UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Documento de especificación de requisitos

HERRAMIENTA PARA LA OPTIMIZACIÓN DE REDES DE DISTRIBUCIÓN DE AGUA POTABLE

EQUIPO DE DESARROLLO:

Nombre	**Rol**	**Contacto**
Gabriel Sanhueza Fuentes	Administrador, Analista, Diseñador, Implementador y Tester.	gsanhueza15@alumnos .utalca.cl

CONTRAPARTE:

Nombre	**Rol**	**Contacto**
Jimmy H. Gutiérrez-Bahamondes	Cliente/Profesor guía	
Jimmy H. Gutiérrez-Bahamondes	Cliente/Profesor co-guía	

HISTORIAL DE CAMBIOS

Version	**Fecha**	**Modificaciones**
0.1	07/09/2019	Primer borrador
1	15/06/2020	Actualización documento. Agregados los requisitos faltantes.

TABLA DE CONTENIDOS

página

1. Introducción

El proyecto que se desarrollara consiste en la creación de una herramienta que haga uso de algoritmos metaheurísticos para tratar y minimizar problemas existentes en redes de distribución de agua potable.

Este proyecto solo abarcara dos problemas existentes dentro de las redes de distribución de agua potable.

Para el desarrollo de este proyecto se usarán librerías ya existentes con el fin de reducir el tiempo de desarrollo. Estas librerías son epanet2.dll, desarrollada en c, y JNA, librería en java para funciones nativas.

1.1. Propósito del Sistema

El proyecto consiste en el desarrollo de un sistema que permita simular y buscar soluciones a problemas presentes en las redes de distribución de agua potable haciendo uso de algoritmos metaheurísticos. Adicionalmente, el sistema será diseñado de tal forma que pueda ser extendido añadiendo nuevos problemas, algoritmos u operadores.

1.2. Alcance del proyecto

Al final del periodo de desarrollo la herramienta contara con las siguientes prestaciones.

- El sistema permitirá la carga y la visualización de la red gráficamente.
- El sistema solo resolverá dos clases de problemas de optimización, uno mono-objetivo y el otro multiobjetivo. El problema mono-objetivo será el de los costos

de inversión. En cuanto al problema multiobjetivo, este será el de los costos energéticos y el número de encendidos y apagado de las bombas.

- El sistema únicamente contara con dos algoritmos implementados los cuales serán el algoritmo genético y NSGA-II. El algoritmo genético será el usado para tratar el problema mono-objetivo, mientras que NSGA-II será aplicado al multiobjetivo.
- El sistema permitirá visualizar y guardar las soluciones de los algoritmos en un archivo.
- El sistema permitirá que el usuario agregue nuevos algoritmos, operadores o problemas sin tener que modificar la interfaz de usuario.

Este proyecto no contempla la creación de la red por lo que estas deberán ser ingresadas como entradas al programa.

Además, esta herramienta únicamente podrá ser ocupada en equipos cuyo sistema operativo sea Windows debido a que se realizan llamadas a librerías nativas.

1.3. Contexto

Este sistema será desarrollado utilizando el lenguaje de programación java. Debido a que este es un lenguaje ampliamente utilizado y que cuenta con un gran soporte y comunidad que lo utilizan.

Como motor de cálculo para llevar a cabo las simulaciones se utilizará una librería desarrollada en c, que cuenta con funciones para llevar a cabo simulaciones de redes de agua potable. El nombre de esta librería es epanet2.dll. Las funciones que incorpora esta librería se encuentran explicadas en [3].

Desde lenguaje se realizarán llamadas a librerías nativas usando la librería JNA existente en java. Esta librería cuenta con las clases y métodos necesarios para poder acoplar este sistema a la librería epanet2.dll desarrollada en c y que será usada como motor de cálculo para llevar a cabo las simulaciones. Puesto que una de las funcionalidades del sistema es permitir la ejecución de algoritmos metaheurísticos, se toma como base la arquitectura presentada por el framework JMetal.

JMetal es un framework para la optimización multiobjetivo con metaheurísticas. Su arquitectura inicial [1] involucraba una serie de problemas y dificultaban la

realización de ciertas acciones que eran recurrentes. Además, esta no hacia uso de las novedades incorporadas por Java como los genéricos. Es por esto, que posteriormente fue rediseñada, haciendo uso de patrones de diseño, principios de la programación orientada a objetos y aprovechando las características del lenguaje Java. Este rediseño se presenta en [2].

El contexto en el que se desenvolverá este sistema será en ambientes universitario, de investigación y en el ambiente laboral.

1.4. Definiciones, Acrónimos y Abreviaturas

NSGA-II: Non-dominated Sorting Genetic Algorithm

1.5. Referencia

[1] Juan J. Durillo, Antonio J. Nebro, and Enrique Alba. The jMetal framework for multi-objective optimization: Design and architecture. *2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, pages 1–8, 2010.

