Plan de Pruebas de Sistema

|  |  |  |  |
| --- | --- | --- | --- |
| **Elaborado por:** | Click IT | **Fecha de Elaboración:** | 02-04-2014 |
| **Fecha de entrega** |  |

Control de cambios

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 02/04/2014 | 1.0 | Versión Inicial | Equipo Click IT |
| 20/04/2014 | 2.0 | Se incluyeron pruebas para las funcionalidades del ciclo 2 | Equipo Click IT |
|  |  |  |  |
|  |  |  |  |

Contenido

[1. Propósito 3](#_Toc385193823)

[2. Plan de Pruebas 3](#_Toc385193824)

[2.1. Estrategia para las pruebas 3](#_Toc385193825)

[2.2. Técnica para pruebas de sistema 3](#_Toc385193826)

[3. Especificaciones de casos de Prueba 3](#_Toc385193827)

[3.1. Casos de pruebas de caja Negra 3](#_Toc385193828)

# Propósito

Este documento tiene como objetivo realizar los planes para las pruebas de sistema que permita evaluar las funcionalidades del programa en condiciones normales e inusuales y evaluar que el sistema cumpla con los requerimientos funcionales y no funcionales.

# Plan de Pruebas

## Estrategia para las pruebas

Para las pruebas de sistema, se optó por evaluar primero las funcionalidades de bajo nivel en condiciones normales, anormales, para luego poder evaluar las funcionalidades de alto nivel.

## Técnica para pruebas de sistema

Caja negra, basándose en los requerimientos establecidos previamente y en escenarios reales.

# Especificaciones de casos de Prueba

## Casos de pruebas de caja Negra

|  |  |
| --- | --- |
| **Prueba No.:** | 1 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo valide cuando el usuario ingresa una ruta no válida o apunte a un directorio inexistente, en el campo “Ruta del proyecto”. |
| **Entradas** | Ruta de la versión anterior del programa: xx |
| **Salidas** | Ruta no válida |

|  |  |
| --- | --- |
| **Prueba No.:** | 2 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo permita seleccionar a través de la interfaz la ruta del proyecto que el usuario está trabajando y agregarla en el campo “ruta del programa” |
| **Entradas** | Seleccionar la ruta del programa por medio de la interfaz. |
| **Salidas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0 |

|  |  |
| --- | --- |
| **Prueba No.:** | 3 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo no permita al usuario adicionar una versión, sin haber seleccionado la ruta del programa previamente |
| **Entradas** | Ruta del programa:  Usuario quien realiza el cambio: Layne granados  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | No se pudo adicionar la nueva versión. Seleccione la ruta del programa. |

|  |  |
| --- | --- |
| **Prueba No.:** | 4 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo no permita adicionar una nueva versión, sin haber ingresado el campo usuario. |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio:  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | No se pudo adicionar la nueva versión. Ingrese el nombre de usuario. |

|  |  |
| --- | --- |
| **Prueba No.:** | 5 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo no permita adicionar una nueva versión, sin haber ingresado la razón de cambio. |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio: Layne Granados  Razón del cambio: |
| **Salidas** | No se pudo adicionar la nueva versión. Ingrese la razón del cambio |

|  |  |
| --- | --- |
| **Prueba No.:** | 6 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo permita adicionar una nueva versión. |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio: Layne Granados  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | Se adicionó la versión correctamente! |

|  |  |
| --- | --- |
| **Prueba No.:** | 7 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo despliegue la información de los cambios de la nueva versión, con relación a la anterior en el campo “Cambios del proyecto”, cada vez que el usuario seleccione una ruta del proyecto. |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio: Layne Granados  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | El campo “Resumen de cambios” del programa debe mostrar la siguiente información:  \*Clases eliminadas  \*Clases adicionadas  \* Cantidad Lineas Agregadas del Programa:  \*Cantidad Lineas Eliminadas del Programa:  \*Cantidad Lineas Modificadas del Programa:  \*Cantidad Total de Lineas Lógicas del Programa |

|  |  |
| --- | --- |
| **Prueba No.:** | 8 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente y enumere correctamente los archivos “adicionados” de texto plano en la versión actual con relación a la versión anterior. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario seleccione una ruta del proyecto.  Para la prueba, subiremos 2 versiones y en la última incluiremos 2 archivos de texto plano (Ej: .java) que no se encuentran en la primera versión del proyecto |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto. Pero la última versión debe contener 2 archivos nuevos de texto plano, llamados ClaseNueva1.java y ClaseNueva2.java |
| **Salidas** | \*Cantidad de archivos adicionados: 2  \*Nombre de los archivos adicionados:  -ClaseNueva1.java  -ClaseNueva2.java |

