



Plan de prueba “Tomaster FM v1.0”

Equipo Desarrollador

Francisco José Adaros Tapia.
Matías Jesús Egaña Alfaro.

Asignatura

Programación Avanzada.

Profesor

Guillermo Alonso Leyton Garcia.



Propósito del documento	2
Ficha documento	2
Alcance	3
Clases del sistema	3
Pruebas	4

Propósito del documento

El presente documento tiene como objetivo la creación del exponer las pruebas a realizar al sistema (funciones, módulos, subsistemas, etc.) que actualmente se pueden aplicar, de esta forma podremos apreciar de mejor manera cómo deberían comportarse las distintas componentes del prototipo actual ante condiciones inesperadas. Estas pruebas tienen el objetivo fundamental de encontrar tantos errores como sea posible y mitigarlos antes de que este sea distribuido al cliente. Para cada unidad, se detalla el tipo de prueba a aplicar, su prioridad, datos de entrada junto a su evento esperado y por último detallar en cómo se podrá verificar que se cumpla el evento esperado.

Ficha documento

Nombre del proyecto	Asignatura
Tomaster FM	Programación Avanzada

Testers	Asignación
Bastián Rojas	Equipo SQA
Cristian Soto	Equipo SQA
Francisco Adaros	Equipo Desarrollador
Matías Egaña	Equipo Desarrollador

Alcance

Las pruebas para ésta iteración del documento Plan de Pruebas se realizarán según los siguientes requerimientos del documento “Especificación de Requerimientos Tomaster v1.2”.

Funcionalidades	Descripción
RF005	<ul style="list-style-type: none"> • Nombre: Motor de inferencia. • Descripción: El prototipo contará con un sistema que inferirá¹, por medio de los hechos, la plaga o enfermedad que esté afectando al cultivo. • Prioridad Alta
RF001	<ul style="list-style-type: none"> • Nombre: Sistema de preguntas. • Descripción: El prototipo preguntará al usuario sobre los síntomas y/o condiciones que presenta el cultivo. • Prioridad Alta
RF008	<ul style="list-style-type: none"> • Nombre: Base de recomendación. • Descripción: El prototipo contará con un dataset, donde almacenará todas las recomendaciones de las plagas o enfermedades, que identificará el sistema. • Prioridad Alta

Clases del sistema

En el siguiente apartado se presentarán las clases del prototipo actual a los cuales se le harán pruebas.

Clase [C1]: Program.cs

- Función [C1.M2]: addLink
- Función [C1.M3]: graf
- Función [C1.M3]: node
- Función [C1.M3]: BackTrack

¹ Extraer un juicio o conclusión a partir de hechos, proposiciones o principios, de lo más general a lo particular.

- Función [C1.M3]: TravelGraph
- Clase [C2]: anadirHistorial.cs
 - Función [C2.M1]: historial
- Clase [C3]: historial.cs
 - Función [C3.M1]: mostrar

Pruebas

En este apartado se presentará cada uno de los componentes y las pruebas a realizar de cada uno, detallando lo siguiente:

- Nombre del método
- Descripción de la prueba
- Método de la prueba (Caja negra / Caja blanca)
- Prioridad de la prueba (Alta / Media / Baja)
- Datos de entrada
- Pasos que sigue la función.
- Salida esperada
- Entradas no válidas

[C1.M1]	Nombre: AddLink	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Añade los enlaces correspondientes al nodo señalado.			
Entrada: <ul style="list-style-type: none"> objeto tipo Node 			
Pasos: <ol style="list-style-type: none"> //Agrega los hijos al nodo actual if (from.Value == 0) { to.preguntar = true; } from.Links.Add(to); //agrega los padres del nodo to.dad.Add(from); 			
Salida Esperada: Nada			
Entradas no válidas: El Objeto tipo nodo es null.			

