("content-type", "application/json");

3.2 FUNCIONAMIENTO DE LOS COMPONENTES EN LOS DIFERENTES PROTOTIPOS DE APLICACIÓN

Para presentar el funcionamiento de los componentes se parte del siguiente problema base de transportes.

Problema base

Cuatro ciudades se abastecen de electricidad de cuatro centrales eléctricas con capacidades de 12, 46, 83 y 54 mega watts (MW). Las demandas máximas en las cuatro ciudades se estiman en 68, 22, 47 y 90 MW. El precio por MW en las cuatro ciudades se muestra en la tabla 19.

Tabla de Costos

Tabla 20. Costo en MW para las cuatro ciudades del problema base.

		Ciudad			
		1	2	3	4
Planta	1	40	73	25	69
	2	82	67	51	34
	3	22	98	4	10
	4	35	46	73	52

3.2.1 Funcionamiento de las aplicaciones de Escritorio. En esta sección, se presentan los prototipos de aplicación de prueba en los diferentes entornos de ejecución.

En .Net se realiza la integración de los componentes de SISNOVA, donde se muestra la interfaz del proceso de solución del problema propuesto inicialmente. El mismo diseño de interfaz ha sido implementado para los prototipos de prueba en Visual Basic y C#. La diferencia se denota en el código fuente de cada prototipo.

En Java se muestra la interfaz del prototipo de prueba para demostrar la funcionalidad del servicio en un entorno diferente al de .Net.

Los pantallazos siguientes muestran la solución inicial lograda por los métodos de Esquina Noroeste y Vogel con sus respectivas soluciones óptimas.