[2] Antonio J. Nebro, Juan J. Durillo, and Matthieu Vergne. Redesigning the jMetal multi-objective optimization framework. In *GECCO 2015 - Companion Publication of the 2015 Genetic and Evolutionary Computation Conference*, 2015.

[3] Lewis Rossman. *EPANET 2.0 en español.Analisis hidraulico y de calidad del agua en redes de distribución de agua. manual del usuario.* 2017.

2. Descripción General

En esta sección se describirán las características de los usuarios que harán uso del sistema. Además, se mencionará el ambiente operacional de la solución y la relación que este proyecto tiene con otros proyectos. Finalmente, también se mencionará las restricciones generales que existe en la realización de esta herramienta.

2.1. Características de los Usuarios

Este sistema solo cuenta con un tipo de usuario el cual tendrá acceso a todas las funcionalidades. Se espera que el usuario que este sistema sean ingenieros, investigadores u personas cuenten con el conocimiento básico acerca de redes de distribución de agua potable.

2.2. Ambiente operacional de la solución

El ambiente operacional en el que se desarrolla el sistema es el siguiente:

- Intel(R) Core(TM) i7-7700HQ CPU @ 2.80Ghz 2.8Ghz
- RAM 16GB DDR4
- HDD 7200rpm 1T
- SSD 256GB PCIe NVME M.2
- Windows 10 x64
- NVIDIA GeForce GTX 1050

2.3. Relación con otros proyectos

El sistema depende de la librería nativa epanet2_64bit.dll, ya que usara esta librería como motor de cálculo. La librería cuenta con 54 funciones dentro de las cuales se encuentran funciones para correr las simulaciones, modificar y obtener configuraciones de la red, modificar los elementos que conforman la red y generar reportes.

Las llamadas desde java a la librería nativa serán realizadas a través de la librería JNA.

Adicionalmente, el sistema toma la arquitectura utilizada por JMETAL como base para agregar los algoritmos, operadores y problemas. Esta arquitectura será modificada según se necesite para satisfacer los requisitos del sistema.

2.4. Restricciones Generales

- La red será ingresada como entrada al programa a través de un archivo .inp.
- La herramienta solo estará disponible para el sistema operativo Window.

3. Requisitos

En esta sección se presentan los requisitos que pueden están sujetos a posibles cambios en las siguientes iteraciones.

3.1. Requisitos de usuario

A continuación, se presentarán los requisitos de usuarios que han sido obtenidos para el desarrollo de este proyecto

RU001 – Cargar una red.	
Descripción:	La red que es representada por el archivo .inp debe ser cargada en el programa para poder llevar a cabo la simulación.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	1
Tipo:	Funcional

RU002 – Resolver el problema monoobjetivo (Pipe Optimizing) usando el Algoritmo Genético.	
Descripción:	El algoritmo genético debe ser aplicado para resolver el problema monoobjetivo que tiene como función objetivo el costo de inversión y como variable de decisión el diámetro de las tuberías.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU003 – Resolver el problema multiobjetivo (Pumping Scheduling) usando el Algoritmo NSGA-II.	
Descripción:	El algoritmo NSGA-II debe ser aplicado al problema multiobjetivo cuyas funciones a optimizar son los costos energéticos y el número de encendido y apagado de las bombas (Pumping Schedule).
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	4
Tipo:	Funcional

