



“Documentación AGENDA DE TAREAS”

Alumno: Jesus Alejandro Serrano Meza

Alumno: Edgar Fabian Castañeda Segura

Ingeniería en Administración de Sistemas Computacionales, Centro Universitario Isic

Materia: Análisis y Diseño de Sistemas

Mtro: Jhonat Fernando Acevedo Polanco

4 de Octubre de 2025

ÍNDICE

Introducción.....	4
Planeación.....	5
Comunicación con el cliente:	5
Se pasa a elaborar el plan del proyecto:.....	5
Objetivo:.....	5
Metodología KANBAN.....	5
Diagrama de Gantt.....	6
Análisis de requerimientos	7
Requerimientos Funcionales.....	7
Requerimientos No Funcionales	7
Los actores y los casos de uso.....	8
Diseño	9
Primer diseño:.....	9
Segundo diseño:.....	10
Tercer diseño:	11
Diagrama UML	12
Implementación (Codificación).....	13
Barra superior y funcionalidades (cerrar, minimizar, fecha y hora del sistema)	14
Formulario de registro inicial	15
Área de agregado de tareas y funcionalidades.....	16

Área de listbox (estados de las tareas)	18
Área de exportado a excel	22
Pruebas	23

Introducción

En este proyecto se documentará el desarrollo de nuestra aplicación de escritorio, la cual es una “**Agenda de tareas**” que fue desarrollada en el lenguaje de programación **C#**, en base a los criterios y necesidades del cliente.

Se explicarán cada una de las fases de nuestro proyecto con el fin de justificar las decisiones tomadas por el equipo de trabajo, así mismo como el cumplimiento de lo solicitado por el cliente y el funcionamiento de la aplicación.

Las fases del **ciclo de vida del software** se mencionarán a continuación y además cada fase estará documentada.

- Planificación
- Análisis de requerimientos
- Diseño
- Implementación (Codificación)
- Pruebas
- Implementación en producción
- Mantenimiento

EQUIPO DE DESARROLLO



Jesus Meza
App Designer



Edgar Segura
Developer

Planeación

En esta etapa se procede a elaborar un plan de proyecto el cual deberá de contemplar lo siguiente...

Comunicación con el cliente:

Con el objetivo para el desarrollo de esta aplicación en base a sus necesidades y objetivos. Además, el cliente hace mención de cada uno de sus requerimientos y necesidades que se deberán tomar muy en cuenta.

Se pasa a elaborar el plan del proyecto:

Se estimarán recursos como, la cantidad de personal para cada tarea, presupuesto, equipo tecnológico y tiempo estimado de duración. En este caso el personal encargado constará de 2 personas, el encargado del diseño e implementación y el encargado de la lógica y codificación.

Objetivo:

Se plantea la forma en la que se llevará a cabo el desarrollo para una mejor eficiencia en tiempo y en recursos. Además de que optará por utilizará “**Windows**” como sistema operativo para el desarrollo, con una estimación de 1 mes como tiempo límite de entrega final.

Metodología KANBAN

Se optó por utilizar la metodología “KANBAN” debido a que resultó más fácil la visualización del proyecto, es decir que el flujo y eficiencia de trabajo son mejores con el uso de la pizarra de tareas ya sea virtual o física, separando por “Tareas pendientes”, “Tareas en proceso (WIP)” y “Tareas finalizadas”.

Agenda de tareas

Legenda:

Según lo previsto

Riesgo bajo

Riesgo medio

Riesgo alto

Sin asignar

ISC

Jesus Alejandro Serrano Meza & Edgar Fabian Castañeda Segura

Fecha de inicio del proyecto:

24/09/2025

Incremento de despliegues:

1

Descripción del hito	Categoría	Agenda	Inicio	Fin
----------------------	-----------	--------	--------	-----

Financiado

Entrevistas con el cliente en persona

Riesgo alto

Jesus & Edgar

24/09/2025

1

Definir los objetivos del proyecto

Riesgo alto

Jesus & Edgar

26/09/2025

2

Estimar la actividad y presupuesto del proyecto, el cliente el personal que se dedicará al desarrollo.

Riesgo medio

Jesus & Edgar

30/09/2025

6

Análisis de requerimientos

Usar las funciones requeridas por el cliente

Riesgo alto

Jesus & Edgar

01/10/2025

1

Conocer los equipos y las características de los sistemas con los que el cliente cuenta

Riesgo medio

Jesus & Edgar

03/10/2025

2

Establecer prioridad entre requerimientos funcionales y no funcionales

Riesgo medio

Jesus & Edgar

06/10/2025

3

Definir acciones y roles de uso, respecto a la aplicación

Riesgo medio

Jesus & Edgar

07/10/2025

1

Diseño

Crear un prototipo visual

Riesgo alto

Jesus

09/10/2025

1

Crear el diagrama UML

Riesgo medio

Jesus

12/10/2025

1

Diseñar los módulos cliente, el cliente los backend y frontend, cliente de la aplicación móvil y web (ver video 2023)

Riesgo alto

Jesus

13/10/2025

6

Diseñar una interfaz de usuario para la exportación del reporte

Riesgo medio

Jesus

14/10/2025

1

Agrupar los iconos, tanto del aplicativo web como de la aplicación en la barra de tareas

Riesgo alto

Jesus

15/10/2025

1

Implementación (Calificación)

Configuración inicial en lenguaje C#

Riesgo alto

Edgar

17/10/2025

30

Separar la data necesaria y agregar funciones y métodos

Riesgo alto

Edgar

18/10/2025

5

Programar la exportación a excel con una plantilla

Riesgo medio

Edgar

20/10/2025

1

Verificación de bugs de manera simultánea con la programación del código

Riesgo alto

Edgar

21/10/2025

30

Análisis de datos

Los miembros del equipo de desarrollo ejecutan pruebas individuales

Riesgo alto

Jesus & Edgar

22/10/2025

12

Revisión y actualización de datos, revisión para cambios posteriores

Riesgo alto

Jesus & Edgar

23/10/2025

12

Implementación (By producción)

Implementación del proyecto "Sabor" o instalador que se usó para la distribución de la aplicación

Riesgo medio

Jesus & Edgar

24/10/2025

2

Se desarrolló el software para obtener los datos de los clientes y registrar al mismo funcionamiento

Riesgo medio

Jesus & Edgar

25/10/2025

1



Diagrama de Gantt

Análisis de requerimientos

Después de realizar la entrevista con el cliente, se mencionó lo que necesitaba y fue clasificado en base a su funcionalidad (funcionales, no funcionales), así como las funciones más importantes a petición del cliente tomando en cuenta sus necesidades y el personal que hará uso de la aplicación.

Requerimientos Funcionales

La aplicación deberá de contemplar las siguientes características y funcionalidades para satisfacer las necesidades del cliente.

Funciones que se contemplan de manera urgente:

- -Agregar tareas
- -Marcar tareas
- -Listar tareas
- -Eliminar tareas
- -Guardar tareas para generar un reporte de tareas en un archivo de Excel.