|  |  |
| --- | --- |
| **Prueba No.:** | 9 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo no cuente los archivos adicionados que no son de texto plano en la versión actual con relación a la versión anterior. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario seleccione la ruta del proyecto.  Para la prueba, subiremos 2 versiones y en la última incluiremos 1 archivos de texto plano (Ej: .java) y otro archivo que no sea de texto plano (Ej: .mp3) |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto. Pero la última versión debe contener 1 archivo nuevo de texto plano, llamado ClaseNueva1.java y 1 archivo que no sea de texto plano llamado prueba.mp3 |
| **Salidas** | \*Cantidad de archivos adicionados: 1  \*Nombre de los archivos adicionados:  -ClaseNueva1.java |

|  |  |
| --- | --- |
| **Prueba No.:** | 10 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente y enumere correctamente los archivos “eliminados” de texto plano en la versión actual con relación a la versión anterior. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario seleccione la ruta del proyecto.  Para la prueba, subiremos 2 versiones y en la primera versión incluiremos 2 archivos de texto plano (Ej: .java) que no se encuentren en la última versión del proyecto |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto. Pero la primera versión debe contener 2 archivos de texto plano que no se encuentren en la última versión, llamados ClaseEliminada1.java y ClaseEliminada2.java |
| **Salidas** | \*Cantidad de archivos eliminados: 2  \*Nombre de los archivos eliminados:  - ClaseEliminada1.java  - ClaseEliminada2.java |

|  |  |
| --- | --- |
| **Prueba No.:** | 11 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo no cuente los archivos eliminados que no son de texto plano en la versión actual con relación a la versión anterior. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario seleccione la ruta del proyecto.  Para la prueba, subiremos 2 versiones y en la última no incluiremos 1 archivo de texto plano (Ej: .java) y otro archivo que no sea de texto plano (Ej: .mp3) que se encuentren en la primera versión |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto. Pero solo la primera versión debe contener 1 archivo de texto plano, llamado ClaseEliminada1.java y 1 archivo que no sea de texto plano llamado prueba.mp3 |
| **Salidas** | \*Cantidad de archivos eliminados: 1  \*Nombre de los archivos eliminados:  - ClaseEliminada1.java |

|  |  |
| --- | --- |
| **Prueba No.:** | 12 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo despliegue la estructura del proyecto, cada vez que el usuario suba una versión del proyecto a excepción de la primera vez que se anexa el proyecto.  La estructura del proyecto desplegada, debe corresponder a la última versión ingresada. |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio: Layne Granados  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | Despliegue de la estructura del directorio de la última versión del proyecto, mediante la interfaz gráfica |

|  |  |
| --- | --- |
| **Prueba No.:** | 13 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo despliegue en la estructura del proyecto, solo archivos de texto plano. (Ejemplo: .java, .xhtml, .php, .html). Para la prueba dentro de la carpeta ingresaremos un archivo .mp3 y no se debe mostrar en la estructura del proyecto |
| **Entradas** | Ruta del programa: C:\workspace\pruebas\nuevo\_tareapsp0  Usuario quien realiza el cambio: Layne Granados  Razón del cambio: Se adicionó una clase para leer el archivo |
| **Salidas** | Despliegue de la estructura del directorio del proyecto, enlistando solo los archivos de texto plano |

|  |  |
| --- | --- |
| **Prueba No.:** | 14 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo despliegue en la estructura del proyecto, solo los archivos que se encuentran en las últimas dos versiones ingresadas. Para la prueba, subiremos en la última versión una nueva clase, como esta no existe en la versión anterior, el archivo no se debe mostrar en la estructura del proyecto |
| **Entradas** | Adicionar mediante la interfaz dos versiones, pero la última versión, debe tener una nueva clase. |
| **Salidas** | Despliegue de la estructura del directorio del proyecto, sin mostrar la clase ingresada en la última versión. |

|  |  |
| --- | --- |
| **Prueba No.:** | 15 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo despliegue en la estructura del proyecto, solo los archivos que fueron modificados en la versión más reciente. Para la prueba, subiremos 2 versiones del proyecto y en ambos adicionaremos la misma clase principal.java , como la clase no tiene modificaciones, no se debe mostrar en la estructura del proyecto |
| **Entradas** | Adicionar mediante la interfaz dos versiones con la misma clase |
| **Salidas** | Despliegue de la estructura del directorio del proyecto, sin mostrar la clase principal.java |