RU004 – Visualizar red en una interfaz gráfica.	
Descripción:	Se debe mostrar en la interfaz gráfica una representación de la red (Un dibujo, etc) generada a partir de la información contenida en el archivo inp.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU005 – Exportar los resultados de los algoritmos aplicados en dos archivos, uno para las variables y otro para los objetivos.	
Descripción:	Se deben poder exportar las soluciones generados por la ejecución de los algoritmos sobre un problema a un conjunto de archivos. Específicamente, serian 2 archivos. El primer archivo guardara las variables de las soluciones mientras que el segundo archivo guardara los objetivos de las soluciones.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU006 – Implementar el operador IntegerSBXCrossover.	
Descripción:	El operador IntegerSBXCrossover es uno de los operadores de cruzamiento. En base a cálculos probabilísticos combina dos soluciones para crear unas dos nuevas soluciones hijas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU007 – Implementar el operador IntegerSinglePointCrossover.	
Descripción:	El operador IntegerSinglePointCrossover es un operador de cruzamiento. Viendo la solución como un vector, este operador toma dos soluciones y elige un punto a partir del cual los valores de una solución se intercambiarán con los valores de otra solución. Este operador usa una probabilidad de cruzamiento y solamente realiza el intercambio de los valores en la solución cuando un numero generado aleatoriamente es menor que la probabilidad de cruzamiento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU008 – Implementar el operador IntegerPolynomialMutation.	
Descripción:	El operador IntegerPolynomialMutation es un operador de mutación. Este operador de mutación usa cálculos probabilísticos para mutar algunos variables de decisión que forman parte de la solución.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU009 – Implementar el operador IntegerSimpleRandomMutation.	
Descripción:	El operador IntegerSimpleRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU010 – Implementar el operador IntegerRangeRandomMutation (Asi lo llame).	
Descripción:	El operador IntegerRangeRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio que se encuentre entre un rango establecido. Ejemplo: Variable de decisión: 3 Rango: 2 La variable de decisión después de aplicado el operador puede tomar un valor entre [1, 5].
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU011 – Implementar el operador UniformSelection.	
Descripción:	El operador UniformSelection es un operador de selección. Este operador de selección ordena la población y asigna una probabilidad máxima y mínima a la mejor y peor solución respectivamente. A las soluciones que se encuentran entre la mejor y la peor solución se le asigna una probabilidad que se encuentra entre la probabilidad máxima y mínima. Si la probabilidad de la solución es mayor a 1.5 entonces la solución se duplica en la nueva población. Si la probabilidad esta entre 0.5 y 1.5, entonces en la nueva población se agrega la solución solo una vez. Las soluciones cuya probabilidad es menor que 0.5 no aparecen en la nueva población.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RU012 – Crear archivo .inp de la solución generada.	
Descripción:	Al ejecutar un algoritmo metaheurístico este devuelve una o un conjunto de soluciones. A partir de alguna de estas soluciones se debe crear un archivo inp en el que se vean reflejados los resultados de la solucion.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	15/10/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU013 – Mostrar las soluciones de los algoritmos en la interfaz de usuario.	
Descripción:	Mostrar los resultados de la ejecución del algoritmo, es decir las variables y los valores objetivos resultantes. Adicionalmente, estas deben poder ser guardadas en el equipo. Las variables de decisión se guardarán en un archivo (VAR), mientras que los valores de los objetivos se guardarán en otro (FUN).
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU014 – Mostrar las características de la red.	
Descripción:	Mostrar las características que posee la red. Esto puede ser realizado cuando se presiona el elemento de la red o agregando algún componente que muestre los elementos que conforman la red.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU015 – Graficar las soluciones.	
Descripción:	Mostrar en un plano cartesiano las soluciones que se van obteniendo a medida que se ejecuta el algoritmo. Solo considerar hasta 2 objetivos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RU016 – Hacer el programa fácil de ampliar.	
Descripción:	El programa debe poder fácilmente agregar nuevos algoritmos, operadores y problemas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Critica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	No Funcional