Requerimientos No Funcionales

Funciones que no se deben contemplar primero:

- -Diseño de la interfaz
- -Rendimiento
- -Optimización
- -Tamaño de la aplicación

Los actores y los casos de uso

Actores: Los actores serán los usuarios que tengan autorizado el acceso a la aplicación dentro de la organización, además de que la organización podrá optar por tener un usuario administrador el cual se encargará de dar la capacitación en el uso de la aplicación

Casos de uso: La aplicación podrá permitir realizar las siguientes tareas y usos...

- Registrar usuario y correo electrónico.
- Agregar tareas con nombre y descripción.
- Eliminar tareas.
- Mostrar las tareas en una lista en la pantalla.
- Definir los estados de las tareas.
- Crear un reporte y exportarlo a un archivo de Excel.

Diseño

En esta etapa se hicieron varios diseños dependiendo de las funciones que se fueron agregando en el desarrollo de esta aplicación.

Primer diseño:

Este diseño simple se usó para la base de la aplicación e ir probando su funcionamiento, funciona más como un prototipo que como una versión completa, ayudó a las pruebas iniciales y posteriormente se fue mejorando la interfaz.



The screenshot shows a web application interface titled "Form1" in the top right corner. The interface is divided into several sections:

- Form Fields:** On the left side, there are three input fields labeled "Nombre", "Titulo de la tarea", and "Descripcion". Below these fields is a button labeled "Agregar Tarea".
- Form Fields:** On the right side, there are two input fields labeled "Correo" and "Registrar Usuario".
- List of tasks:** In the center, there is a large rectangular box labeled "Lista de tareas registradas".
- Buttons:** At the bottom left, there are two buttons labeled "Cambiar Estado" and "Exportar a Excel".

Segundo diseño:

El diseño se fue mejorando y se empezó con una interfaz un poco más desarrollada con las funciones que se le fueron agregando a la aplicación, como nuevos añadidos se agregó fecha, hora y un menú minimalista, este segundo diseño se usó como base para las futuras versiones y se mejoraría.

The screenshot displays a web application for task management. The interface is structured as follows:

- Top Navigation Bar (Gray):**
 - Logo: A stylized 'A' with a globe inside, followed by 'ISTC'.
 - User Information: Fields for 'Nombre' and 'Correo'.
 - Registration: A button labeled 'Registro del usuario'.
 - Task Title: A field labeled 'Titulo de la tarea'.
 - Description: A field labeled 'Descripcion'.
 - Action: A button labeled 'Agregar Tarea'.
- Main Content Area (Yellow):**
 - Title: 'Lista de tareas registradas'.
 - Columns:
 - Lista de tareas por realizar:** A large empty box for tasks to be done.
 - Tareas en proceso:** A large empty box for tasks in progress.
 - Tareas finalizadas:** A large empty box for completed tasks.
 - Actions: Two buttons at the bottom left, 'Cambiar Estado' and 'Exportar a Excel'.
- Bottom Status Bar (Gray):**
 - Time and Date: '05:45:59 p. m.' and 'miércoles, 17 de septiembre de 2025'.
 - Navigation: A minimal menu icon on the left and a close icon on the right.

Tercer diseño:

Posiblemente el diseño final de la aplicación, e hicieron ciertos cambios que el cliente solicito y se agregaron las funciones solicitadas por él, se mejoró un poco el menú y se cambiaron los botones de lugar para una mejora visual e intuitiva.

Nombre

Correo

Registro del usuario

Título de la tarea

Descripción

Agregar Tarea

Buscar Tarea

Eliminar Tarea

Listar Tareas

Lista de tareas registradas

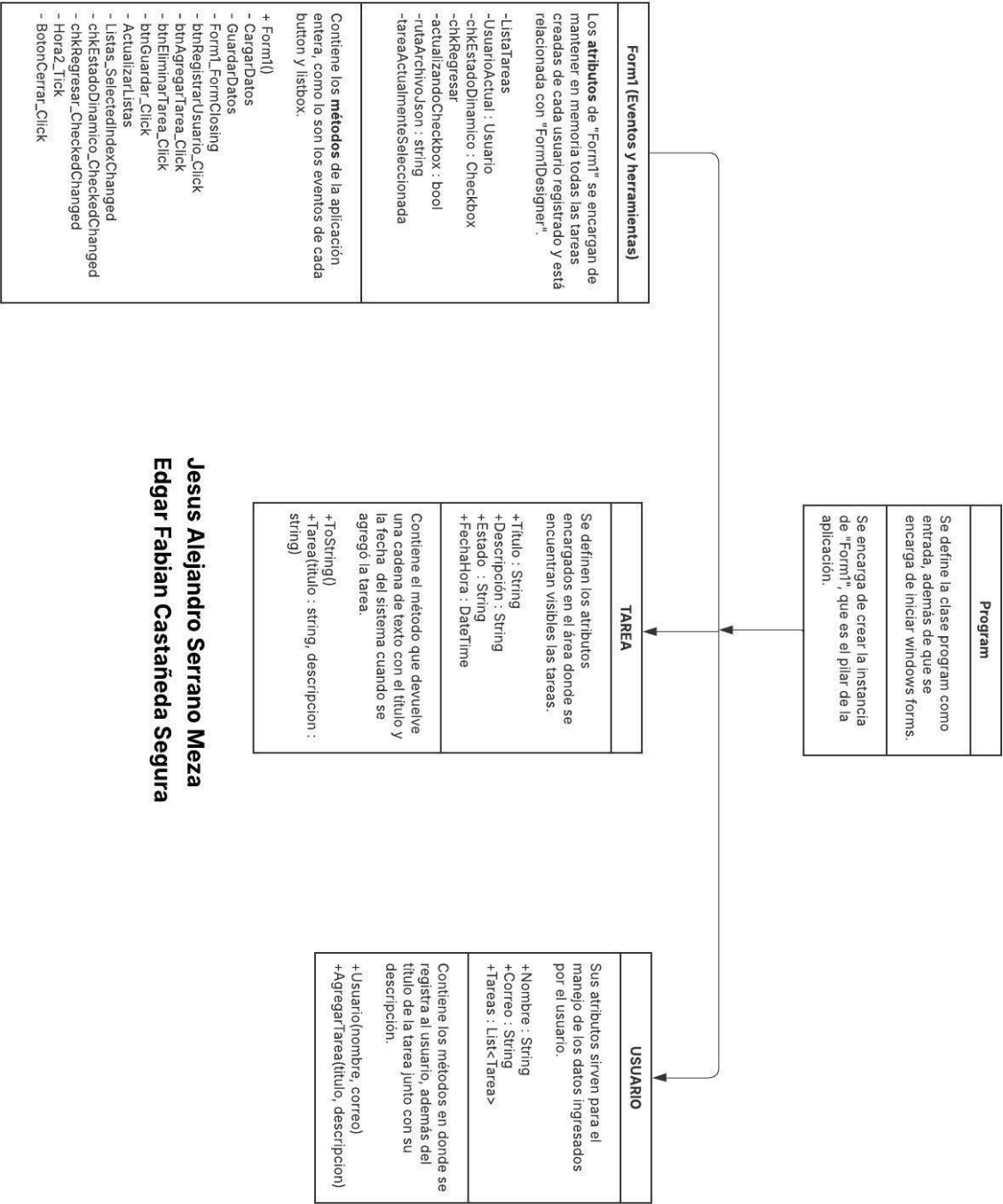
Tareas pendientes	Tareas en proceso	Tareas finalizadas
<div></div>	<div></div>	<div></div>

Guardar proyecto

Exportar a Excel

Diagrama UML

Posterior al diseño de la parte visual de la interfaz de usuario deseada, se creó el diagrama UML el cual contiene la división de clases de la aplicación, además de sus atributos y métodos con una breve explicación de que se encarga cada clase.



