Juan Fernando Giraldo, David Bolívar

Reto 5, Solución 1

Contexto problemático

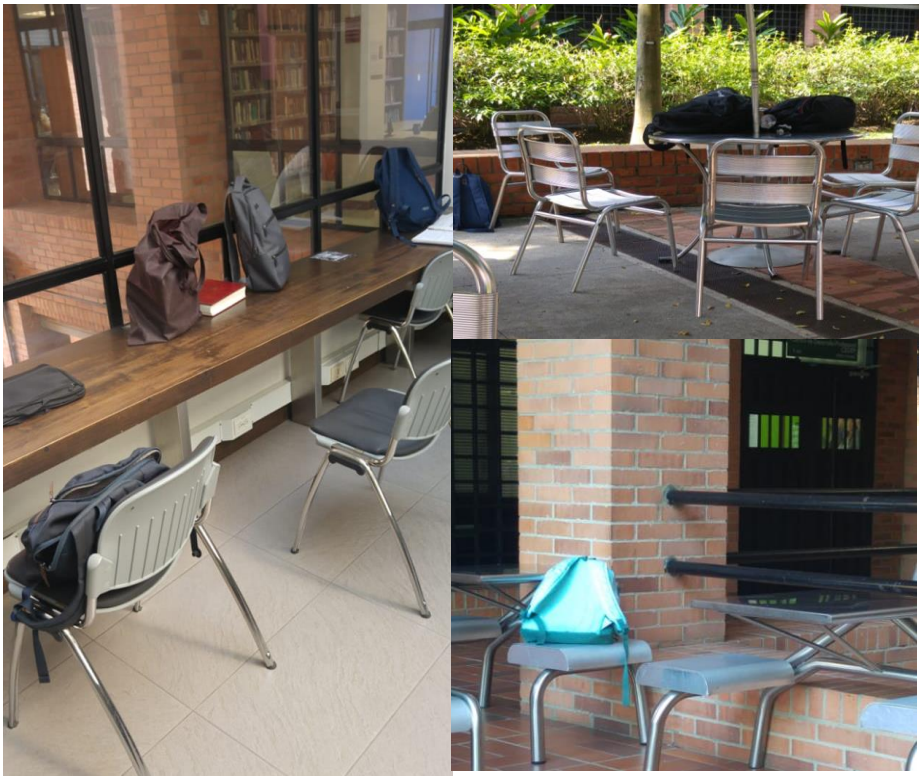
La universidad Icesi se ha visto afectada por situaciones de inseguridad en varias ocasiones. Específicamente, debido a aquellos que son amigos de lo ajeno. Las cafeterías han sido los espacios en donde más se han presentado situaciones de este tipo, ya que allí es donde los estudiantes han denunciado las pérdidas de sus pertenencias. Esto también se ha presentado en las diferentes oficinas, los baños cercanos a los escenarios deportivos, y en áreas de estudio en general. Ante esto, el personal de seguridad decidió tomar medidas al respecto, entre los cuales se destacan la vigilancia en las cafeterías durante la hora de almuerzo, al igual que el incremento de cámaras alrededor del campus universitario. ¿Estas medidas han sido suficientes para evitar los robos dentro del campus universitario? podría o debería la universidad llevar a cabo acciones diferentes frente a esta situación? (¿mejoras o cambios radicales?).

Desarrollo de la solución

1. **Identificación del problema**
   Teniendo en cuenta la situación planteada anteriormente y sabiendo que una de las medidas más eficaces implementadas por la universidad para prevenir los robos es el patrullaje continuo de guardas de seguridad, los cuales identifican pertenencias que han sido dejadas sin supervisión para posteriormente hacer un reporte de estas y guardarlas para que su dueño pueda recuperarlas, se decidió dar solución al problema del registro los objetos recuperados por los guardas.

   - **Identificación de requerimientos:**
     1. Registrar objeto encontrado por el guarda, y sus respectivas características.
     2. Persistencia de los datos registrados.
     3. Consistencia en la escritura de los datos. Se requiere sean escritos en un formato claro y legible para uso posterior del programa
     4. Oportunismo. Se requiere acceso a los datos (Lectura de datos) desde el programa.
     5. Búsqueda de objetos en el programa.
   - **Definición del problema**
     La universidad requiere un sistema para el registro y reporte de los artículos recuperado por los guardas de seguridad en las diferentes zonas de la universidad.