Figura 95. Dialogo para la edición de ofertas

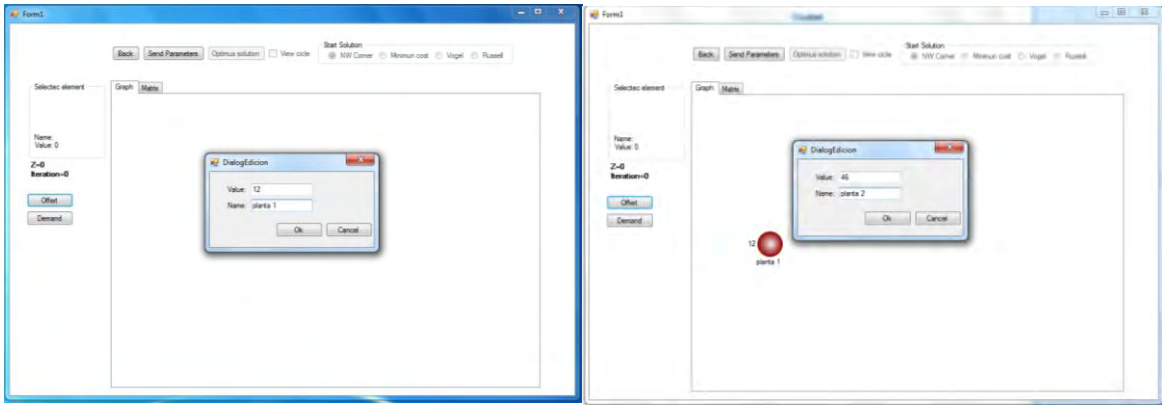


Figura 96. Dialogo para la edición de nodos existentes

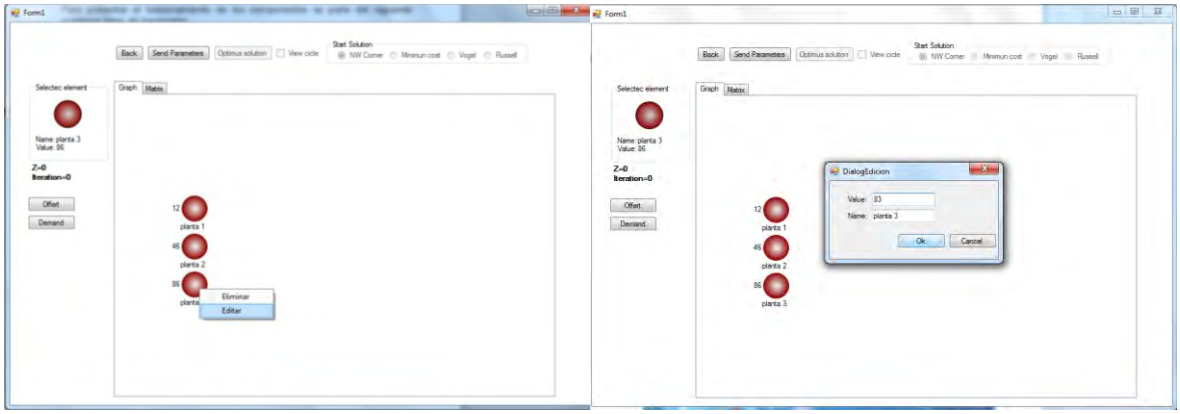


Figura 97. Dialogo para la edición de demandas

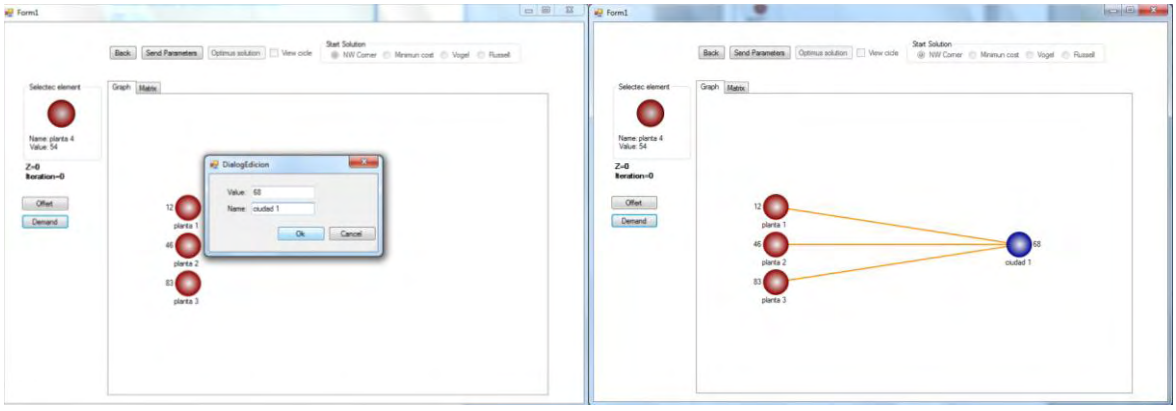


Figura 98. Opción eliminar para los nodos existentes

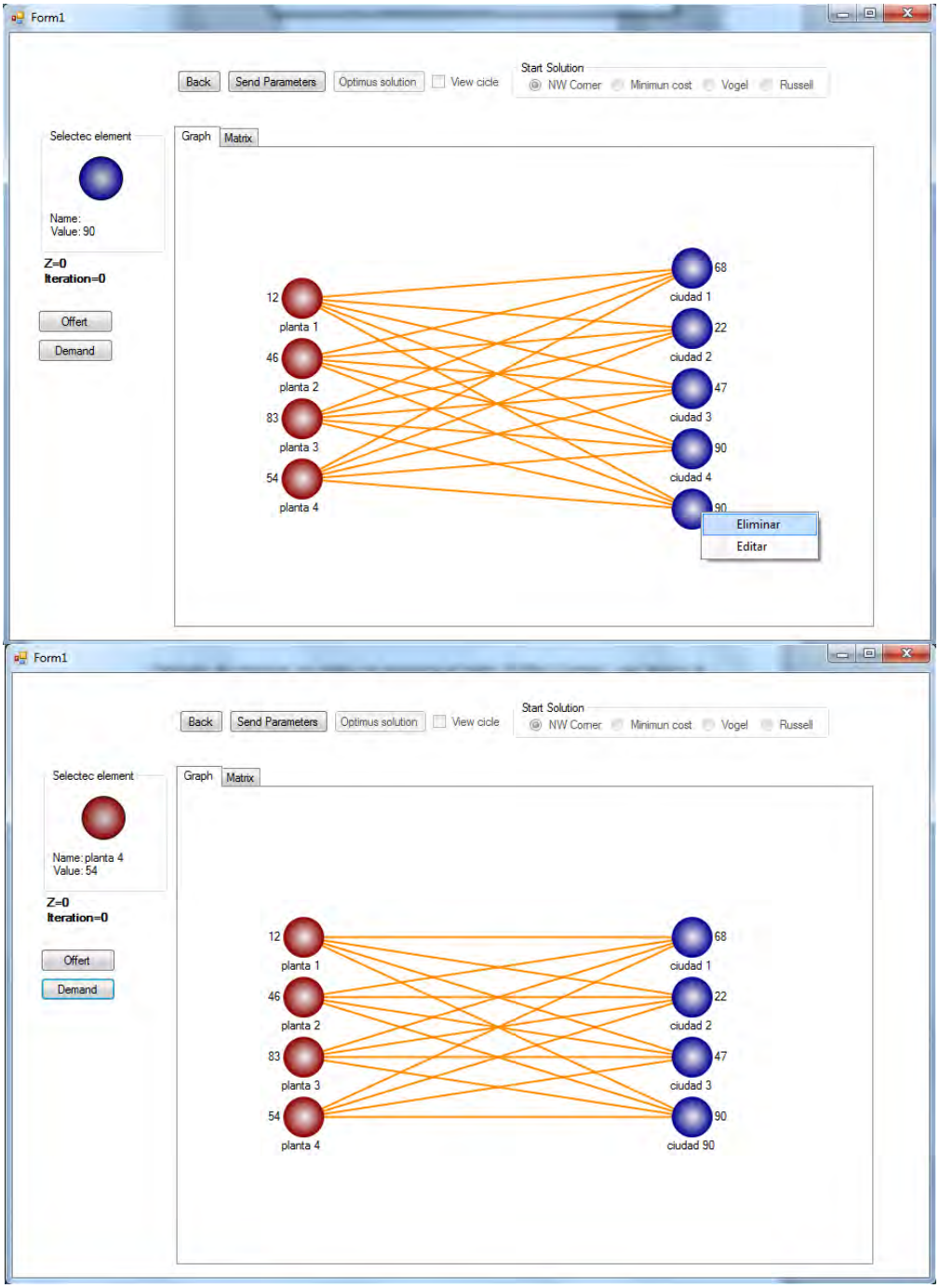


Figura 99. Visualización matricial del problema

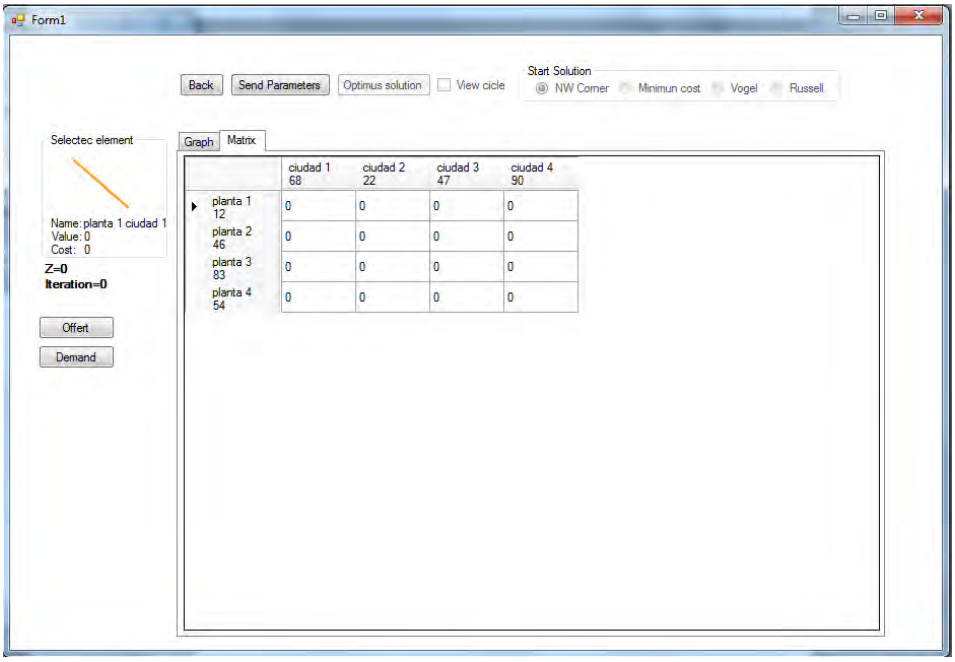


Figura 100. Edición de costos en el grafo

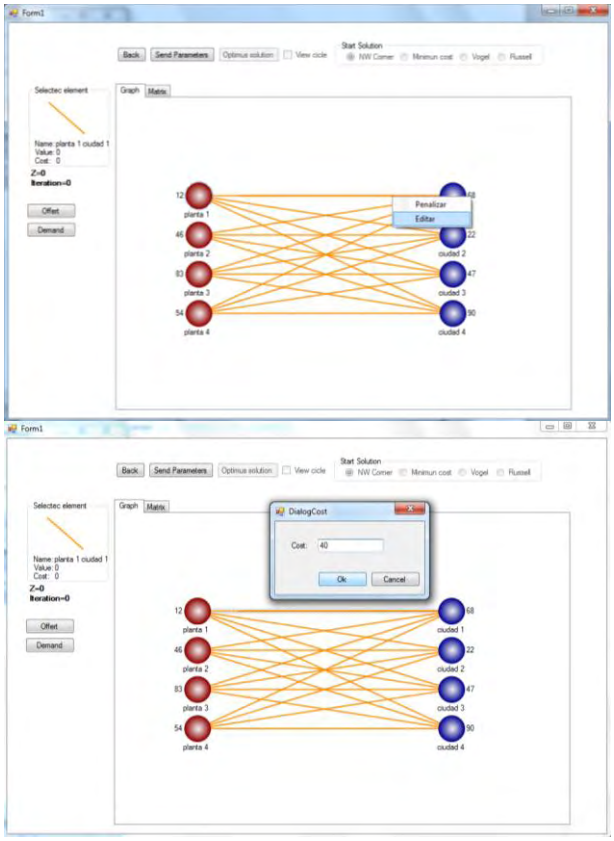


Figura 101. Edición de costos en la matriz y envió del planteamiento del problema por medio del botón Send Parameters.

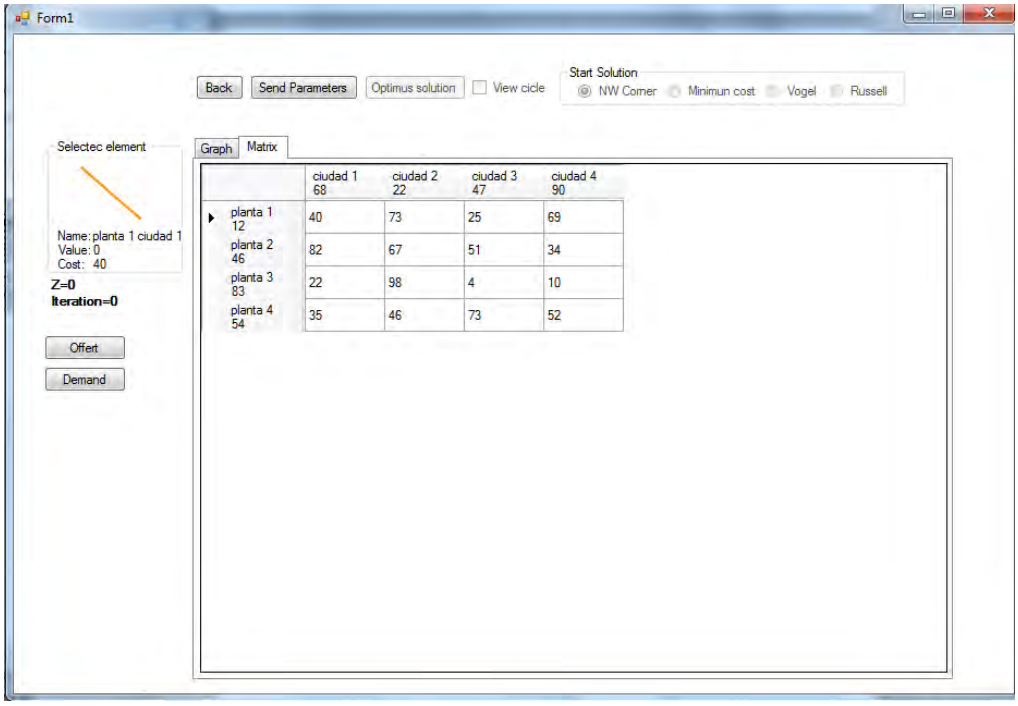


Figura 102. Recepción de parámetros iniciales por servicio y reenvió de los parámetros a través del botón Start Solution - Vista Grafo

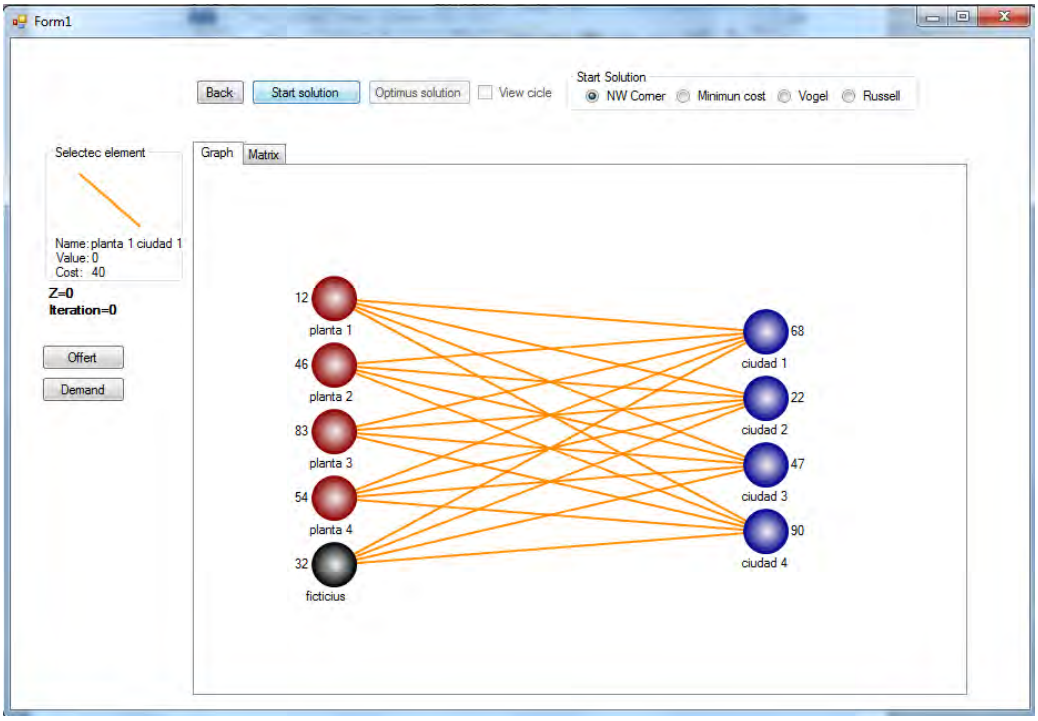


Figura 103. Recepción de parámetros iniciales por servicio y reenvió de los parámetros a través del botón Start Solution - Vista Matriz