Jesus Alejandro Serrano Meza
Edgar Fabian Castañeda Segura

Implementación (Codificación).

De manera posterior al diseño, en la fase de la codificación se optó por utilizar el lenguaje de programación “C#” utilizando el IDE “Visual Studio 2022” con el entorno gráfico de Windows forms y de manera simultánea con los códigos principales.

El proyecto cuenta con las siguientes clases...

- ⇒ Clase Program.cs: Esta clase main o principal, que busca la aplicación al ser ejecutada como punto de partida, además de que se encarga de decidir que va a mostrar primero, así como información cargada.
- ⇒ Clase Tarea.cs: Esta clase se encarga de la información de cada tarea, es decir, es responsable de la información que el usuario ingresa como, por ejemplo, “El título, la descripción, el estado (pendiente, en proceso y finalizado), además de la fecha y hora en la que cada tarea es agregada.
- ⇒ Clase Usuario.cs: Esta clase es la que se encarga de guardar la información en tiempo de ejecución de todo lo que el usuario ingrese, desde el nombre de usuario, correo, así como la lista completa de tareas.
- ⇒ Clase Form1.cs: Esta clase se encarga de gestionar las tareas, es decir, de mostrar al usuario las tareas que están pendientes, en proceso y finalizadas. Además de que se encarga de los eventos o funciones que cada botón para que mediante las herramientas de visual studio, se puedan asociar eventos a cada botón, cada listbox y cada textbox. También se encarga de la lógica detrás de los check box dinámicos y la exportación del registro de tareas a un archivo de Excel mediante una plantilla.

Barra superior y funcionalidades (cerrar, minimizar, fecha y hora del sistema)

A continuación, se mostrarán los códigos de cada evento, cada botón y cada funcionalidad de la aplicación separado por áreas, se utilizó el IDE “Visual Studio Code” para una mejor apreciación de los códigos.

Fecha y hora del sistema, en este apartado de la clase form1.cs se utilizan estas líneas de código para que se encarguen de utilizar la hora y fecha del sistema mediante las etiquetas previamente elegidas en diseño, para ser mostradas durante la ejecución de la aplicación.

```
1 // Este método se ejecuta cada segundo para actualizar los labels de hora y fecha.
2 private void Hora2_Tick(object sender, EventArgs e)
3 {
4     lblhora.Text = DateTime.Now.ToString("HH:mm:ss");
5     lblfecha.Text = DateTime.Now.ToLongDateString();
6 }
```

Botón minimizar, este botón está encargado de minimizar la aplicación al hacer click en él, utilizando el evento “pictureBox1_click” el cual está asociado con el recurso gráfico en forma de botón de minimizar “picture box1”.

```
1 // Evento que se dispara al hacer clic en el botón de minimizar.
2 private void pictureBox1_Click(object sender, EventArgs e)
3 {
4     this.WindowState = FormWindowState.Minimized;
5 }
```

Botón cerrar, mediante “Application.Exit()” este botón se encarga de cerrar y finalizar la ejecución de la aplicación mediante el evento “BotonCerrar_Click”.

```
1 // Evento que se dispara al hacer clic en el botón de cerrar.
2 private void BotonCerrar_Click(object sender, EventArgs e)
3 {
4     Application.Exit();
5 }
```

Formulario de registro inicial

Textbox “Nombre y correo”, son los encargados de registrar los datos de registro de cada usuario.

```

0 references
1  private System.Windows.Forms.TextBox txtNombre;
0 references
2  private System.Windows.Forms.TextBox txtCorreo;

```

En este apartado fue necesario incluir la siguiente línea de código en el Form1.Designer.cs, para que los controles (Nombre y Correo) tengan un límite específico de 256 caracteres al ingresar datos.

```

1  this.txtNombre.MaxLength = 256;

```

Están asociados al evento del botón registrar usuario de esta manera: “btnRegistrarUsuario_click” el cual colecta la información de ambos campos de registro para crear una nueva instancia, es decir un nuevo usuario que podrá tener acceso a las funciones completas de la aplicación, además de que se asegura que no existan usuarios repetidos o que el usuario llene todos los campos y no deje vacío ninguno.

```

1  private void btnRegistrarUsuario_Click(object sender, EventArgs e)
2  {
3      string nombre = txtNombre.Text.Trim();
4      string correo = txtCorreo.Text.Trim();
5      if (string.IsNullOrEmpty(nombre) || string.IsNullOrEmpty(correo))
6      {
7          MessageBox.Show("Completa nombre y correo.");
8          return;
9      }
10     if (usuarioActual != null)
11     {
12         var resultado = MessageBox.Show("Ya existe un usuario. ¿Deseas reemplazarlo? Se perderán las tareas anteriores.",
13             "Confirmar", MessageBoxButtons.YesNo);
14         if (resultado == DialogResult.No) return;
15     }
16     usuarioActual = new Usuario(nombre, correo);
17     MessageBox.Show("Usuario registrado correctamente.");
18     ActualizarListas();
19 }

```

Área de agregado de tareas y funcionalidades

Textbox “Título y descripción de las tareas”, son dos controles o botones los cuales están encargados guardar la información del título y la descripción de la tarea que el usuario prefiera.

```

1 private System.Windows.Forms.TextBox txtDescripcion;
2 private System.Windows.Forms.TextBox txtTitulo;

```

Cabe **destacar** que fue necesario colocar un límite de 256 caracteres a los controles de cada apartado de ingreso de información (Título, Descripción), para un ahorro de memoria.

```

1 this.txtNombre.MaxLength = 256;

```

Botón “agregar tarea”, está asociado al evento llamado “btnAgregarTarea_Click” el cual se encarga de checar que no exista un usuario repetido, además de que los campos no estén vacíos y también la tarea creada se le asignará el estado inicial de “pendiente”. También se encarga de mostrar tanto su título, su descripción, su estado y su fecha de creación.

```

1 private void btnAgregarTarea_Click(object sender, EventArgs e)
2 {
3     // ... (validaciones previas)
4     var nueva = new Tarea
5     {
6         Titulo = txtTitulo.Text.Trim(),
7         Descripcion = txtDescripcion.Text.Trim(),
8         Estado = "Pendiente",
9         FechaHora = DateTime.Now
10    };
11    usuarioActual.Tareas.Add(nueva);
12    ActualizarListas();
13    LimpiarCampos();
14 }