[C1.M2]	Nombre: Graf	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Método utilizado para seleccionar y crear el grafo según el tipo de problema que presente el cultivo.			
Entrada: entero			
Pasos: <ul style="list-style-type: none"> si el entero ingresado es: <ol style="list-style-type: none"> 1; Genera el grafo correspondiente a las reglas pertenecientes a síntomas causados por plagas. 2; Genera el grafo correspondiente a las reglas pertenecientes a síntomas causados por enfermedades 			
Salida esperada: Objeto tipo Graph			
Entradas no válidas: <ul style="list-style-type: none"> cualquier carácter distinto a 1 o 2. 			

[C1.M3]	Nombre: node	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Contiene la estructura de cada nodo del grafo, según ésta estructura se almacenan los hechos.			
Entrada: int value correspondiente al número del nodo.			
Pasos : Get y Set de los siguientes parámetros <ul style="list-style-type: none"> • Value = value; • Links = new List<Node>(); • dad = new List<Node>(); • validated = "0"; • preguntar = false; • allValided = false; • allDenied = false; 			
Salida esperada: Regla eliminada del ArrayList.			
<ul style="list-style-type: none"> ○ cualquier valor que no corresponda con el tipo de variable del nodo. 			

[C1.M4]	Nombre: BackTrack	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Método para devolverse en el grafo.			
Entrada: objeto tipo Nodo			
Pasos: <pre> • foreach (Node dad in node.dad) • { • if (dad.valided != "1") • { • node = dad; • break; • } • else if(dad.valided == "1") • { • node = Backtrack(dad); • } • } • return node; </pre>			
Salida esperada: Nodo anterior al que se ingresó.			
Entradas no válidas: <ul style="list-style-type: none"> ○ Objeto Nodo = null 			

[C1.M5]	Nombre: TravelGraph	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Se mueve a través del grafo marcando como no válido a cualquier nodo que suceda al nodo invalidado.			
Entrada: Objeto tipo nodo.			
Pasos: <ul style="list-style-type: none"> • <code>if (node.Links.Count > 0)</code> • <code>{</code> • <code>foreach (Node n in node.Links)</code> • <code>{</code> • <code>n.validated = "2";</code> • <code>travelGraph(n);</code> • <code>}</code> • <code>}</code> • <code>}</code> 			
Salida esperada: Nodos hijos y los hijos de los hijos invalidados.			
Entradas no válidas: <ul style="list-style-type: none"> ○ Objeto tipo nodo = null 			

[C2][M1]	Nombre: anadirHistorial	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: <ul style="list-style-type: none"> • añade los campos requeridos por RF008 a un archivo donde se almacena el historial 			
Entrada: <p>String</p>			
Pasos: <ul style="list-style-type: none"> • <code>public void historial(string texto) {</code> • <code>string dir = @"C:\Users\adaro\Desktop\sintomas\historial.txt";</code> • <code>using (StreamWriter sw = File.AppendText(dir))</code> • <code>{</code> • <code>sw.WriteLine(texto);</code> • <code>}</code> • <code>}</code> 			
Salida Esperada: <ul style="list-style-type: none"> • String añadido al archivo 			
Entradas no válidas: <ul style="list-style-type: none"> • String nulo 			

[C3.M1]	Nombre: historial	Prioridad: Alto	Método de prueba: Caja Negra
Descripción: Se muestra el contenido del archivo de historial.			
Entrada: Nada.			
Pasos : <ul style="list-style-type: none"> • internal void mostrar() • { • string dir = @"C:\Users\adaro\Desktop\sintomas\historial.txt"; • //Process.Start(@"C:\Users\adaro\Desktop\sintomas\1."); • • //System.Diagnostics.Process.Start("explorer.exe", • "C:\Users\adaro\Desktop\sintomas\baki.jpg"); • string[] lineas = System.IO.File.ReadAllLines(dir); • int largo = lineas.Length; • int count = 0; • while (largo != count) • { • Console.WriteLine(lineas[count]); • count++; • } • • Console.WriteLine("\n\n presione cualquier tecla para continuar: "); • Console.ReadKey(); • } 			
Salida Esperada: Salida por pantalla del historial			
Entradas no válidas: No posee entradas.			