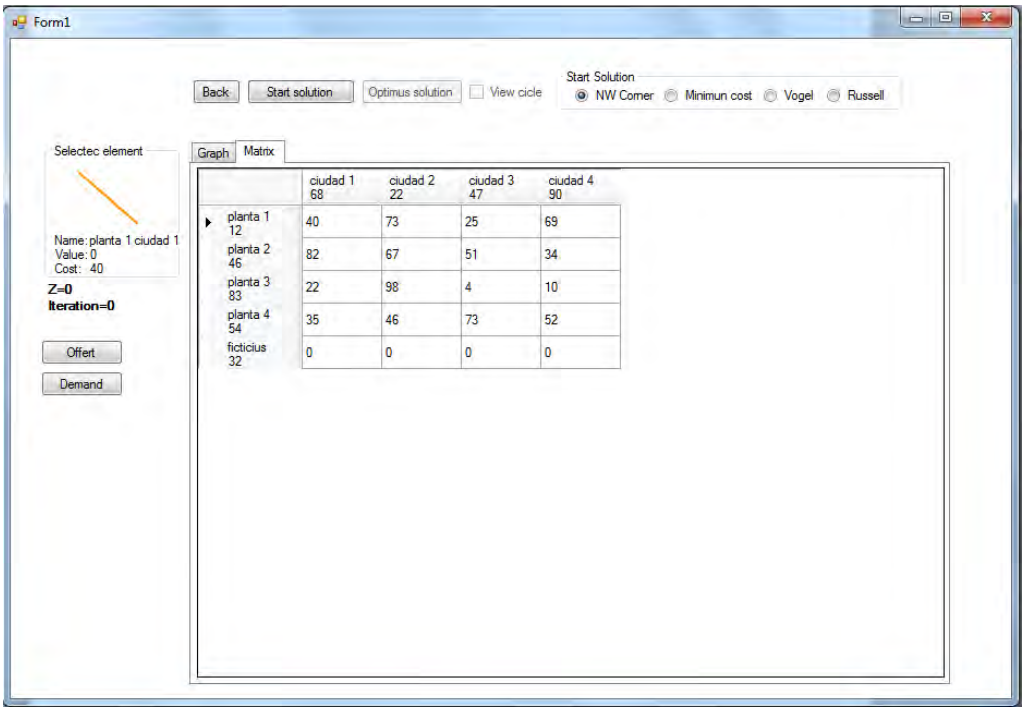


Figura 104. Solución inicial por el método Esquina Noroeste (NW Corner) – Vista Grafo

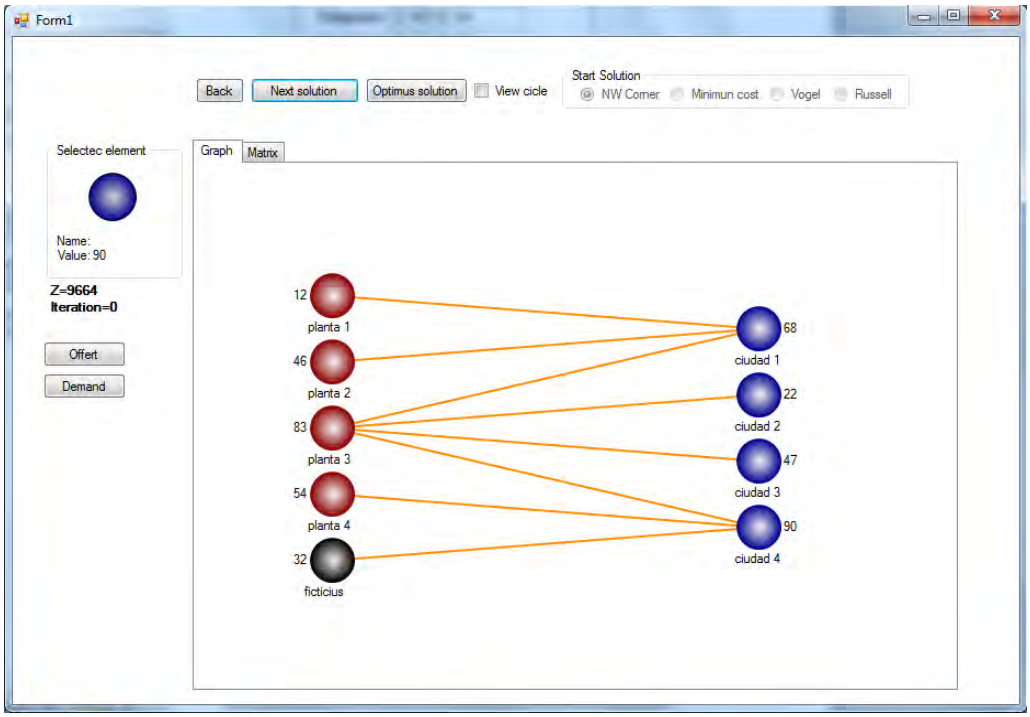


Figura 105. Solución inicial por el método esquina noroeste (NW Corner) – Vista Matriz

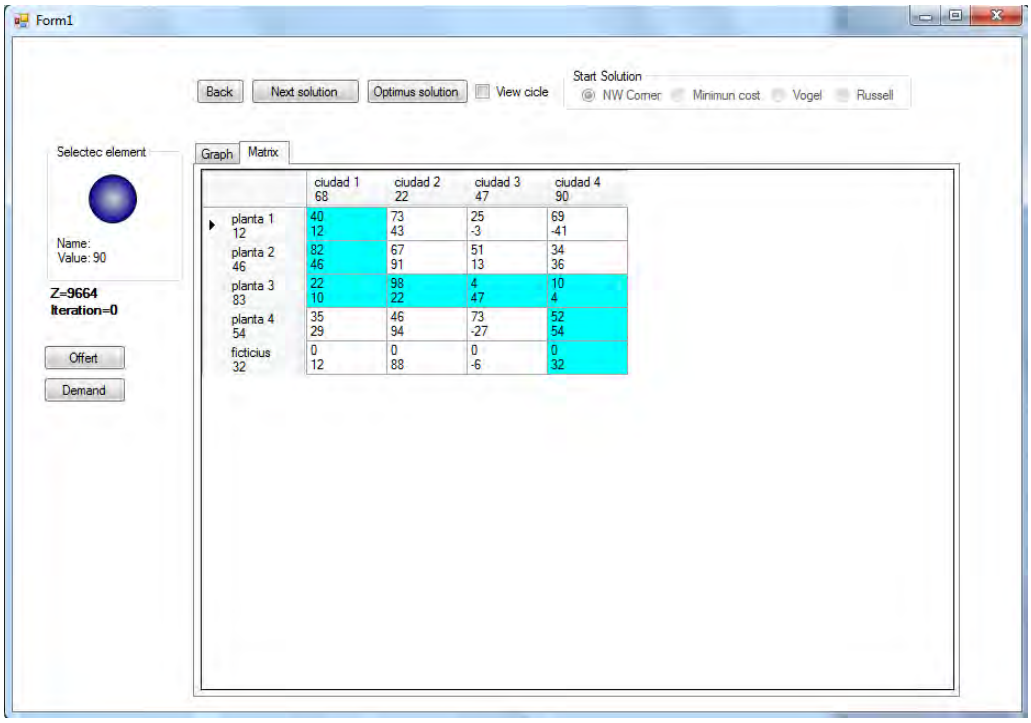


Figura 106. Activación de la vista del ciclo cerrado con las variables de entrada y salida – Vista Grafo

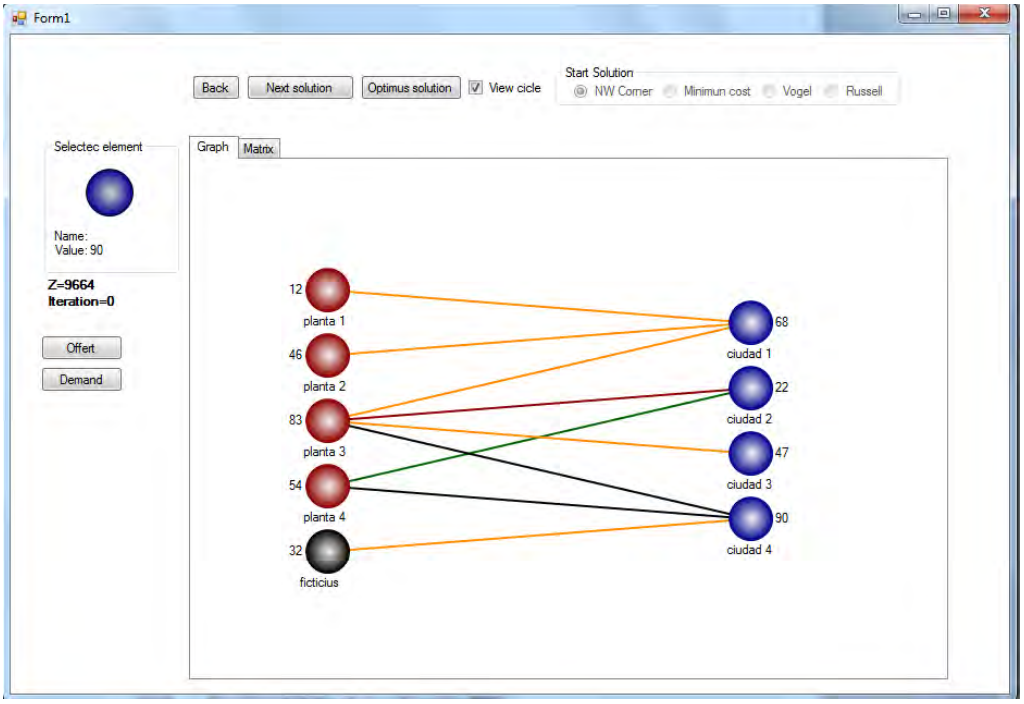


Figura 107. Activación de la vista del ciclo cerrado con las variables de entrada y salida – Vista Matriz

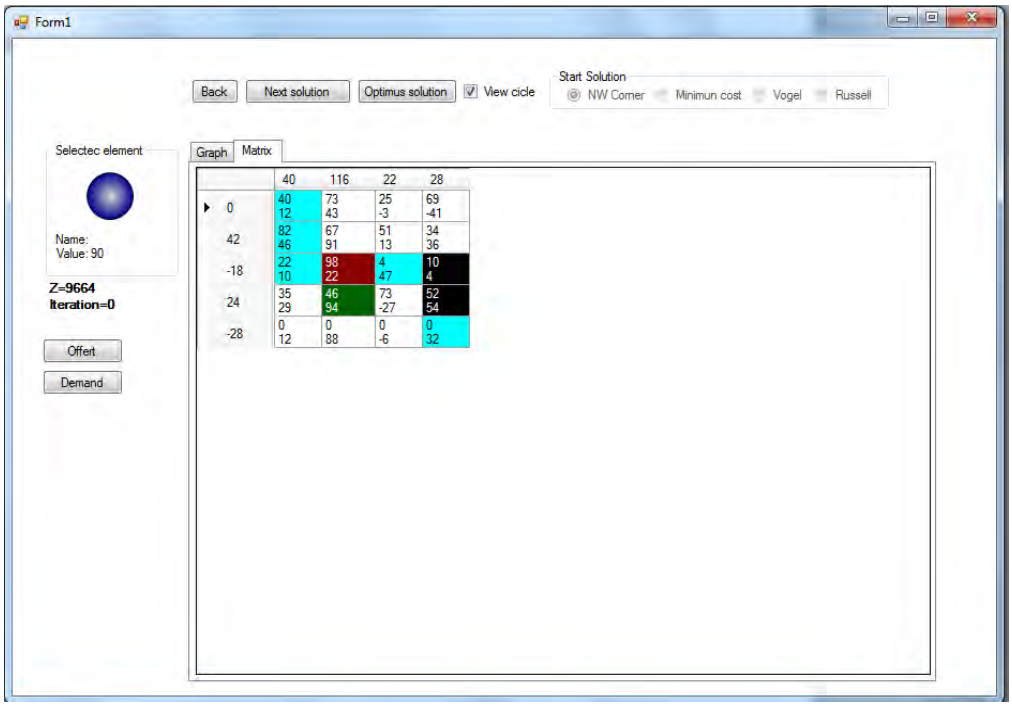


Figura 108. Solución óptima del problema de transportes método esquina noroeste – Vista Grafo