2. **Recopilación de información**
   Se recopilo información sobre como se maneja el registro de los artículos recuperados por los guardas de seguridad y se encontró que solo se maneja un sistema de registro físico el cual se tiene que hacer cada vez que se encuentra un artículo sin supervisión en la universidad, esto dificulta el proceso de hacer reportes y ralentiza el patrullaje.
   Para hacer el registro es necesario contar con la siguiente información:
   - Articulo encontrado
   - Descripción física del articulo
   - Lugar donde se encontró el articulo
   - Fecha cuando se encontró el articulo
   - Hora cuando se encontró el articulo
   - Comentarios adicionales del guarda



Link video de hurtos: http://www.youtube.com/watch?v=uLAolAaoZ9c

Pruebas de recolección de información

3. **Búsqueda de soluciones creativas**
   Se utilizo la técnica de lluvia de ideas para generar posibles soluciones el problema planteado anteriormente. Las ideas se clasificaron en tres categorías: Inicio de sesión para los guardas, registro de artículos encontrados y reportes de la información recopilada para posterior análisis. Las ideas generadas fueron las siguientes:

   -Inicio de sesión:
   - Crear una ventana de registros para que el usuario se pueda registrar y posteriormente iniciar sesión.
   - Dejar que cualquier persona inicie en la aplicación.
   - Vincular el log-in con una base de datos previa para así solo dejar que inicien sesión lo que están registrados.

   -Registro de artículos:
   - Utilizar un campo de texto donde el guarda puede hacer el informe a modo de párrafo
   - Utilizar múltiples campos de texto para separar las características
   - Guardar los reportes como texto plano en un archivo .txt

   -Reporte de artículos:
   - Separar los datos de cada registro del archivo de texto y mostrarlos en un DataGridView.
   - Mostrar los registros en un Textbox en forma de lista y separar cada dato con algún carácter o espacio.
   - Mostrar los registros en los controles donde se registra la información (textbox, combobox, etc) y agregar botones de navegación (adelante y atrás) para poder acceder a todos los registros.
   - Utilizar el lenguaje nativo de consulta LinQ para facilitar el reporte de información.

4. **Transición de las ideas a los diseños preliminares**

   Teniendo en cuenta los diferentes aspectos del programa y quien es el usuario final, se evaluaron las ideas obtenidas en la búsqueda de soluciones creativas.

   Para el inicio de sesión, se tuvo en cuenta que los guardas son el usuario final. Estos serán los que durante su patrullaje usarán la aplicación en caso de encontrar un objeto abandonado por una persona. Sabiendo esto, la idea de que cualquier

usuario tenga acceso a esta no es para nada ideal. Por lo que esta idea será descartada.

Para el registro de artículos se descartará la primera idea. Esta idea no solo genera inconsistencias y ambigüedades para un posterior acceso, sino que es poco adecuado para un uso rápido y sencillo. El usuario no se toma el detalle suficiente al momento de escribir párrafos, por lo que ciertos detalles necesarios no estarían integrados. No cumple uno de los requerimientos, por ende, se descarta de forma inmediata.

Para el reporte de artículos se descarta la segunda opción, pues puede generar ambigüedad al momento en que el usuario este haciendo la lectura. Seria dificultoso para el usuario final y por ende se descarta.

5. **Evaluación y selección de la mejor solución**

Criterios:

- Implementación de la solución: De 1 a 5 determina cuan sencilla es la implementación, 5 siendo muy fácil y 1 siendo muy difícil.
- Sencillez de la solución: De 1 a 5 determina cuan simple y sencillo es para el usuario final, 5 siendo muy fácil y 1 siendo muy difícil.
- Eficacia de la solución: De 1 a 5 determina cuan eficaz es la solución al momento de solucionar el problema y cumplir con los requerimientos.