|  |  |
| --- | --- |
| **Prueba No.:** | 16 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que la interfaz valide que el usuario no pueda comparar archivos sin que el usuario haya seleccionado previamente el archivo a comparar. |
| **Entradas** | -El usuario anexó nueva versión del proyecto y seleccionó la opción “Comparar Archivo” |
| **Salidas** | No se puede comparar archivos. Seleccione un archivo de la estructura del proyecto actual. |

|  |  |
| --- | --- |
| **Prueba No.:** | 17 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo validar que la interfaz gráfica muestre el código completo de la versión antigua y nueva de un archivo .java que se está comparando. Cada línea de código debe indicar el número de la línea |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto, y seleccionar la clase a comparar. |
| **Salidas** | |  |  | | --- | --- | | **Clase de la versión anterior**  1== import java.io.BufferedReader;  2== import java.io.IOException;  3== import java.io.InputStreamReader;  4==  5==  6== public class main { | **Clase de la versión nueva**  1== import java.io.BufferedReader;  2== import java.io.IOException;  3== import java.io.InputStreamReader;  4==  5==  6== public class main { | |

|  |  |
| --- | --- |
| **Prueba No.:** | 18 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo validar que la interfaz gráfica muestre el código completo de la versión antigua y nueva de un archivo de texto plano diferente a un .java para realizar la comparación. Cada línea de código debe indicar el número de la línea |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto, en cada versión adicionar un archivo xhtml.  Al subir la última versión, el probador debe seleccionar para comparar el archivo xhtml. |
| **Salidas** | |  |  | | --- | --- | | **Clase de la versión anterior**  1== <!DOCTYPE composition PUBLIC "-  2== //W3C//DTD XHTML 1.0 Transitional  3==//EN"http://www.w3.org/TR/xhtml1/DTD/  4== xhtml1-transitional.dtd">  5== <ui:composition  6== xmlns=[*http://www.w3.org/1999/xhtml*](http://www.w3.org/1999/xhtml)  7== xmlns:ui=*"http://java.sun.com/*  8==*jsf/facelets"*  9== template=*"/layout/template.xhtml"*>  10==  11== <ui:define name=*"body"*> | **Clase de la versión nueva**  1== <!DOCTYPE composition PUBLIC "-  2== //W3C//DTD XHTML 1.0 Transitional  3==//EN"http://www.w3.org/TR/xhtml1/DTD/  4== xhtml1-transitional.dtd">  5== <ui:composition  6== xmlns=[*http://www.w3.org/1999/xhtml*](http://www.w3.org/1999/xhtml)  7== xmlns:ui=*"http://java.sun.com/*  8==*jsf/facelets"*  9== template=*"/layout/template.xhtml"*>  10==  11== <ui:define name=*"body"*> | |

|  |  |
| --- | --- |
| **Prueba No.:** | 19 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo validar que la interfaz gráfica al mostrar el código de la versión anterior y nueva de una clase, resalte en color verde las líneas de código modificadas. Para la prueba se tomará el programa PSP2, del cual se generarán dos versiones |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto, y seleccionar la clase Main.java |
| **Salidas** | La interfaz debe mostrar el código de la versión antigua y nueva, resaltando en color verde las líneas modificadas y estas deben tener el carácter m al lado del número de la línea   |  |  | | --- | --- | | **Clase de la versión anterior**  20== DistribucionIntegral distribucionIntegral=new DistribucionIntegral();  ***21m distribucionIntegral.setX(x);***  22== distribucionIntegral.setDof(dof);  23== Calculos calculos =new Calculos();  ***24m calculos.controlador(distribucionIntegral,0,0,0);***  25==  26== }catch(NumberFormatException e){ | **Clase de la versión nueva**  20== DistribucionIntegral distribucionIntegral=new DistribucionIntegral();  ***21m distribucionIntegral.setValorP(p);***  22== distribucionIntegral.setDof(dof);  23== Calculos calculos =new Calculos();  ***24m calculos.controlador(distribucionIntegral,0,1,true,1);***  25==  26== }catch(NumberFormatException e){ | |

|  |  |
| --- | --- |
| **Prueba No.:** | 20 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo validar que la interfaz gráfica al mostrar el código de la versión anterior y nueva de una clase, resalte en color rojo y agregue el carácter – a las líneas de código eliminadas en el campo de la versión anterior. Para la prueba se tomará el programa PSP2, del cual se generarán dos versiones |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto, y seleccionar la clase Main.java |
| **Salidas** | La interfaz debe mostrar el código de la versión antigua y nueva, resaltando en color rojo en la versión antigua las líneas eliminadas y estas deben tener los caracteres -- al lado del número de la línea   |  |  | | --- | --- | | **Clase de la versión anterior**  86== private void calcularDesviacion(){  ***87-- Iterator it = this.lista.descendingIterator();***  ***88-- double temporal = 0.0;***  ***89-- if(this.getMedia()!=-1){***  ***90-- while(it.hasNext()){***  ***91-- String num = (String)it.next();***  ***92-- double numero = Double.parseDouble(num);***  ***93-- double res = numero - this.getMedia();***  ***94-- res = res\*res;***  ***95-- temporal+=res;*** | **Clase de la versión nueva**  86== private void calcularDesviacion(){  ***87--***  ***88++***  89== }  90==  91==  92== public String calcular(){  93== this.crearLista();  94== this.calcularMedia();  95== this.calcularDesviacion(); | |