```

Botón “buscar tarea”, se asocia al evento llamado “btnBuscarTarea_Click” el cual simplemente mediante la información ingresada en el textbox de título, poder ubicar tareas, así

como su estado en las 3 listas de tareas. En caso de que no encuentre la tarea escrita, saltará un mensaje que dice “No se encontró la tarea”.

```

1 private void btnBuscarTarea_Click(object sender, EventArgs e)
2 {
3     if (usuarioActual == null) return;
4     string titulo = txtTitulo.Text.Trim();
5     var tarea = usuarioActual.Tareas.FirstOrDefault(t => t.Titulo.Equals(titulo, StringComparison.OrdinalIgnoreCase));
6     if (tarea != null)
7         MessageBox.Show($"Encontrada: {tarea.Titulo}\nEstado: {tarea.Estado}\n{tarea.Descripcion}");
8     else
9         MessageBox.Show("No se encontró la tarea.");
10 }

```

Botón “eliminar tarea”, este botón se asocia al evento “btnEliminarTarea_Click” el cual está encargado de que al momento de seleccionar alguna tarea y hacer click en el botón, esta se elimine en ese momento y en caso de no seleccionar ninguna tarea y hacer click sobre el botón, nos salte un mensaje de “Selecciona una tarea para eliminar”. Además de que actualiza las listas para que el cambio se mantenga.

```

1 private void btnEliminarTarea_Click(object sender, EventArgs e)
2 {
3     if (tareaActualmenteSeleccionada != null)
4     {
5         usuarioActual.Tareas.Remove(tareaActualmenteSeleccionada);
6         ActualizarListas();
7     }
8     else
9     {
10        MessageBox.Show("Selecciona una tarea para eliminar.");
11    }
12 }

```

Botón “listar tareas”, este botón se asocia al evento “btnListarTareas_Click” el cual hace que se ejecute el método de “Actualizar listas()” el cual prácticamente devuelve el total de tareas que hay registradas con el usuario actual con un mensaje de texto.

```

1 private void btnListarTareas_Click(object sender, EventArgs e)
2 {
3     if (usuarioActual == null) { MessageBox.Show("No hay un usuario registrado."); return; }
4     ActualizarListas();
5     MessageBox.Show($"Total de tareas: {usuarioActual.Tareas.Count}");
6 }

```

Área de listbox (estados de las tareas)

En este apartado se encuentran las tres listbox (pendientes, en proceso y finalizadas), en donde muestran en pantalla el título de la tarea, la descripción y fecha/hora. Además de que se incorpora una scrollbar para que se pueda visualizar todo sin perder detalle.

```
0 references
1 private System.Windows.Forms.ListBox lstTareas; //Tareas pendientes
0 references
2 private System.Windows.Forms.ListBox Proceso; //Tareas en proceso
0 references
3 private System.Windows.Forms.ListBox Finalizadas; //Tareas finalizadas
```

El funcionamiento de las tres listbox se resume en lo siguiente... “ActualizarListas()” es el encargado de sincronizar el funcionamiento y las tareas con sus estados que el usuario actual haya ingresado en cualquiera de las listas siguientes según corresponda su tipo de estado “lstTareasPendientes, lstTareasEnProceso y lstTareasFinalizadas”.

```
1 private void ActualizarListas()
2 {
3     // ... (validación de usuario)
4     lstTareasPendientes.Items.Clear();
5     lstTareasEnProceso.Items.Clear();
6     lstTareasFinalizadas.Items.Clear();
7
8     foreach (var tarea in usuarioActual.Tareas.OrderBy(t => t.FechaHora))
9     {
10         if (tarea.Estado == "Pendiente")
11         {
12             lstTareasPendientes.Items.Add(tarea);
13         }
14         else if (tarea.Estado == "EnProceso")
15         {
16             lstTareasEnProceso.Items.Add(tarea);
17         }
18         else if (tarea.Estado == "Finalizada")
19         {
20             lstTareasFinalizadas.Items.Add(tarea);
21         }
22     }
23 }
```

“**Checkbox dinámicos**”, se encargan de sustituir a los botones “cambiar estado” de los primeros prototipos, su funcionamiento consta de que al seleccionar alguna tarea se mostrará un checkbox dependiendo del estado en la que se encuentre la tarea y posteriormente al hacer click en el checkbox, la tarea cambiará de estado automáticamente, así como regresar a un estado anterior.

```

1 // Se activa cuando se selecciona un ítem en CUALQUIER listBox.
2 private void lstTareas_SelectedIndexChanged(object sender, EventArgs e)
3 {
4     // ... (código para deseleccionar otras listas y encontrar la tarea)
5
6     if (tareaActualmenteSeleccionada != null)
7     {
8         // Muestra y posiciona los CheckBox al lado de la tarea seleccionada
9         PosicionarYConfigurarCheckboxes(listBoxActivo, indiceSeleccionado, tareaActualmenteSeleccionada);
10    }
11 }
12
13 // Se activa cuando el estado del CheckBox principal cambia (Lo marcas/desmarcas).
14 private void chkEstadoDinamico_CheckedChanged(object sender, EventArgs e)
15 {
16     if (actualizandoCheckbox || tareaActualmenteSeleccionada == null) return;
17
18     if (tareaActualmenteSeleccionada.Estado == "Pendiente" && chkEstadoDinamico.Checked)
19     {
20         tareaActualmenteSeleccionada.Estado = "EnProceso";
21     }
22     else if (tareaActualmenteSeleccionada.Estado == "EnProceso" && chkEstadoDinamico.Checked)
23     {
24         tareaActualmenteSeleccionada.Estado = "Finalizada";
25     }
26     // ... (lógica para desmarcar)
27
28     ActualizarListas(); // Refresca las listas para mostrar el cambio.
29 }

```

Modo “edición o actualización”, se incorpora una función que se activa al hacer doble click sobre una tarea que ya se encuentre en cualquiera de las tres listas de tareas, para tener la posibilidad de “editarla o actualizarla”, tanto su título como su descripción. Al activarse el modo edición (bool=true), de lado izquierdo cambia de color para que el usuario sepa que puede comenzar a editar el nombre y la descripción, además de que el botón “Guardar proyecto” funciona para un doble propósito.

```

1  /// <summary>
2  /// Se activa al hacer doble clic en una tarea. Prepara el formulario para el modo de edición.
3  /// </summary>
4  private void ListaTareas_DoubleClick(object sender, EventArgs e)
5  {
6      if (tareaActualmenteSeleccionada != null)
7      {
8          enModoEdicion = true;
9          txtTitulo.Text = tareaActualmenteSeleccionada.Titulo;
10         txtDescripcion.Text = tareaActualmenteSeleccionada.Descripcion;
11         btnGuardar.Text = "Actualizar Tarea";
12         btnAgregarTarea.Enabled = false;
13         panel1.BackColor = System.Drawing.Color.LightSteelBlue; //aquí puedes cambiar el color
14     }
15 }