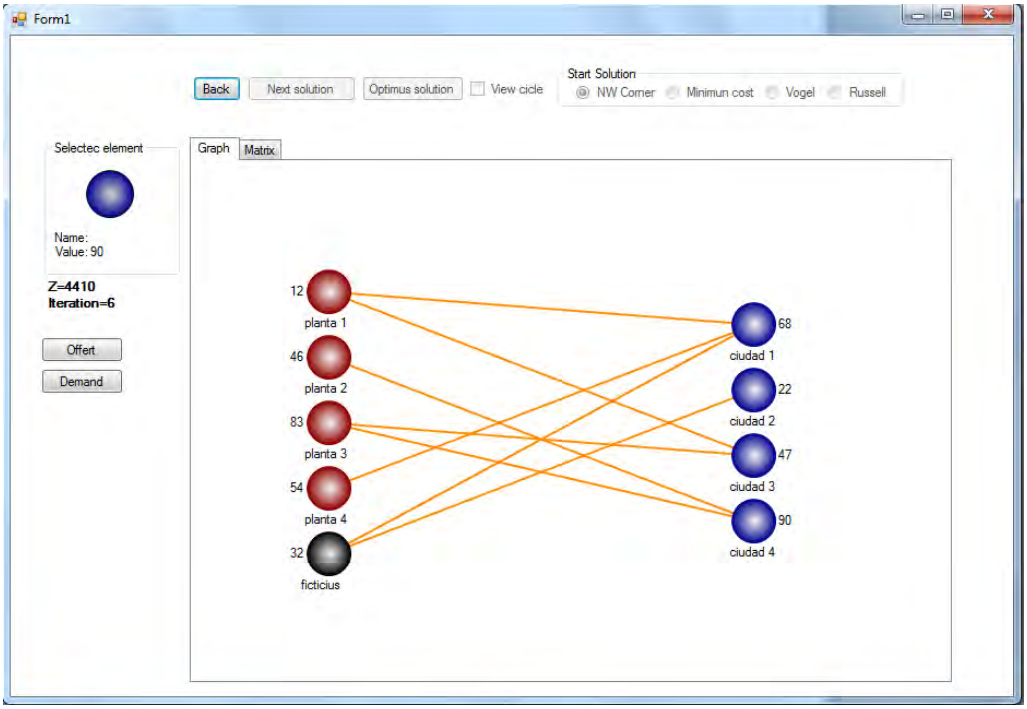
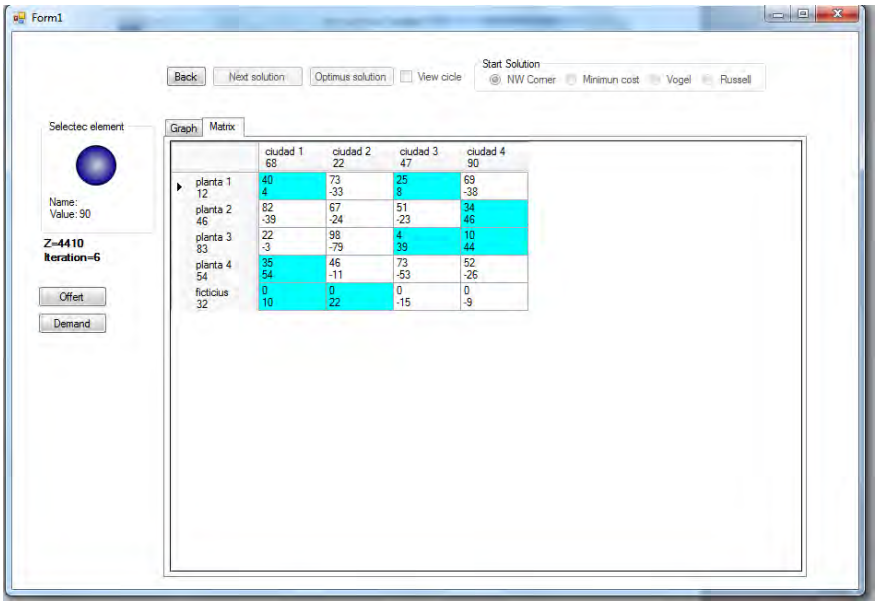


Figura 109. Solución óptima del problema de transportes método esquina noroeste – Vista Matriz



Solución por método de Vogel

En las siguientes figuras, se muestra la solución inicial lograda por el método de Vogel con su respectiva solución óptima.

Figura 110. Solución inicial del problema de transportes método Vogel – Vista Grafo

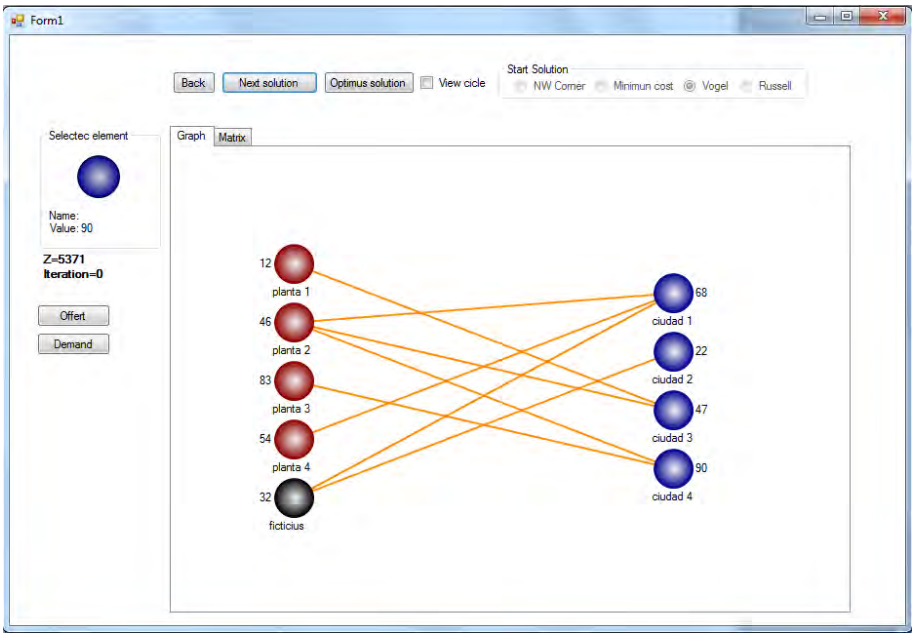


Figura 111. Solución inicial del problema de transportes método Vogel – Vista Matriz

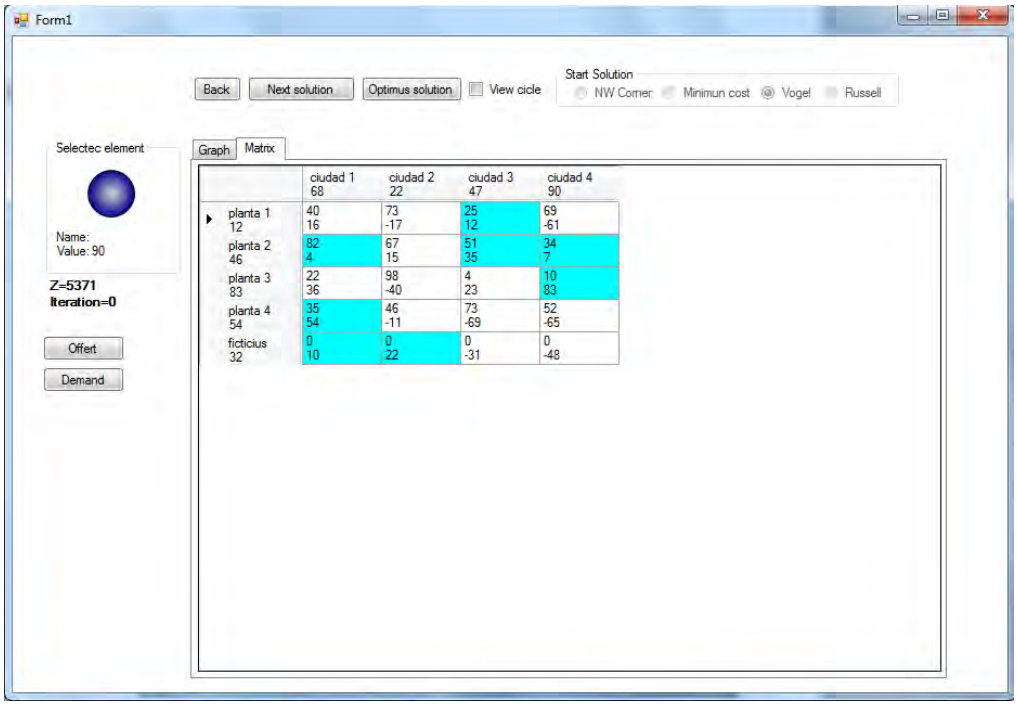


Figura 112. Solución óptima del problema de transportes método Vogel – Vista Grafo