RU017 – Mostrar en la interfaz de usuario los problemas, agrupando los algoritmos que pueden ser utilizados para resolverlos.	
Descripción:	En el menú donde se muestran los problemas que pueden ser resueltos debe aparecer el nombre del problema y en este se deben agrupar los algoritmos que pueden ser utilizados para resolverlo. EJ: Pumping Scheduling ¿NSGAII ¿SPA2
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RU018 – Permitir realizar múltiples simulaciones independientes para resolver el problema multiobjetivo.	
Descripción:	Durante la resolución del problema multiobjetivo se debe poder escoger cuantas veces se aplicará el algoritmo, independientemente uno de otros. La solución final será la Frontera de Pareto del conjunto formado por todas las soluciones generadas a partir de cada ejecución independiente del algoritmo. Este concepto se conoce en algunos framework como Experimentos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RU019 – Guardar los resultados temporales por cada simulación independiente del problema multiobjetivo y generar los archivos al final de todas las simulaciones con los mejores resultados obtenidos.	
Descripción:	Las simulaciones multiobjetivos permiten realizar múltiples simulaciones independientes, entregando cada ejecución como resultado el conjunto de soluciones que conforman su frontera de Pareto. Sin embargo, una vez se terminan todas las repeticiones y se genera la solución final, se pierden los resultados intermedios de cada repetición. Es por ello, que mientras se van terminando cada simulación independiente, se debe guardar un respaldo de los resultados de cada repetición del algoritmo.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RU020 – Permitir realizar simulaciones hidráulicas utilizando los valores por defectos que vienen en el archivo .inp y visualizar los resultados.	
Descripción:	Utilizando los valores que vienen por defecto en el archivo inp se debe poder llevar a cabo la simulación hidráulica de la red. Posteriormente, los resultados podrán ser visualizados por el usuario.
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RU021 – Agregar el algoritmo multiobjetivo SMPSO.	
Descripción:	Con el fin de comprobar que se pueden acoplar nuevos algoritmos se solicita por parte del usuario que se agregue el algoritmo SMPSO.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU022 – Agregar el algoritmo multiobjetivo SPA2.	
Descripción:	Con el fin de comprobar que se pueden acoplar nuevos algoritmos se solicita por parte del usuario que se agregue el algoritmo SPA2.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU023 – Incluir en el dibujo de la red símbolos para diferencias los elementos que componen la red.	
Descripción:	Se requiere que se agreguen en la representación gráfica de la red símbolos para distinguir los distintos elementos que la componen.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU024 – Permitir realizar múltiples simulaciones independientes para resolver el problema monoobjetivo.	
Descripción:	Durante la resolución del problema monoobjetivo se debe poder escoger cuantas veces se repetirá el algoritmo, siendo independiente una repetición de la otra. Al finalizar cada repetición se solicita mostrar los resultados de cada repetición.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU025 – Agregar un menú de configuración.	
Descripción:	Agregar un menú de configuraciones donde poder establecer el tamaño de los puntos, si mostrar o no el grafico de resultados y limitar el número de resultados retornados por el problema multiobjetivo.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU026 – Usar en el gráfico, para cada repetición del algoritmo en un experimento, un color distinto.	
Descripción:	Para facilitar la distinción entre cada repetición de un algoritmo en un experimento se solicita variar que se cambie el color mostrado para cada repetición.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU027 – Mostrar una leyenda que pueda ser activada y desactivada con los símbolos mostrados sobre el dibujo de la red.	
Descripción:	Se requiere que en la ventana de representación de la red se muestre una leyenda en la que se presenten los símbolos utilizados por la aplicación para representar cada componente de la red.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU028 – Mostrar en la ventana de configuración información sobre el problema, como los objetivos, el nombre del algoritmo a utilizar y el nombre del problema que se quiere resolver.	
Descripción:	Al momento de querer realizar una optimización no se entrega suficiente información acerca de los objetivos del problema. Es por esto, que se requiere poder agregar alguna descripción que pueda ser visualizada cuando se abra la ventana de configuración del experimento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU029 – Añadir en la ventana de resultados del problema columnas extras que muestren las configuraciones utilizadas con el problema.	
Descripción:	Se solicita que se pueda agregar mas información a la ventana de resultados mostrada cuando se termina una optimización. Se propone usar un mapa con los valores adicionales que se quieran mostrar, en donde la clave sea el nombre de la columna y el valor sea lo mostrado en la celda.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU030 – Exportar los resultados de los algoritmos aplicados como un Excel.	
Descripción:	Se requiere exportar la table de resultados de la optimización del algoritmo a un archive excel.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU031 – Exportar el gráfico utilizado para mostrar visualmente las soluciones a una imagen (PNG o SVG)	
Descripción:	Se requiere exportar el gráfico de resultados mostrado durante la ejecución de un experimento, ya sea para el problema monoobjetivo como el multiobjetivo. Se requiere idealmente SVG, en caso de que este formato no sea posible se acepta la utilización de PNG.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RU032 – Permitir indicar valores por defecto a los operadores o a los problemas.	
Descripción:	Se solicita que se puedan ingresar valores por defectos que puedan ser visualizados en la ventana de configuración del problema antes de llevar a cabo la resolución del algoritmo. Los valores por defecto deben ser agregados donde se esperen recibir valores numéricos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

3.2. Requisitos de sistema

En esta sección se presentarán los requisitos de sistema obtenido a partir de los requisitos de usuarios.

RS001 – Leer red .inp.	
Descripción:	Leer un archivo inp desde la aplicación.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	1
Tipo:	Funcional

RS002 – Cargar red dentro del programa.	
Descripción:	Generar una representación de la red en el programa a partir del archivo leído, es decir, almacenar crear una jerarquía de clases que contenga los valores de la red leidos del archivo inp.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	1
Tipo:	Funcional

RS003 – Implementar algoritmo genético.	
Descripción:	Implementar el algoritmo genético. La versión del algoritmo genético a ser implementado consiste en el algoritmo genético generacional.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS004 – Implementar el problema multiobjetivo Pipe Optimizing.	
Descripción:	Implementar la clase que lleva a cabo la evaluación de las soluciones generadas por los algoritmos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS005 – Utilizar algoritmo genético para resolver el problema Pipe Optimizing desde el enfoque monoobjetivo.	
Descripción:	Utilizando el algoritmo genético y la clase que lleva a cabo la evaluación de las soluciones (La clase problema) se debe realizar la evaluación y optimización de los objetivos del problema.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS006 – Evaluar soluciones al problema monoobjetivo (Pipe optimizing) usando la librería Epanet.	
Descripción:	Se deben evaluar el cumplimiento de las restricciones establecidas para el problema. La restricción establecida para el problema Pipe Optimizing es cumplir con un nivel de presión mínimo.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS007 – Implementar NSGA-II.	
Descripción:	Implementar el algoritmo NSGA-II. El cual es usado para tratar con problemas multiobjetivo.
Fuente:	Jimmy Gutiérrez
Prioridad:	
Estabilidad:	
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	4
Tipo:	Funcional