|  |  |
| --- | --- |
| **Prueba No.:** | 21 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo validar que la interfaz gráfica al mostrar el código de la versión anterior y nueva de una clase, resalte en color amarillo y agregue el carácter ++ a las líneas de código adicionadas en el campo de la versión nueva. Para la prueba se tomará el programa PSP2, del cual se generarán dos versiones |
| **Entradas** | Adicionar mediante la interfaz dos versiones del proyecto, y seleccionar la clase Main.java |
| **Salidas** | La interfaz debe mostrar el código de la versión antigua y nueva, resaltando en color amarillo en la versión nueva las líneas adicionadas y estas deben tener los caracteres ++ al lado del número de la línea   |  |  | | --- | --- | | **Clase de la versión anterior**  86== private void calcularDesviacion(){  ***87-- Iterator it = this.lista.descendingIterator();***  ***88-- double temporal = 0.0;***  ***89-- if(this.getMedia()!=-1){***  ***90-- while(it.hasNext()){***  ***91-- String num = (String)it.next();***  ***92-- double numero = Double.parseDouble(num);***  ***93-- double res = numero - this.getMedia();***  ***94-- res = res\*res;***  ***95-- temporal+=res;*** | **Clase de la versión nueva**  86== private void calcularDesviacion(){  ***87--***  ***88++***  89== }  90==  91==  92== public String calcular(){  93== this.crearLista();  94== this.calcularMedia();  95== this.calcularDesviacion(); | |

|  |  |
| --- | --- |
| **Prueba No.:** | 22 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas agregadas al programa de un archivo .java. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última incluiremos en una clase una nueva línea de código. |
| **Salidas** | \*Cantidad Líneas Agregadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 23 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas agregadas al programa de un archivo .html. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última incluiremos en un archivo html una nueva línea de código. |
| **Salidas** | \*Cantidad Líneas Agregadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 24 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas modificadas de un archivo.java. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última modificaremos una línea de código de una clase. |
| **Salidas** | \*Cantidad Líneas Modificadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 25 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas modificadas de un archivo.html. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última modificaremos una línea de código de un archivo .xhtml. |
| **Salidas** | \*Cantidad Líneas Modificadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 26 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas eliminadas de los archivos .java. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última eliminaremos una línea de código de una clase. |
| **Salidas** | \*Cantidad Líneas eliminadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 27 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente la cantidad de líneas eliminadas de archivos .xhtml. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, subiremos 2 versiones y en la última eliminaremos una línea de código de un archivo .xhtml. |
| **Salidas** | \*Cantidad Líneas eliminadas del Programa:1 |

|  |  |
| --- | --- |
| **Prueba No.:** | 28 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente las líneas lógicas del programa. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, versionaremos el proyecto de psp0 desarrollado por el líder de calidad, el cual cumple con el estándar establecido en TSP y del cual ya tenemos la cantidad de líneas lógicas. |
| **Salidas** | \*Cantidad de líneas lógicas:56 |
| **Prueba No.:** | 29 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo cuente correctamente las líneas lógicas del programa. Esta información debe mostrarse en el campo “Cambios del proyecto”, cada vez que el usuario selecciona la ruta del proyecto. |
| **Entradas** | Para la prueba, versionaremos el proyecto de psp0 desarrollado por el líder de calidad, el cual cumple con el estándar establecido en TSP y del cual ya tenemos la cantidad de líneas lógicas. |
| **Salidas** | \*Cantidad de líneas lógicas:56 |

|  |  |
| --- | --- |
| **Prueba No.:** | 30 |
| **Descripción del caso de prueba (propósito y método):** | La prueba tiene como objetivo verificar que el aplicativo, cuando el usuario selecciona por la interfaz comparar un archivo, muestre las diferencias de los dos archivos en el área “Cambios de archivo”. |
| **Entradas** | Para la prueba, versionaremos el proyecto de psp0 desarrollado por el líder de calidad, el cual cumple con el estándar establecido en TSP y del cual ya tenemos la cantidad de líneas lógicas. |
| **Salidas** | \*Cantidad de líneas lógicas:56 |