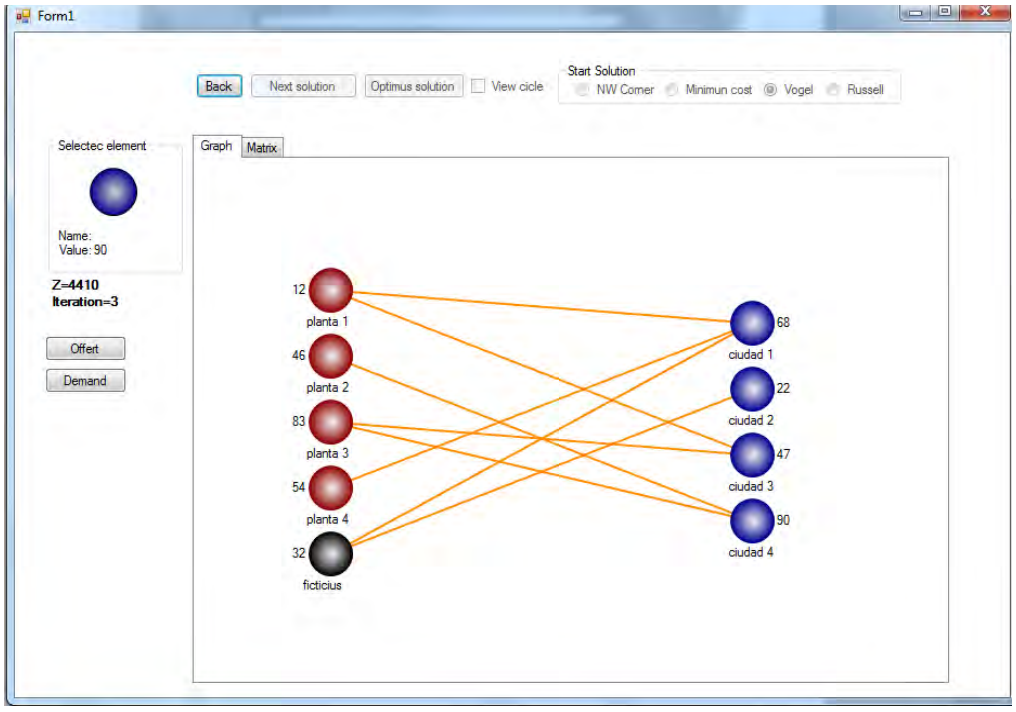
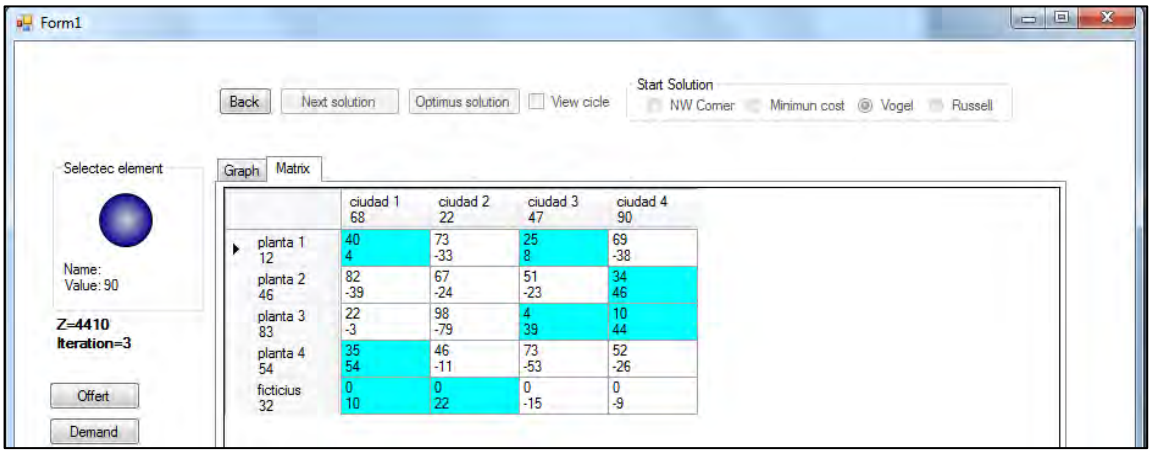


Figura 113. Solución óptima de problema de transportes método Vogel – Vista Matriz



Prototipo de aplicación de escritorio bajo el entorno JAVA

Para un entorno de JAVA se presenta las siguientes figuras, que muestran los pantallazos del prototipo de aplicación con la solución del problema inicial tanto para Esquina Noroeste como para Vogel.

Figura 114. Interfaz inicial de la aplicación de escritorio - Java

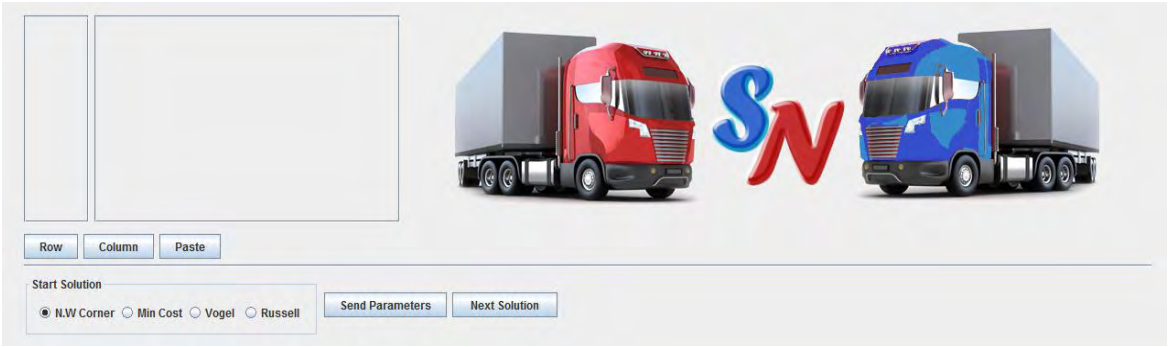


Figura 115. Diálogos para el ingreso de origen (oferta) y destino (demanda) - Java

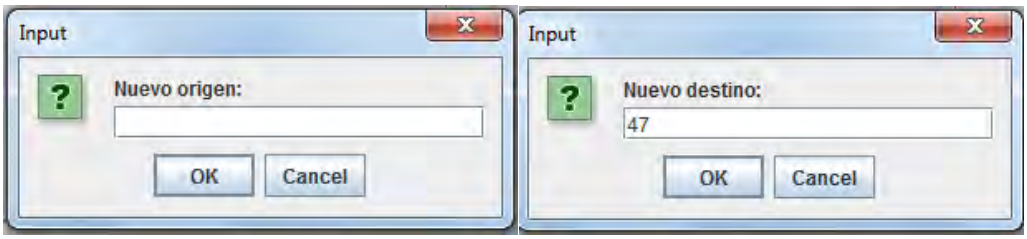
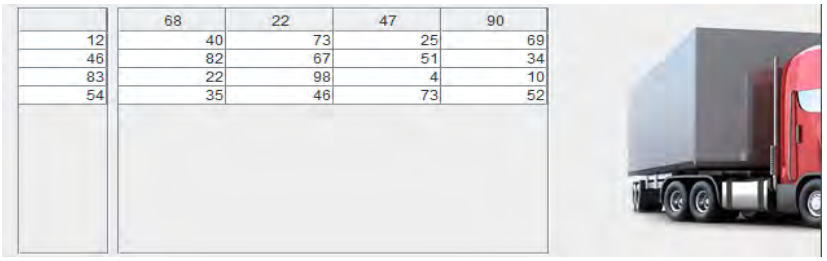


Figura 116. Edición de los costos de transportes - Java



	68	22	47	90
12	40	73	25	69
46	82	67	51	34
83	22	98	4	10
54	35	46	73	52

Figura 117. Problema de transportes balanceado después del envío de los parámetros iniciales – Java

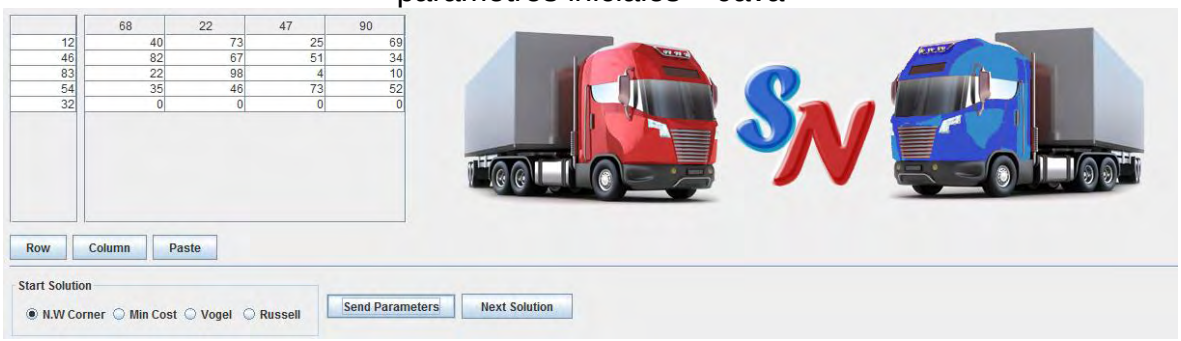


Figura 118. Solución inicial del problema de transportes método esquina noroeste – Java

Figura 119. Solución óptima del problema de transportes método esquina noroeste - Java

Figura 120. Solución inicial del problema de transportes método Vogel - Java

Figura 121. Solución óptima del problema de transportes método Vogel - Java

3.2.2 Funcionamiento de la aplicación Web. En esta sección, se presenta el prototipo de aplicación de prueba construida en PHP. La aplicación Web fue montada en dos sistemas operativos Windows y Linux.

El acceso para aplicación alojada en Windows está disponible en la siguiente url:

http://190.254.4.125:90/TestserviceTransport/

El acceso para aplicación alojada en Linux está disponible en la siguiente url:

http://ingenieria.udenar.edu.co/tsisnova/

A continuación, se muestra las interfaces de la aplicación Web en las cuales se indica la solución inicial y optima del problema de transportes inicialmente planteado por medio de los métodos de costo mínimo y Russell.

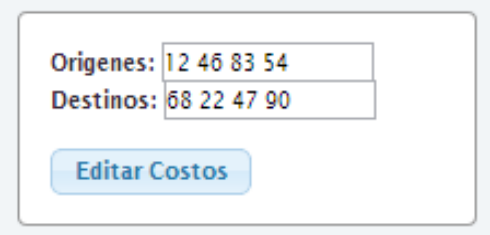
Página de inicio: Parámetros Iniciales.

Esta página consta de dos cajas de texto (Orígenes, Destinos) y un botón (Editar Costos), los cuales se explican después con mayor detalle.