RS008 – Implementar el problema multiobjetivo Pumping Scheduling.	
Descripción:	Implementar la clase que lleva a cabo la evaluación de las soluciones generadas por los algoritmos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	4
Tipo:	Funcional

RS009 – Utilizar el algoritmo NSGA-II para resolver el problema Pumping Scheduling desde el enfoque multiobjetivo.	
Descripción:	Utilizando el NSGA-II y la clase que lleva a cabo la evaluación de las soluciones (La clase problema) se debe realizar la evaluación y optimización de los objetivos del problema.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	4
Tipo:	Funcional

RS010 – Evaluar soluciones al problema multiobjetivo (Pumping Scheduling) usando la librería Epanet.	
Descripción:	Se deben evaluar el cumplimiento de las restricciones establecidas para el problema.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	01/10/2019
Estado:	Cumple
Incremento:	4
Tipo:	Funcional

RS011 – Visualizar red dentro de un canvas.	
Descripción:	Se debe mostrar la representación gráfica de la red utilizando un Canvas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS012 – Crear archivo TSV para guardar los valores de los objetivos.	
Descripción:	Se debe crear un archivo TSV (archivo de texto separado por tabuladores) con los objetivos de las soluciones obtenidas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS013 – Crear archivo TSV para guardar los valores de las variables.	
Descripción:	Se debe crear un archivo TSV (archivo de texto separado por tabuladores) con las variables de las soluciones obtenidas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	09/09/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS014 – Implementar el operador IntegerSBXCrossover.	
Descripción:	El operador IntegerSBXCrossover es uno de los operadores de cruzamiento. En base a cálculos probabilísticos combina dos soluciones para crear unas dos nuevas soluciones hijas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS015 – Implementar el operador IntegerSinglePointCrossover.	
Descripción:	El operador IntegerSinglePointCrossover es un operador de cruzamiento. Viendo la solución como un vector, este operador toma dos soluciones y elige un punto a partir del cual los valores de una solución se intercambiarán con los valores de otra solución. Este operador usa una probabilidad de cruzamiento y solamente realiza el intercambio de los valores en la solución cuando un numero generado aleatoriamente es menor que la probabilidad de cruzamiento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS016 – Implementar el operador IntegerPolynomialMutation.	
Descripción:	El operador IntegerPolynomialMutation es un operador de mutación. Este operador de mutación usa cálculos probabilísticos para mutar algunos variables de decisión que forman parte de la solución.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS017 – Implementar el operador IntegerSimpleRandomMutation.	
Descripción:	El operador IntegerSimpleRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS018 – Implementar el operador IntegerRangeRandomMutation (Asi lo llame).	
Descripción:	El operador IntegerRangeRandomMutation es un operador de mutación. Este operador muta una variable de decisión cuando un numero generado aleatoriamente es menor que la probabilidad de mutación establecida. El operador recorre cada variable de decisión realizando lo descrito anteriormente. La mutación realizada por este operador consiste en cambiar el valor de la variable de decisión por otro valor aleatorio que se encuentre entre un rango establecido. Ejemplo: Variable de decisión: 3 Rango: 2 La variable de decisión después de aplicado el operador puede tomar un valor entre [1, 5].
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS019 – Implementar el operador UniformSelection.	
Descripción:	El operador UniformSelection es un operador de selección. Este operador de selección ordena la población y asigna una probabilidad máxima y mínima a la mejor y peor solución respectivamente. A las soluciones que se encuentran entre la mejor y la peor solución se le asigna una probabilidad que se encuentra entre la probabilidad máxima y mínima. Si la probabilidad de la solución es mayor a 1.5 entonces la solución se duplica en la nueva población. Si la probabilidad esta entre 0.5 y 1.5, entonces en la nueva población se agrega la solución solo una vez. Las soluciones cuya probabilidad es menor que 0.5 no aparecen en la nueva población.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	00
Estado:	Cumple
Incremento:	2
Tipo:	Funcional