```

Finalmente, la opción de “**Cancelar edición**” (bool=false), devuelve la aplicación a sus funciones normales.

```

1  /// <summary>
2  /// Restablece la interfaz a su estado normal, saliendo del modo de edición.
3  /// </summary>
4  private void CancelarEdicion()
5  {
6      txtTitulo.Clear();
7      txtDescripcion.Clear();
8      btnGuardar.Text = "Guardar proyecto";
9      btnAgregarTarea.Enabled = true;
10     panel1.BackColor = System.Drawing.SystemColors.ScrollBar;
11     enModoEdicion = false;
12 }

```

Botón “Guardar proyecto”, asociado al evento “btnGuardar_Click, este botón está encargado de guardar el avance del proyecto actual del usuario, para evitar la pérdida de datos. Utiliza el guardado de datos mediante un archivo “Json”(JavaScript Object Notation) con el cual se puede decir que la aplicación cuenta con “persistencia” y se guarda en la misma ruta del ejecutable exe. Además, que se le incorporó una doble función, la cual realiza lo antes mencionado y además guarda los cambios de **edición** o **actualización** de las tareas, del “**modo edición**” de tareas al hacer doble click sobre una tarea, dando distintos mensajes dependiendo del proceso en curso seleccionado por el usuario.

```

5      /// </summary>
6      private void btnGuardar_Click(object sender, EventArgs e)
7      {
8          if (enModoEdicion)
9          {
10             if (tareaActualmenteSeleccionada != null)
11             {
12                 if (string.IsNullOrEmpty(txtTitulo.Text))
13                 {
14                     MessageBox.Show("El título no puede estar vacío.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
15                     return;
16                 }
17                 tareaActualmenteSeleccionada.Titulo = txtTitulo.Text.Trim();
18                 tareaActualmenteSeleccionada.Descripcion = txtDescripcion.Text.Trim();
19                 ActualizarListas();
20                 GuardarDatos();
21                 MessageBox.Show("¡Tarea actualizada y proyecto guardado!", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information);
22                 CancelarEdicion();
23             }
24         }
25         else
26         {
27             GuardarDatos();
28             MessageBox.Show("¡Proyecto guardado exitosamente!", "Guardado", MessageBoxButtons.OK, MessageBoxIcon.Information);
29         }
30     }
31

```

Cargar los datos al iniciar, también cuenta con la función que se encarga de que al iniciar la aplicación se carguen los datos guardados previamente en el archivo “Json” y que se visualicen automáticamente en cada inicio de la aplicación mediante “Form1_Load”.

```

1      private void Form1_Load(object sender, EventArgs e)
2      {
3          if (File.Exists(rutaArchivoJson))
4          {
5              string jsonString = File.ReadAllText(rutaArchivoJson);
6              usuarioActual = JsonSerializer.Deserialize<Usuario>(jsonString);
7              if (usuarioActual != null)
8              {
9                  ActualizarListas();
10             }
11         }
12     }

```

Finalmente se aclara que, en cuanto a esta base de datos simulada, es necesario saber que la aplicación está pensada para instalarse en cada equipo por separado, así cada miembro del equipo podrá hacer uso individual de la aplicación. Sin embargo, no es posible ser utilizada por múltiples usuarios por que el archivo “Json” se sobrescribe cada vez.

Área de exportado a excel

Botón de “Exportar a Excel”, que gracias a la librería “ClosedXML” se puede utilizar el evento “btnExportar_click” para que sea posible generar un reporte y exportarlo a un archivo xlsx de Excel, en esta ocasión mediante una plantilla previamente diseñada en la cual se agregan solamente los datos sobre las celdas asignadas que son las siguientes... “Título, Descripción, Estado y Fecha/Hora”.

```
1 private void btnExportar_Click(object sender, EventArgs e)
2 {
3     // ... (validación de usuario)
4     string rutaPlantilla = Path.Combine(Application.StartupPath, "Plantilla.xlsx");
5     string rutaSalida = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "Reporte de Tareas.xlsx");
6
7     // Carga la plantilla de Excel.
8     var workbook = new XLWorkbook(rutaPlantilla);
9     var worksheet = workbook.Worksheet(1); // Accede a la primera hoja.
10
11     int filaActual = 2; // Comienza a escribir en la segunda fila.
12     foreach (var tarea in usuarioActual.Tareas)
13     {
14         worksheet.Cell(filaActual, 1).Value = tarea.Titulo;
15         worksheet.Cell(filaActual, 2).Value = tarea.Descripcion;
16         worksheet.Cell(filaActual, 3).Value = tarea.Estado;
17         worksheet.Cell(filaActual, 4).Value = tarea.FechaHora;
18         filaActual++;
19     }
20
21     // Guarda el nuevo archivo en el escritorio.
22     workbook.SaveAs(rutaSalida);
23     MessageBox.Show("Reporte exportado a Excel en tu escritorio.");
24 }
```

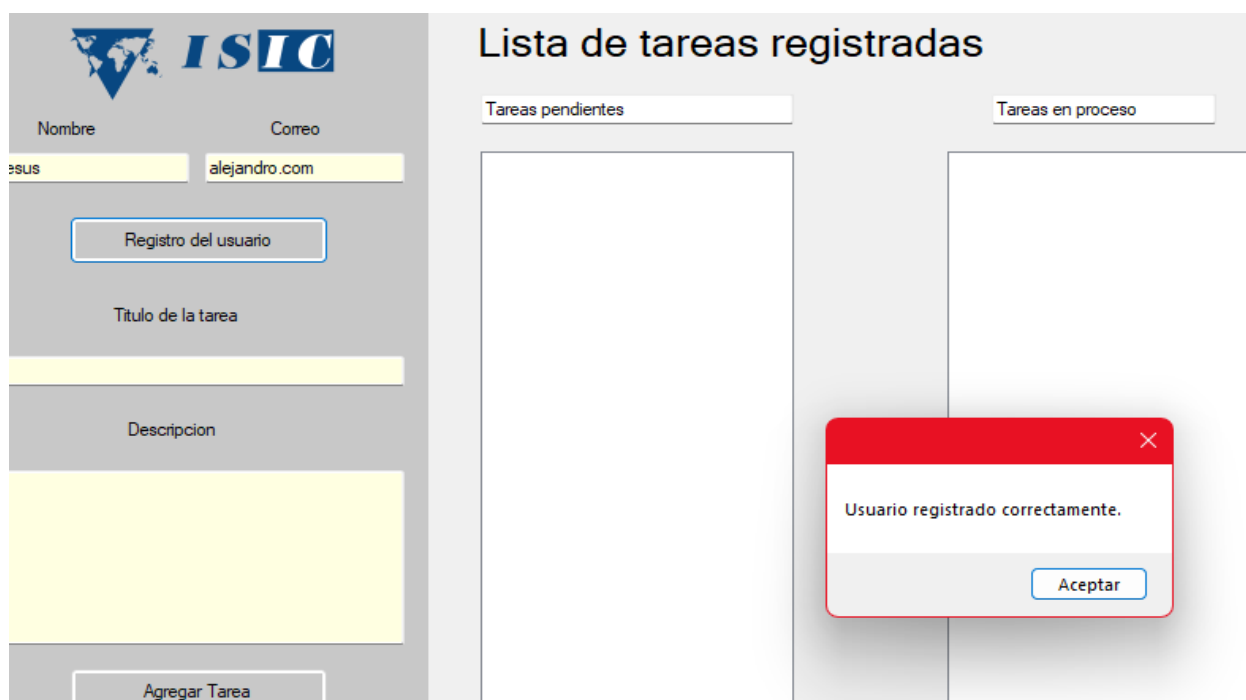
Pruebas

-Se realizaron las pruebas del funcionamiento del programa para testear su correcta aplicación, como primer paso se testeo el formulario de registro que es obligatorio para el uso del programa.