Figura 122. Parámetros iniciales aplicación Web

Aquí se introduce en cada una de las cajas de texto (Orígenes, Destinos) los respectivos valores separados por espacios como se muestra en el siguiente ejemplo. Se observa también, que los números con decimales se separan con un punto.

Figura 123. Ingreso de orígenes (ofertas) y destinos (demandas) – aplicación Web

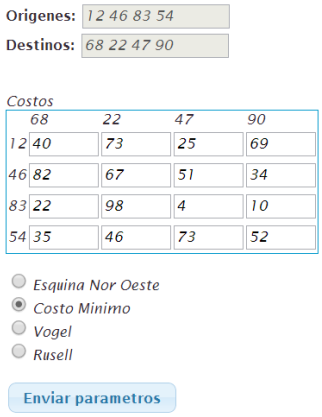


Después de ingresar los datos se presiona el botón (Editar Costos), que lleva a la próxima vista, donde se muestra la tabla de costos, un listado de los métodos de solución inicial y un botón (Enviar parámetros).

Figura 124. Vista de la matriz de costos y de los métodos de solución – aplicación Web

En la tabla Costos, se digita en cada celda el valor de envió correspondiente, según el origen o destino respectivo. Hay que recordar, que los números con decimales se deben separar con un punto. A continuación se presenta el ingreso de los costos de acuerdo al ejemplo anterior.

Figura 125. Edición de costos y selección del método de solución costo mínimo – aplicación Web



Después de llenar la Tabla de Costos se escoge el método que se prefiera y se presiona el botón (Editar parámetros), el cual lleva a la vista que muestra la resolución del problema de transporte planteado. En este caso se escogió el método de Costo Mínimo para la solución inicial y el resultado se ve a continuación.

Figura 126. Vista de la solución inicial método costo mínimo – aplicación Web

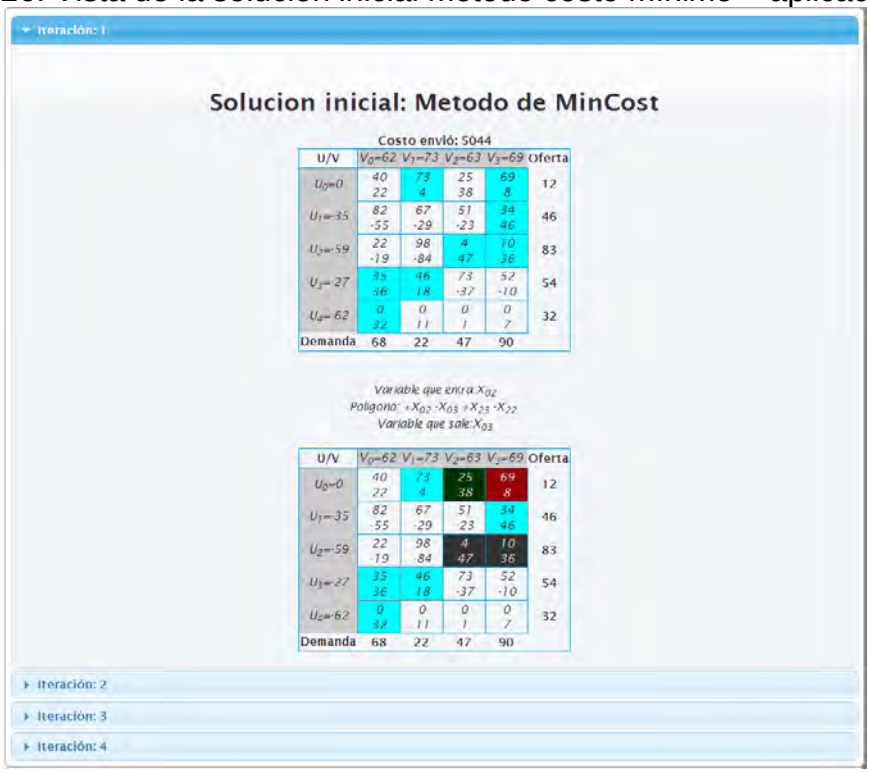
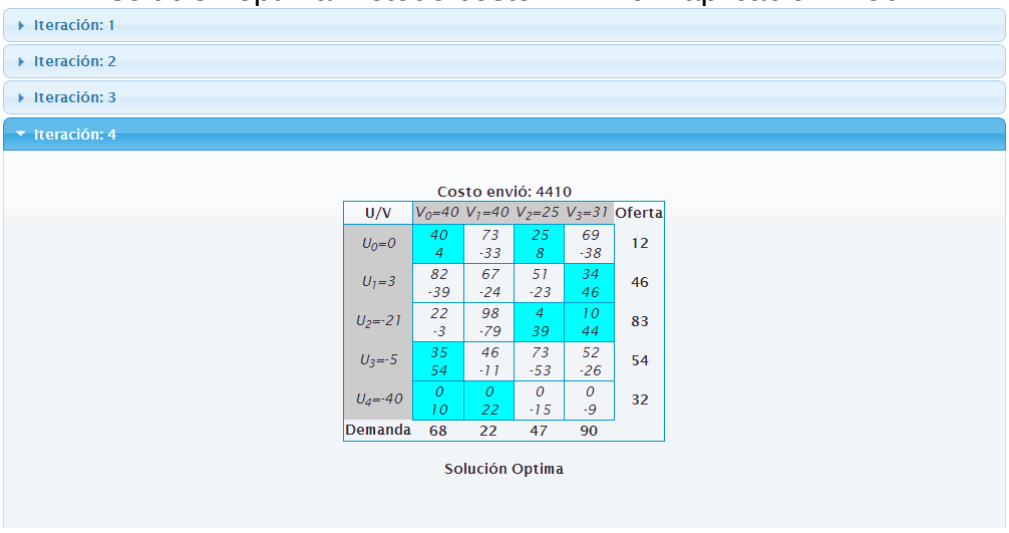


Figura 127. Vista de la solución por iteraciones hasta la iteración donde muestra la solución óptima método costo mínimo – aplicación Web



En esta página se indican las tablas con el proceso de solución del problema de transportes, según el método escogido se presenta el número de iteración que se utilizaron para llegar a la solución óptima. Para el caso de Costo Mínimo se presentaron 4 iteraciones.

Si para este mismo ejemplo, se escoge el método de Russell y se sigue las indicaciones anteriormente nombradas se tiene los siguientes resultados.

Figura 128. Selección del método de solución Russell – aplicación Web

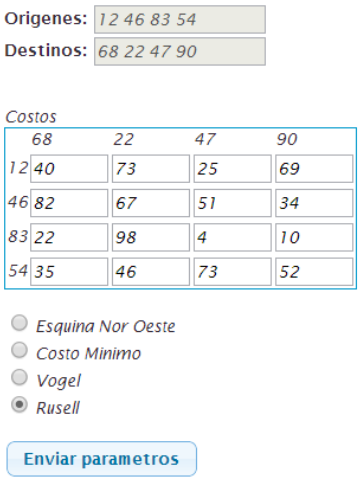
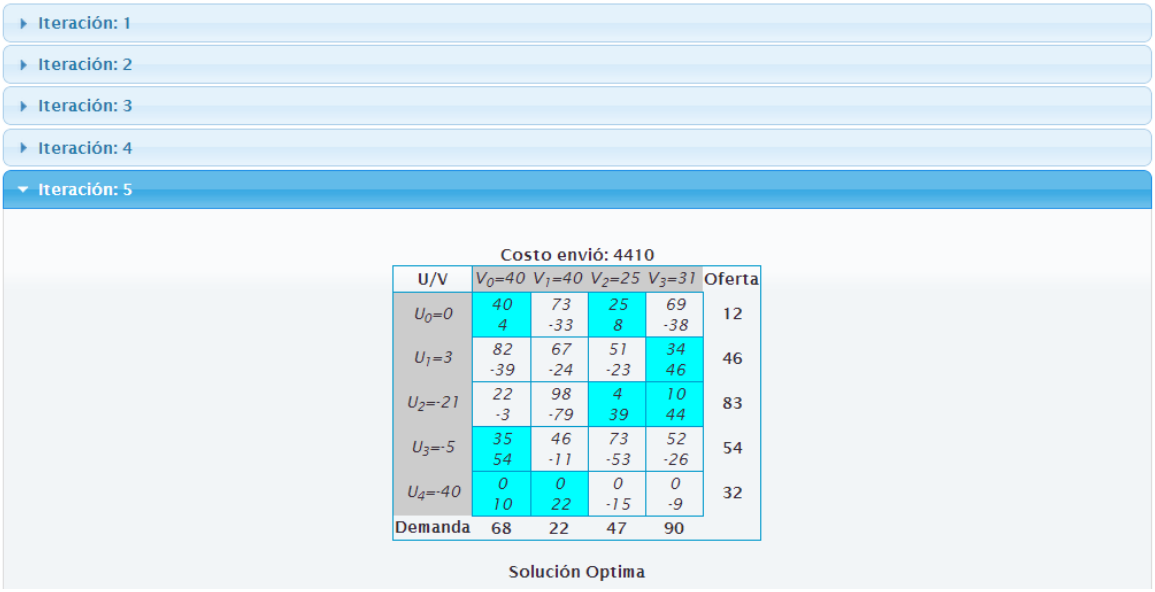


Figura 129. Vista de la solución inicial método Russell – aplicación Web

Figura 130. Vista de la solución por iteraciones hasta la iteración donde muestra la solución óptima método Russell – aplicación Web.



Aquí se observa que el número de iteraciones se redujo a 5 por la escogencia del método de Russell. Así mismo según el método escogido por el usuario el número de iteraciones varía.

3.2.3 Funcionamiento de la aplicación Móvil. En esta sección se presenta el prototipo de aplicación móvil de prueba construida para ejecutarse en dispositivos con sistema operativo Android y codificado mediante el lenguaje Java.

La prueba de este prototipo se realizó en un dispositivo móvil y en una Tablet, que previamente tenían conexión a Internet, así entonces en estos entornos se demostró la funcionalidad del servicio.

A continuación, se muestran pantallazos del prototipo de prueba de aplicación en un dispositivo móvil, escogiendo los métodos de Costo Mínimo y Russell para la obtención de la solución inicial y óptima.

Figura 131. Dispositivo móvil usado en la prueba del prototipo de aplicación móvil.

Figura 132. Interfaz de inicio de la aplicación móvil.