RS020 – Aplicar una solución al objeto que representa la red.	
Descripción:	Cuando el algoritmo haya generado soluciones para el problema, se debe poder seleccionar una de ellas y aplicarlas sobre un objeto ?Network? el cual podrá ser exportado posteriormente a un archivo inp.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	15/10/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS021 – Exportar el objeto red a un archivo .inp.	
Descripción:	Se debe exportar el objeto ?Network? a un archivo .inp, el cual debe poder ser cargado en el programa de simulación Epanet.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	15/10/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS022 – Crear la pestaña que permite visualizar los resultados.	
Descripción:	Cuando el experimento haya terminado su ejecución se debe crear un nuevo componente para mostrar los resultados.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS023 – Mostrar los resultados de la optimización.	
Descripción:	En la pestaña de resultados abierta se deben mostrar los resultados de la optimización.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS024 – Implementar un componente que permita mostrar los elementos que componen la red.	
Descripción:	El componente debe poder filtrarse por el tipo de elemento que compone la red.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS025 – Implementar componente que permitan mostrar las características de un elemento seleccionado de la red.	
Descripción:	Se debe crear una ventana o un componente en el que se muestres la configuración de los elementos de la red. Este componente debe actualizarse cada vez que se seleccione un nuevo elemento, ya sea desde la lista o desde la representación gráfica de la red.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS026 – Permitir seleccionar en la lista de elementos de la red el componente del que se van a mostrar sus características.	
Descripción:	En la lista de elementos de la red, al hacer doble click, se debe mostrar el componente que muestra las configuraciones que están establecidas para el elemento seleccionado.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS027 – Permitir seleccionar el dibujo de la red el componente del que se van a mostrar sus características.	
Descripción:	Al hacer doble click sobre un elemento, en el canvas que representa la red, se debe abrir el componente para ver la configuración que tiene establecida dicho elemento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS028 – Implementar un componente que muestre un plano cartesiano.	
Descripción:	Este componente gráfico debe tener un plano cartesiano. El plano solo puede ser ocupado cuando la optimización es de como máximo dos objetivos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS029 – Mostrar las soluciones del experimento monoobjetivo en el plano cartesiano.	
Descripción:	El componente de gráficos debe permitir agregar el valor de la solución al plano cartesiano. El plano cartesiano tendrá en el eje y el objetivo, y en el eje x el número de generaciones.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS030 – Mostrar las soluciones del experimento multiobjetivo en el plano cartesiano.	
Descripción:	El componente de gráficos debe permitir agregar el valor de la solución al plano cartesiano. El plano cartesiano usará en el eje x el objetivo 1, mientras que el objetivo 2 estará en el eje y.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	Funcional

RS031 – Implementar una jerarquía de clases que a través de la implementación de interfaces se pueda ampliar los algoritmos.	
Descripción:	Implementar una jerarquía de clases que permita agregar nuevos algoritmos o modificar los ya existentes fácilmente. Entendiendo por fácilmente, que solo se necesita implementar una interfaz para agregar un nuevo algoritmo o extender alguna clase de un algoritmo ya existente para modificar su comportamiento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	No Funcional

RS032 – Implementar una jerarquía de clases que a través de la implementación de interfaces se pueda ampliar los problemas.	
Descripción:	Implementar una jerarquía de clases que permita agregar nuevos problemas fácilmente. Entendiendo por fácilmente, que solo se necesita implementar una interfaz para agregar un nuevo problema o extender de alguna clase de un problema ya existente para modificar su comportamiento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	No Funcional

RS033 – Implementar una jerarquía de clases que a través de la implementación de interfaces se pueda ampliar los operadores.	
Descripción:	Implementar una jerarquía de clases que permita agregar nuevos operadores o modificar los ya existentes fácilmente. Entendiendo por fácilmente, que solo se necesita implementar una interfaz para agregar un nuevo operador o extender alguna clase de un operador ya existente para modificar su comportamiento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	30/11/2019
Estado:	Cumple
Incremento:	3
Tipo:	No Funcional

RS034 – Agrupar para cada problema mostrado en el menú de la interfaz de usuario los algoritmos que pueden ser usados.	
Descripción:	Dentro del menú de optimización cada problema será un nuevo menú, los cuales indicaran los algoritmos que pueden ser usados.
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS035 – Implementar mecanismo que permita realizar múltiples repeticiones del mismo algoritmo para un problema específico desde el enfoque multiobjetivo.	
Descripción:	Durante la resolución del problema multiobjetivo se debe poder escoger cuantas veces se aplicará el algoritmo, independientemente uno de otros. La solución final será la Frontera de Pareto del conjunto formado por todas las soluciones generadas a partir de cada ejecución independiente del algoritmo. Este concepto se conoce en algunos framework como Experimentos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS036 – Mantener los resultados de cada repetición de un algoritmo cuando se resuelve un problema multiobjetivo.	
Descripción:	Al resolver un problema multiobjetivo se deben mantener los resultados generados cuando se termina cada repetición del algoritmo dentro de un experimento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS037 – Guardar los resultados almacenados de cada repetición del algoritmo multiobjetivo.	
Descripción:	Los resultados mantenidos mientras se iban terminando las ejecuciones independientes del algoritmo configurado para el experimento, deben ser guardados en la carpeta que indica el Experimento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS038 – A partir de los resultados de cada repetición del algoritmo, para el problema multiobjetivo, obtener las mejores soluciones.	
Descripción:	A partir de los resultados generados por cada repetición del algoritmo se debe obtener las soluciones no dominadas.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS039 – Guardar las mejores soluciones obtenidas a partir de cada una de las repeticiones del algoritmo.	
Descripción:	Guardar las soluciones no dominadas obtenidas al unir los resultados de cada repetición del algoritmo dentro de un experimento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS040 – Realizar una simulación hidráulica utilizando los valores por defecto de la red que vienen en el archivo inp.	
Descripción:	Se debe poder realizar la simulación hidráulica utilizando únicamente los valores preconfigurados en el archivo inp.
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS041 – Obtener los resultados de la simulación hidráulica desde el simulador.	
Descripción:	A medida que se van realizando la simulación hidráulica se deben recuperar los resultados del simulador y almacenarlos en memoria para posteriormente ser presentados al usuario.
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS042 – Mostrar los resultados de la simulación hidráulica en la interfaz de usuario.	
Descripción:	Mostrar al usuario los resultados obtenidos al realizar la simulación.
Fuente:	Daniel Mora-Meliá
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	27/01/2020
Estado:	Cumple
Incremento:	5
Tipo:	Funcional