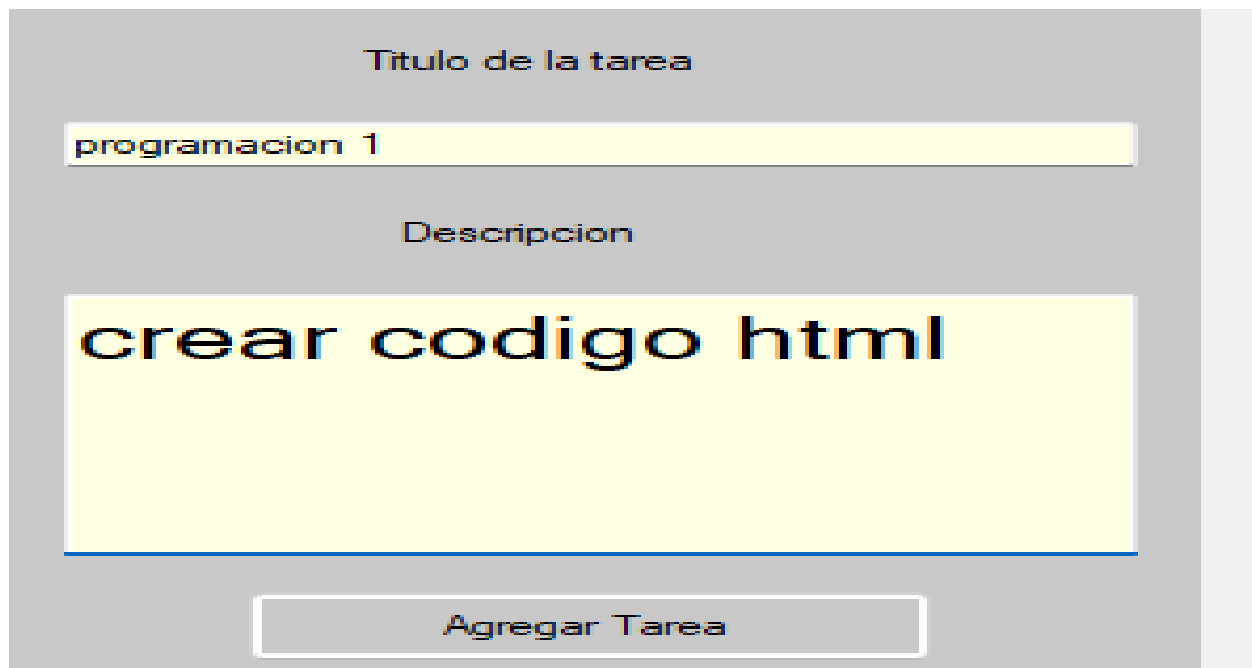
The screenshot shows a registration form for a system. At the top, there is a logo consisting of a blue triangle with a white world map inside, followed by the text "ISIC" in a bold, blue, serif font. Below the logo, there are two input fields. The first is labeled "Nombre" and contains the text "jesus". The second is labeled "Correo" and contains the text "alejandro.com". Below these fields is a button labeled "Registro del usuario".

-El resultado esperado de este formulario nos debería mostrar una pequeña ventana mencionando que el usuario se registró correctamente y nos debería dejar utilizar el programa.



The screenshot shows the main interface of the program. On the left, there is a sidebar with the same "ISIC" logo at the top. Below the logo, there are input fields for "Nombre" (containing "jesus") and "Correo" (containing "alejandro.com"), and a "Registro del usuario" button. Below this, there are fields for "Titulo de la tarea" and "Descripcion", and an "Agregar Tarea" button. On the right, there is a section titled "Lista de tareas registradas". It has two tabs: "Tareas pendientes" and "Tareas en proceso". Below the tabs are two empty rectangular boxes. A red dialog box with a white border and a close button (X) is overlaid on the right side. It contains the text "Usuario registrado correctamente." and an "Aceptar" button.

-**Segunda prueba:** A continuación, se probó el apartado para agregar las tareas con su título y descripción, así como el botón de “agregar tarea”, en teoría este apartado debe de funcionar correctamente y agregar la tarea con su descripción en nuestra tabla de tareas.



Formulario para agregar una tarea. El formulario tiene un fondo gris y una barra lateral gris a la derecha. En el centro, hay un campo de texto con el título "Titulo de la tarea" y un campo de texto con la descripción "Descripcion". El campo de descripción contiene el texto "crear codigo html". Debajo de los campos, hay un botón con el texto "Agregar Tarea".

Una vez en la tabla nos debe mostrar los datos de la tarea, así como el botón para iniciar esta misma.