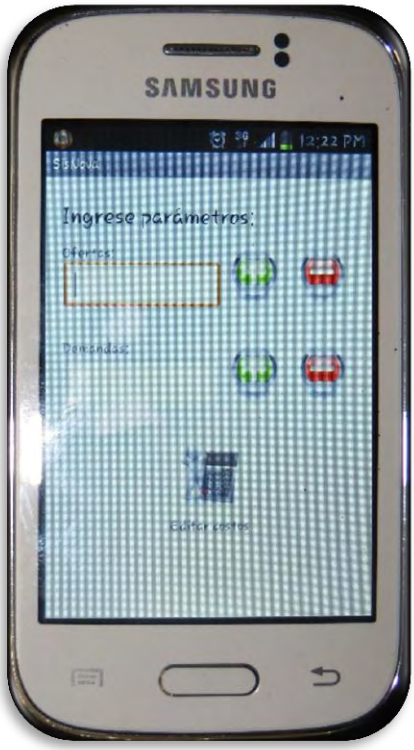


Figura 133. Ingreso de los parámetros ofertas y demandas en la aplicación móvil.

Figura 134. Edición de los costos de transportes en la aplicación móvil.

Figura 135. Selección del método de costo mínimo en la aplicación móvil.

Figura 136. Solución inicial método de costo mínimo - aplicación móvil.

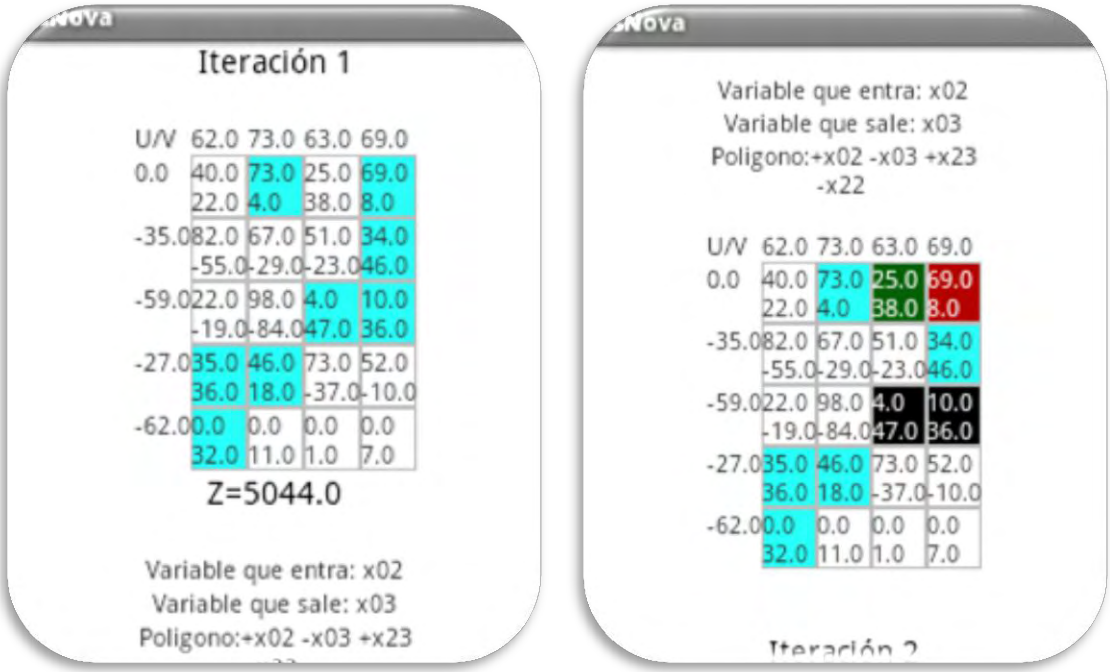
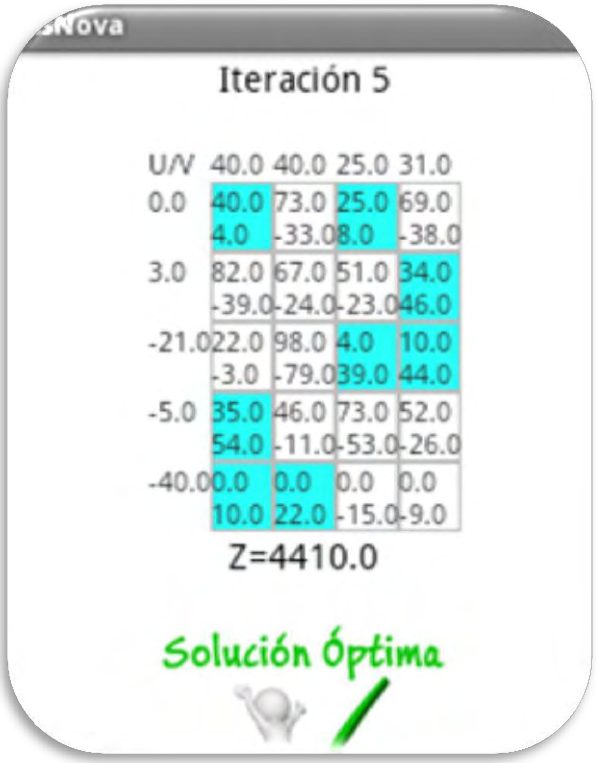


Figura 137. Solución óptima método costo mínimo - aplicación móvil.

Figura 138. Selección del método de Russell en la aplicación móvil.

Figura 139. Solución inicial método Russell - aplicación móvil.

Figura 140. Solución óptima método Russell - aplicación móvil.



3.2.4 Tabla de Resultados. En la tabla siguiente, se muestran los resultados de las pruebas realizadas con los prototipos de aplicación, en donde se comprueba la funcionalidad de todos los métodos de solución inicial y solución óptima. Las coincidencias en los resultados se presentan porque la solución proviene de la misma fuente (Servicio Web).

Tabla 21. Tabla de resultados de los prototipos de aplicación.

Prototipos de Aplicación	Método Escogido	Solución Inicial	Solución Optima	No. De Iteraciones
Escritorio Windows .Net	Esquina Noroeste	9664	4410	6
	Vogel	5371	4410	3
Escritorio Windows Java	Esquina Noroeste	9664	4410	6
	Vogel	5371	4410	3
Web PHP	Costo Mínimo	5014	4410	4
	Russell	5360	4410	5
Móvil Android (Java)	Costo Mínimo	5014	4410	4
	Russell	5360	4410	5

4 CONCLUSIONES

Se ha presentado en este trabajo la construcción de un sistema de elementos de software reutilizables para el apoyo a la construcción de nuevas aplicaciones que requieran funcionalidades para la resolución de problemas de transporte, llamado SISNOVA, basándose en los conceptos de funcionamiento de las bibliotecas de clases o librerías de funciones e implementados en un nivel más alto de reutilización mediante los servicios Web y las bibliotecas de controles de usuario.

Se ha hecho una separación de SISNOVA en dos componentes en base a los dos conceptos de reutilización mencionados en el párrafo anterior. Un componente principal llamado Transport_Service implementado como servicio Web y un componente auxiliar implementado como biblioteca de controles de usuario llamado Transport_LibraryControls. El servicio Web es quien proporciona todas las funcionalidades para la resolución de problemas de transportes y la biblioteca de controles de usuario quien proporciona las interfaces gráficas para la representación y solicitud del planteamiento del problema y soluciones factibles del método de transportes.

Se abstrae la resolución del problema de transportes bajo 4 objetivos: enviar formulación del problema y obtener problema balanceado; enviar problema balanceado y método de solución inicial y obtener solución factible inicial; enviar solución factible inicial y obtener siguiente solución factible; y enviar solución factible y obtener solución óptima. El éxito de sintetizar el problema bajo tan solo 4 objetivos está en la representación estructural de la solución factible y los parámetros iniciales.

Se incluye en la funcionalidad de obtención de la solución inicial de transporte la posibilidad de obtenerse bajo cuatro métodos diferentes de solución, permitiendo realizar una comparación de los resultados de la solución inicial factible entre los diferentes métodos.

Se ha utilizado la metodología RUP para el desarrollo de cada componente, esto ha sido muy acertado ya que debido a que estos componentes son para reutilizarse, cada funcionalidad debe estar debidamente garantizada, es entonces que RUP asegura la calidad del software mediante iteraciones con incrementos en el software acompañados de un flujo adiciona de soporte para la gestión de configuración de cambios, gestión del proyecto y adecuación de entornos de instalación.

Se dota al servicio Web con la capacidad de soportar peticiones por SOAP y REST y en este último bajo los formatos de datos XML y JSON. En el documento se presenta una propuesta general para dotar a un servicio cualquiera de esta

capacidad, donde además se muestra una guía de diseño para la codificación y configuración mediante plantillas de tabla y abstracciones de sintaxis, y al final se emplea esta propuesta tomando a Tranport_Service como especificación.

Se construye una biblioteca de controles de usuario para integrarse a las herramientas del API Windows Forms de .NET. Ofertando tres controles: un control principal (CtrlSisnova) autosuficiente con la capacidad de representar el planteamiento del problema de transportes y sus respectivas soluciones factibles, esto lo hace a través de una vista por grafo y matriz. Los otros dos controles son auxiliares, VIewElement que se usa para extender la visualización de la información del grafo y resultados del problema y NavigateTransport que sirve como puente al control principal para ejecución de sus funcionalidades.

Se construyen prototipos de aplicación de prueba para SISNOVA, como aplicaciones de escritorio, Web y móvil con diferentes lenguajes de programación y sistemas operativos, bajo solicitudes SOAP o REST y consumo síncrono o asíncrono.

El código de los prototipos de aplicación aporta información sobre como consumir el servicio Transport_Service para las diferentes plataformas especificadas en este documento y puede orientar de forma informal sobre el consumo de servicios Web en general, para los entornos propuestos.

La biblioteca de clases únicamente puede usarcé en .NET para el API Windows Forms, esto implica una desventaja dentro y fuera de la plataforma .NET ya que si se usa otro api dentro de .NET como WPF este control ya no puede ser referenciado. A pesar de esta desventaja el uso de la biblioteca de controles dentro del ambiente soportado es una herramienta potente que se puede incorporar a la solución en cuestión de segundos.