RS043 – Implementar el algoritmo SMPSO.	
Descripción:	Agregar el algoritmo SMPSO (Multiobjetivo) y utilizarlo para resolver el problema Pumping Scheduling.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS044 – Implementar el algoritmo SPA2.	
Descripción:	Agregar el algoritmo SPA2 (Multiobjetivo) y utilizarlo para resolver el problema Pumping Scheduling.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS045 – Especificar los símbolos a utilizar para cada componente de la red.	
Descripción:	Establecer los símbolos que serán utilizados para los elementos de la red. Se pueden usar los mismos que el software de simulación hidráulica Epanet.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS046 – Agregar los símbolos de los componentes cuando se muestra la red.	
Descripción:	Agregar sobre cada elemento mostrado en la representación de la red el símbolo que lo representa y que permite distinguirlo de otros elementos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS047 – Implementar mecanismo que permita realizar múltiples repeticiones del mismo algoritmo para un problema específico desde el enfoque monoobjetivo.	
Descripción:	Durante la resolución del problema monoobjetivo se debe poder escoger cuantas veces se repetirá el algoritmo, siendo independiente una repetición de la otra. Al finalizar cada repetición se solicita mostrar los resultados de cada repetición. Así como se hace para problemas multiobjetivos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS048 – Crear una interfaz de usuario para mostrar las configuraciones.	
Descripción:	Implementar una interfaz de usuario para presentar ciertas opciones que el usuario puede configurar al utilizar la aplicación.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS049 – Establecer los valores de la aplicación que se permitirá que el usuario configure.	
Descripción:	Establecer cuáles serán los valores permitidos para que el usuario configure.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	No Funcional

RS050 – Mostrar al usuario la ventana de configuración.	
Descripción:	Agregar la funcionalidad para abrir la ventana de configuración cuando el usuario la requiera.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS051 – Aplicar al sistema las configuraciones establecidas en la ventana de configuraciones.	
Descripción:	Hacer efectivas las configuraciones configuradas en la ventana de configuración en el programa.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS052 – Escoger una paleta de colores a partir de la cual elegir el color para cada iteración.	
Descripción:	Establecer una paleta de colores entre las que poder elegir el color a ser usado por cada iteración dentro de un experimento.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	No Funcional

RS053 – Asignar a cada solución de una iteración un color dentro de la paleta de colores.	
Descripción:	Implementar la funcionalidad para que cada iteración a ser mostrada en el gráfico de resultados tome un color diferente.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Transable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS054 – Mostrar los símbolos usados en la representación de la red y que significan cada uno de ellos.	
Descripción:	Implementar código para que se muestre una leyenda con los símbolos de la red existentes y a qué pertenecen.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS055 – Agregar una opción que permita activar y desactivar la leyenda.	
Descripción:	Agregar en el menú de configuraciones la posibilidad de activar y desactivar la leyenda de símbolos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS056 – Implementar un mecanismo que permite indicar una descripción del problema a resolver.	
Descripción:	Implementar alguna manera de poder agregar una descripción al problema a tratar que debe ser mostrada en la interfaz de configuración del problema como una pestaña. Pueden ser usadas las mismas anotaciones para este fin.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS057 – Mostrar en la ventana de configuración información del problema a resolver.	
Descripción:	Modificar la ventana de configuración del problema para que se muestre la descripción del problema escogido.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS058 – Implementar un mecanismo para indicar los elementos adicionales que quieren ser mostrados en la interfaz de resultados.	
Descripción:	Implementar código para añadir información adicional a la ventana de resultados. Esta información adicional será especificada por el usuario que implemente el problema. Añadir esta información debe ser opcional. Si esta información no se añade entonces mostrar los resultados y objetivos.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS059 – Agregar los datos adicionales que desean ser mostrados en la ventana de resultados.	
Descripción:	Mostrar los parámetros adicionales en la ventana de resultados.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS060 – Exportar todos los datos de la ventana de resultados a un archivo Excel.	
Descripción:	Implementar código para exportar los valores presentes en la ventana de resultados a un archivo Excel.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS061 – Configurar la carpeta en la que se guardaran la imagen del gráfico.	
Descripción:	Mostrar una ventana para que el usuario seleccione donde quiere guardar la imagen del gráfico.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS062 – Guardar el gráfico de resultados en disco en formato PNG.	
Descripción:	Guardar el gráfico en la carpeta seleccionada.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS063 – Agregar un mecanismo para ingresar valores por defecto para los operadores.	
Descripción:	Implementar funcionalidad para agregar valores por defecto a los parámetros que pueden ser configurados desde la ventana de configuración del problema. Estos valores por defecto deben ser para los parámetros numéricos (int o double).
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