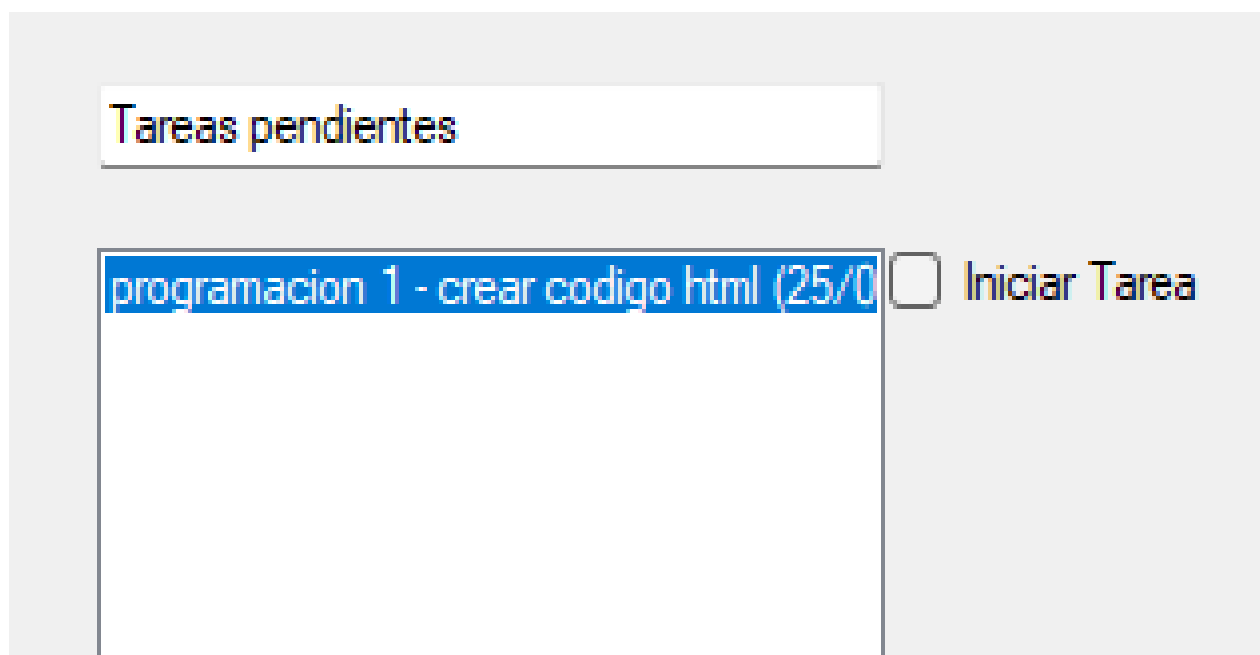


Tabla de tareas pendientes. La tabla tiene un encabezado "Tareas pendientes" y una fila de datos. La fila de datos muestra el título "programacion 1 - crear codigo html (25/0)", un botón "Iniciar Tarea" y un campo de texto vacío.

Tareas pendientes		
programacion 1 - crear codigo html (25/0)	<input type="checkbox"/> Iniciar Tarea	

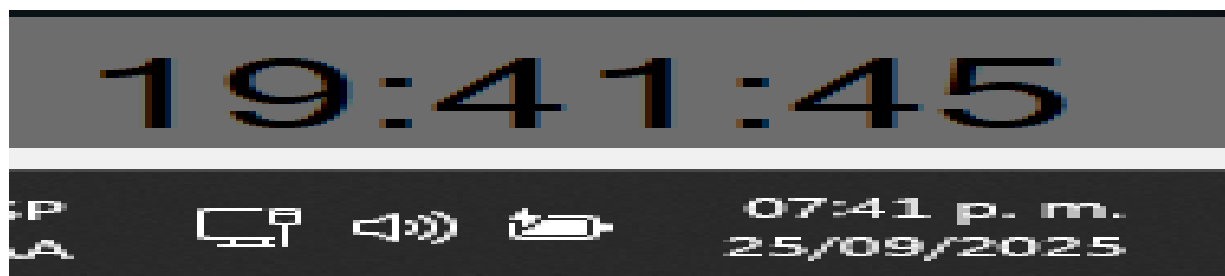
-Tercera prueba: Una vez que nos permita iniciar las tareas estas se agregan al apartado de en proceso, donde nos mostrará 2 opciones las cuales son: Finalizar tarea y regresar a pendiente

programacion 1 - crear codigo html (25/09)	<input type="checkbox"/> Finalizar Tarea
	<input type="checkbox"/> Regresar a Pendiente

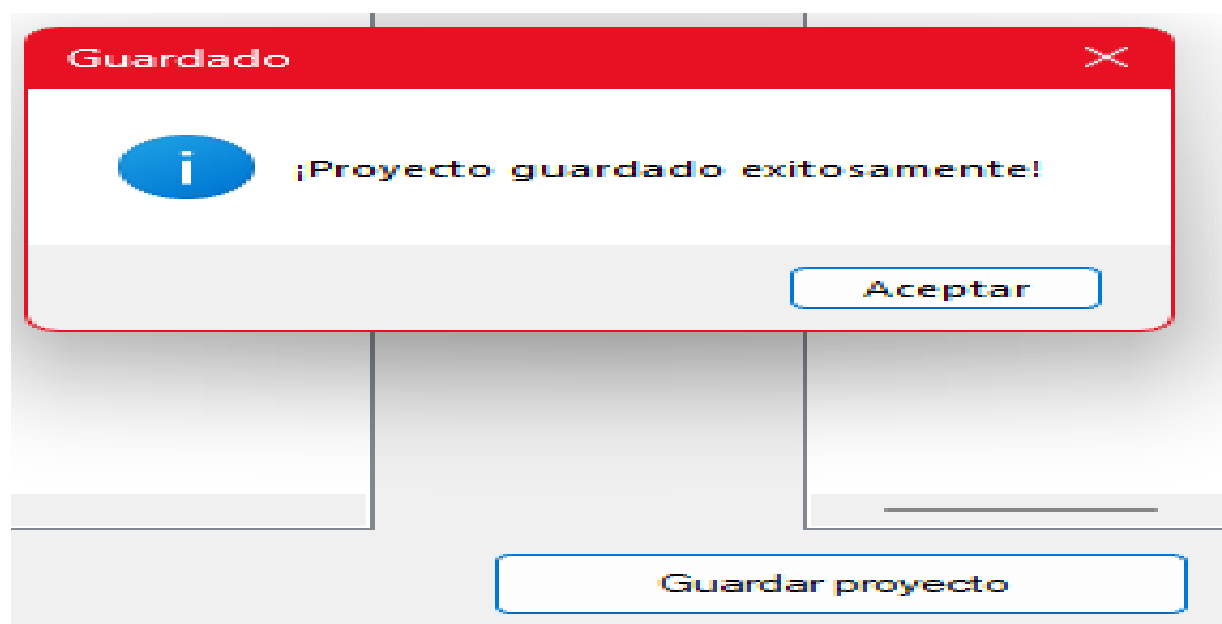
Estas opciones fueron a petición del cliente que su uso depende del propio usuario en base a sus necesidades. Al seleccionar finalizar tarea esta se enviará al último apartado en el cual tendremos otra opción que funciona para regresar la tarea a los otros 2 apartados dependiendo de la necesidad.

programacion 1 - crear codigo html (25/09)	<input checked="" type="checkbox"/> Reabrir Tarea
--	---