El servicio al ser probado en los prototipos propuestos comprueba el soporte especificado en la teoría (multiplataforma, multilenguaje, multiusuario) y además gracias en parte a su implementación, soporte de integración con componentes externos a SISNOVA, características multi-formato (XML, JSON), multi-acceso (SOAP, REST) y soporte de llamadas síncronas y asíncronas. Esto hace a SISNOVA con un amplio soporte y versatilidad.

Al construir los prototipos de aplicación se valida a la reutilización de software en la etapa de implementación como un método eficiente de construcción ya que el tiempo de desarrollo de estos prototipos sin usar SISNOVA se aprecia mucho más extenso y los esfuerzos de acoplamiento son menores a la implementación desde un punto cero. Estos dos hechos incluso representarían un costo de producción menor por ser el costo de producción directamente proporcional al tiempo e inversamente proporcional a los esfuerzos.

Mediante los prototipos de aplicación se observa que los métodos de solución inicial de Vogel y Russell son los más eficientes obteniendo en la mayoría de problemas de prueba el menor número de iteraciones para llegar a la solución óptima de transporte. Pero debido a la informalidad de las pruebas no se puede determinar cuál es el más eficiente de los dos.

Con los prototipos de aplicación se entiende que el consumo del servicio es dependiente de la infraestructura de bibliotecas de clases o librerías que ofrezca la tecnología que se esté tratando. Lo anterior se deduce al notar que las bibliotecas de clase de JAVA.SE para el consumo de SOAP/WSDL no se pueden referenciar en una aplicación Android e incluso si se quiere hacer uso del servicio por SOAP hay que recurrir a una biblioteca externa llamada KSOAP.

Los beneficios multiplataforma de un servicio puede sostenerse únicamente con las tecnologías .NET o JAVA tomadas cada una con independencia total, esto ocurre gracias a sus máquinas virtuales CLR y JVM respectivas disponibles para la mayoría de sistemas operativos. El CLR ofrece esta garantía gracias al proyecto MONO. Los beneficios multiplataforma en PHP son gracias a la implementación estándar de los intérpretes de PHP en servidores como APACHE.

El éxito de la obtención de los resultados en el servicio radica en el hecho de que la infraestructura de los servicios Web utiliza estándares de carácter internacional y este el camino que dio el rumbo correcto a esta investigación. Esto no implica que todo debe hacerse a través de estándares implica que si se encuentra una alternativa mejor hay que proponerla como estándar.

5 RECOMENDACIONES

Usar como estrategia para la construcción de controles de usuario métodos que impliquen una representación de interfaces bajo un lenguaje estándar como HTML, XML, XAML etc.

Aprovechar los métodos de solución inicial de las clases que implementan la lógica de negocios del problema de transportes, para ofrecer funcionalidades que muestren los pasos que sigue cada método de solución. Para lograrlo se invita a encapsular la información en una estructura de clases con herencia, de manera que la información más compleja sea representada a partir de la información más simple y de esta manera se aproveche el polimorfismo para cada clase.

Hacer uso de los métodos del servicio Transport_Service para obtener los parámetros de envió de información.

Realizar una separación de capas similar a la propuesta por SISNOVA para el desarrollo de futuras soluciones, para aprovechar los insumos de la lógica de negocio de cada solución y de esta manera evitar la reconstrucción de funcionalidades iguales.

Garantizar las funcionalidades de los servicios Web mediante un alojamiento en un servidor o servidores confiables y con alta estabilidad.

Analizar el desempeño del servicio para que se proveche de mejor manera los recursos del servidor mediante un estudio de viabilidad de acople de las tecnologías WCF y NODE.

Crear un catálogo software para tener un acceso organizado y estandarizado hacia los artefactos software para cada etapa de reutilización en el proceso de desarrollo.

Reutilizar artefactos software para la construcción de aplicaciones siempre que sea posible y cuando no lo sea crear nuevos productos de manera que se cree una cultura de reutilización de software y una evolución en la generación de conocimiento.

BIBLIOGRAFÍA Y REFERENCIAS

ASCUNTAR Sandra, CASTILLO James, CORREA Ronald. Simplex Media, Universidad de Nariño, 2009.

BRAUDE, Eric J. Ingeniería de Software: Una perspectiva orientada a objetos. Editorial Ra-Ma. 2003

BRAY, Tim. El lenguaje extensible de marcas (XML) 1.0. Recomendación de la W3C. University of Illinois at Chicago. Febrero 1998.

CEBALLOS Javier. Enciclopedia de Microsoft Visual C#. Alfaomega. 2006.

CHAMORRO Vicente, REVELO Oscar. Simulación Digital Un Primer Contacto, Universidad de Nariño, 2006.

—. 2005. DESARROLLADOR 5 ESTRELLAS. Framework 3.5. Microsoft [Citado el: 18 de Noviembre de 2010.] http://mslatam.com/latam/msdn/comunidad/dce2005/.

DE ICAZA, Jepson. Mono & .NET Framework: Una alternativa Open Source. Dr Dobb's. España. Nº1. Marzo 2002.

DEITEL, Harvey M., DEITEL, Paul J., Cómo programar en Java, Pearson Educación, 2004.

DORADO, Domínguez. Todo Programación. Nº 1. Págs. 24-26. Introducción a las aplicaciones Web con ASP e IIS. Editorial Iberprensa. Madrid. Julio, 2004.

FOWLER, M, SCOTT, K. 1999. UML Gota a Gota. México: Addison Wesley Longman de México. 1999.

GIBERT, M. PEÑA, A. Ingeniería del software en entornos de SL. España: Fundación per a la Universitat Oberta de Catalunya. 2005.

GIRONÉS, Jesús Tomás. El gran libro de Android, Tercera edición. Barcelona, 2013.

GROUSSARD, Thierry. Java 7: Los fundamentos del lenguaje Java. Ediciones ENI. Barcelona 2012.

HELJSBERG, Anders, WILTAMUTH, Scott, GOLDE, Peter. “The C# Programming Language”, Pearson Education, Inc., Boston, 2004.

INVOP (INVestigación de OPeraciones 1.0): este software fue desarrollado por Beatriz Loubet y Sandra Segura de la Universidad Nacional de Cuyo (Argentina) en 1997.

JALDIN, Rolando. 2010. Blog Activo. Modelado del Negocio - RUP [Citado el: 8 de 10 de 2010]. http://rolandojaldin.blogspot.com/2010/10/modelado-del-negocio-rup.html

KIRTLAND, Mary. A Platform for Web Services. Microsoft Developer Network. http.//msdn.microsoft.com

LOUBET Beatriz, SEGURA Sandra. INVOP (INVestigación de OPeraciones 1.0), Universidad Nacional de Cuyo. Argentina. 1997.

MARCOS, Javier, NAVARRO, Moya. Investigación de Operaciones, Transporte y Asignación, Editorial EUNED. 1998.

MATHUR, May. Investigación de operaciones. El arte de la toma de decisiones. 2ª Edición. Prentice Hall.1997.

MICROSOFT .NET: Construyendo la 3ª generación de Internet. Monográfico Byte Nº 3, MKM Publicaciones, 2001.

MCHOES Ann Mclver, FLYNN Ida M. Sistemas Operativos, sexta edición, 2011. Cengage Learning Editores S.A. México D.F.

MONTERO, Ramón. XML, el lenguaje universal. Manual formativo. Acta 13. 1999.

MOSKOWITZ Herbert y otros. Investigación de Operaciones. Prentice Hall. 1982.

MOYA, Marcos Javier. Investigación de Operaciones. Transporte y asignación. 3° edición. Universidad Estatal a Distancia San José. Costa Rica. 1998.

PIATTINI Mario G. y otros. Análisis y Diseño de Aplicaciones Informáticas de Gestión: Una perspectiva de Ingeniería del Software. Editorial Ra-Ma. 2003.

PONS Nicolás. Linux, Principios básicos del uso del sistema. Ediciones ENI, Barcelona 2005.

PRESSMAN Roger. Ingeniería del Software: Un Enfoque Práctico. McGraw-Hill. 2006.

RAMOS, Andrés. Modelos matemáticos de optimización. Departamento de organización industrial. Universidad Pontificia Comillas Madrid. Escuela Superior de Ingeniería. 2001.

RUANO Jairo, CABRERA Christian. DIONISIO: Prototipo de modelador de problemas de Simplex, asignación y transportes. Universidad de Nariño. Pasto Nariño 2010.

RUSSELL, Stuart J, NORVIG Peter. Inteligencia artificial. Un enfoque moderno. Editorial: Pearson. 2004.

SCHACH, Stephen R. Ingeniería de Software Clásica y Orientada a Objetos. McGraw-Hill. 2006.

SCOTT Short. Creación de servicios Web XML para la plataforma Microsoft® .NET, McGraw Hill/Interamericana de España, S.A.U. 2002.

SOMMERVILLE, Ian. Ingeniería de Software, Pearson Educación, Madrid. España. 2005.

TAHA Hamdy A. Investigación de Operaciones. 7ª Edición. University of Arkansas, Fayetteville. Person Educación. México. 2004.

TORA. Software incluido en el libro de Hamdy Taha. Investigación de Operaciones. 7ª Edición. 2004.

—. 2009. Modelo de transportes. Instituto Tecnológico De Pachuca [Citado el: 19 de 11 de 2009.] http://WWW.slideshare.net/iorifoar/modelo-de-transporte-2532766.
.

—. 2012. Ingeniería en Desarrollo de Software. Modelado de Negocios. Ciencias Exactas. Ingenierías y Tecnología. Ingeniería en Desarrollo de Software – ESAD. http://WWW.tucancunix.net/ceh/esad/4to/MD/04_PD_DS_MDN.pdf

—. 2010. Windows Communication Foundation. [Citado el: 19 de 02 de 2010.] Microsoft. http://msdn.microsoft.com/es-es/library/ms735119(v=vs.90).aspx.

VAUGHAN, Nichols. Web Services: Beyond the Hype. IEEE Computer, Febrero 2002.

WEITZENFELD, Alfredo. Ingeniería de Software Orientada a Objetos: Teoría y Práctica con UML y Java. Thomson Paraninfo. 2005.

WHITEPAPER. La arquitectura SOA de Microsoft® aplicada al mundo real, 2006.

YIH, Long Chang. KIRAN Desai. WINQSB (modulo: Network Modeling NET), 2002.