RS064 – Mostrar en la ventana de configuración de los problemas los valores por defecto.	
Descripción:	Al abrir la ventana de configuración de problemas mostrar los valores establecidos por defecto.
Fuente:	Jimmy Gutiérrez
Prioridad:	Crítica
Estabilidad:	Intransable
Fecha de actualización:	13/05/2020
Estado:	Cumple
Incremento:	6
Tipo:	Funcional

3.3. Matriz de Trazado Requisitos de Usuario vs. Requisitos de Software

La matriz de trazabilidad de los requisitos de usuario y de sistema que se presenta a continuación permite ver la relación y dependencia que un requisito de sistema tiene con los requisitos de usuario.

	RU001	RU002	RU003	RU004	RU005	RU006	RU007	RU008	RU009	RU010	RU011	RU012	RU013	RU014	RU015	RU016	RU017	RU018	RU019	RU020	RU021	RU022	F
RS001	X			X																			
RS002	X			X																			
RS003		X																					
RS004		X																					
RS005		X																					
RS006		X																					
RS007			X																				
RS008			X																				
RS009			X																				
RS010			X																				
RS011				X																			
RS012				X																			
RS013					X																		
RS014						X																	
RS015							X																
RS016								X															
RS017									X														
RS018										X													
RS019											X												
RS020													X										
RS021													X										
RS022													X										
RS023													X										
RS024														X									
RS025														X									
RS026														X									
RS027														X									
RS028															X								
RS029															X								
RS030															X								
RS031																X							
RS032																X							
RS033																X							
RS034																	X						
RS035																		X					
RS036																		X					
RS037																			X				
RS038																			X				
RS039																			X				
RS040																				X			
RS041																				X			
RS042																				X			
RS043																					X		
RS044																					X		
RS045																						X	
RS046																							
RS047																							
RS048																							
RS049																							
RS050																							
RS051																							
RS052																							
RS053																							
RS054																							
RS055																							
RS056																							
RS057																							
RS058																							
RS059																							
RS060																							
RS061																							
RS062																							
RS063																							
RS064																							

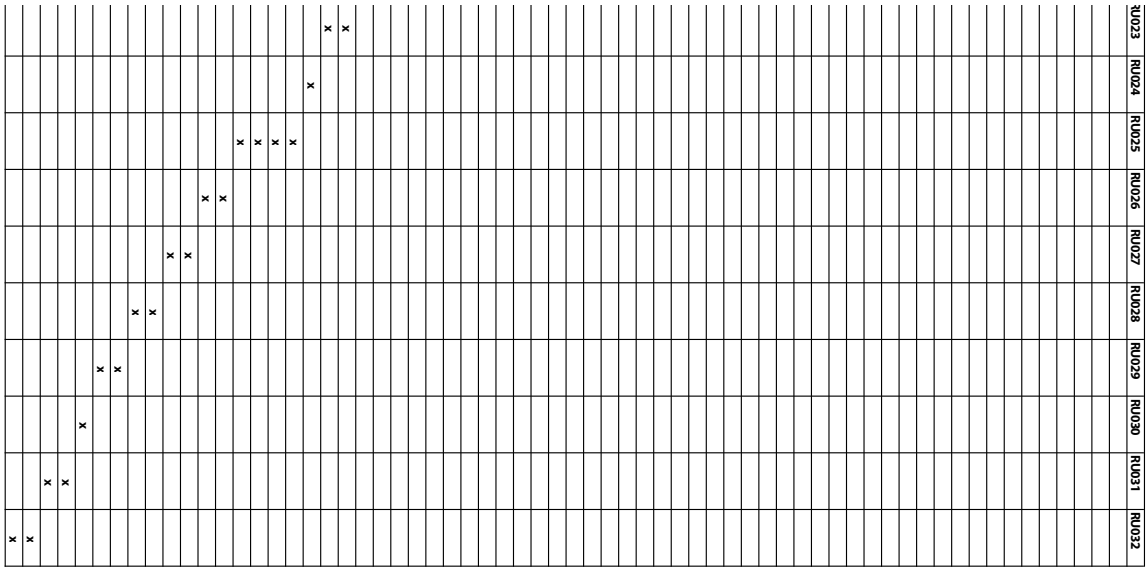

Figura 3.1: Matriz de requisito de usuario versus requisitos de sistema.