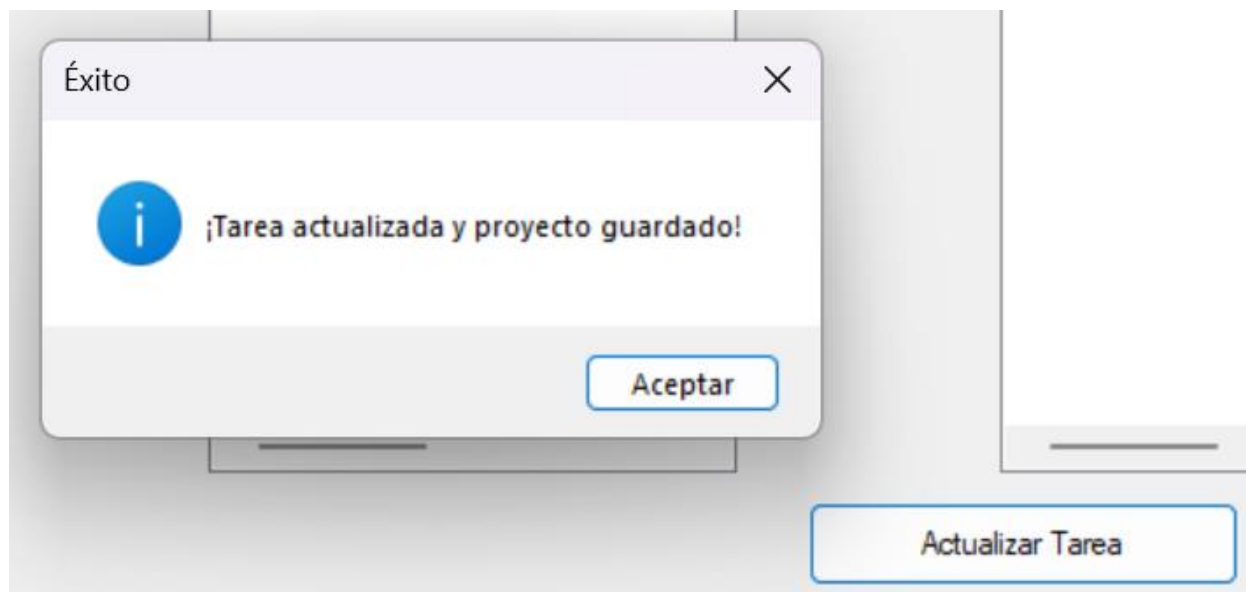
-**Cuarta prueba:** Se probó el que el reloj en la aplicación coincidiera con nuestra hora de escritorio



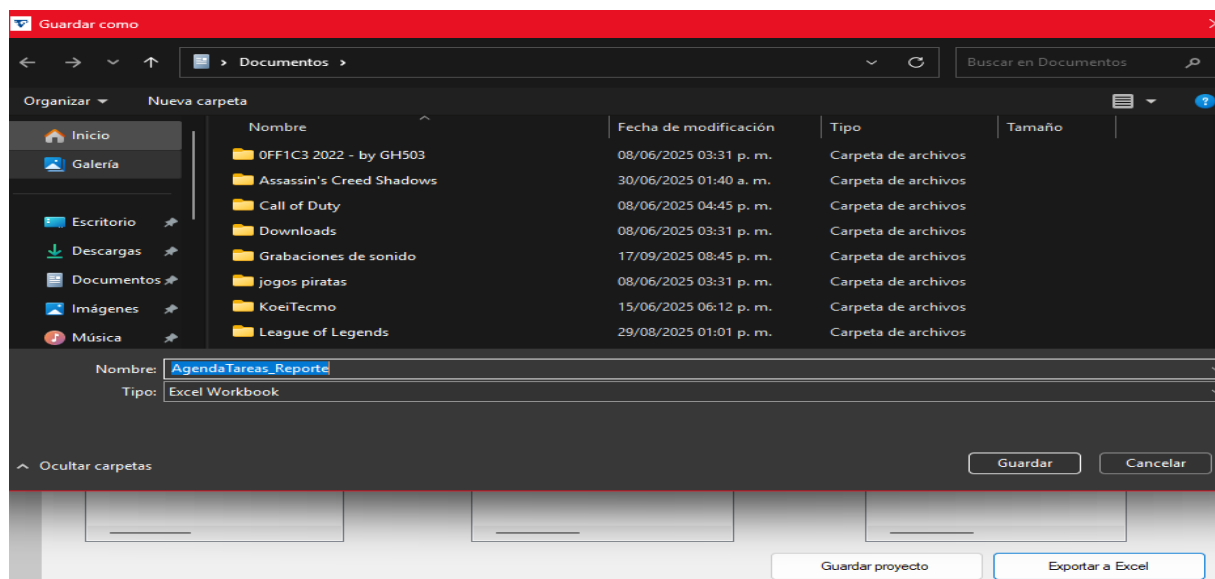
Se probó que el botón de “Guardar proyecto” funcione correctamente. Este botón hace que no se eliminen los avances del proyecto del usuario registrado en ese momento, la función crea un archivo Json, el cual simula una base de datos, en donde guarda todas las tareas que el usuario escriba, para que al momento de cerrar la aplicación no se elimine nada y pueda seguir editando el proyecto.



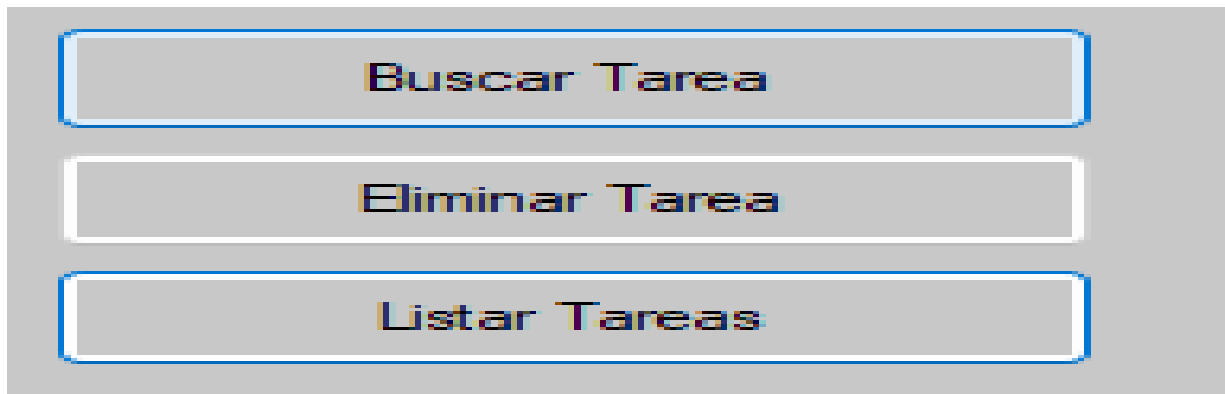
Se probó que la segunda funcionalidad del botón funcione correctamente, esto al terminar de editar cualquier tarea, para salvar los cambios tanto de la edición de la tarea, como de todo el proyecto completo. Se alcanza a apreciar que el botón cambia su nombre solamente cuando está en **modo edición**.



Se probó el botón “**Exportar a Excel**” el cual tiene como función generar un reporte de las tareas, fecha y hora, además de su descripción y estado, todo directo a un archivo.xlsx de Excel, en este caso a una plantilla prediseñada a petición del cliente.



-Para finalizar las pruebas se comprobaron los siguientes botones y se probó que estos realizaban perfectamente sus funciones individuales, por lo cual se finalizaron las pruebas.



Observaciones:

-Se encontró un bug que impedía regresar a los apartados “tareas pendientes y en proceso” una vez que la tarea estuviera como “Finalizada” por lo cual se tuvo que corregir.

-Se descubrió que la aplicación no tenía límite de caracteres para la información que el usuario ingrese, por lo cual se tuvo que corregir.

Github del proyecto

<https://github.com/leyendacodm95/Agenda-de-tareas>