Inicio de sesión				
Criterio **Propuesta**	**Implementación de la solución**	**Sencillez de la solución**	**Eficacia de la solución**	**Totales**

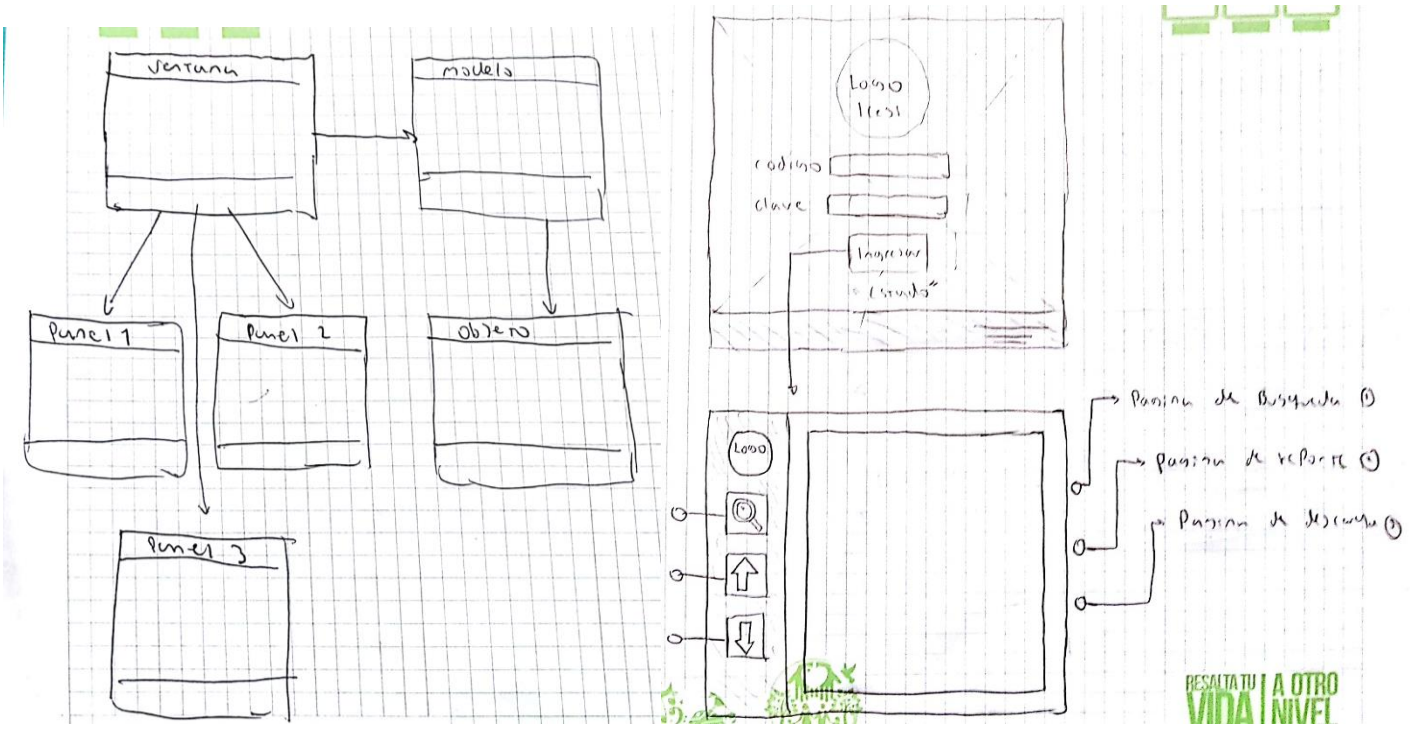
Crear una ventana de registros para que el usuario se pueda registrar y posteriormente iniciar sesión.	**4**	**3**	**5**	**12**
Vincular el log-in con una base de datos previa para así solo dejar que inicien sesión lo que están registrados.	**5**	**5**	**5**	**15**

Registro de artículos				
Criterio **Propuesta**	**Implementación de la solución**	**Sencillez de la solución**	**Eficacia de la solución**	**Totales**
Utilizar múltiples campos de texto para separar las características	**4**	**4**	**5**	**13**
Guardar los reportes como texto plano en un archivo .txt	**4**	**4**	**5**	**13**

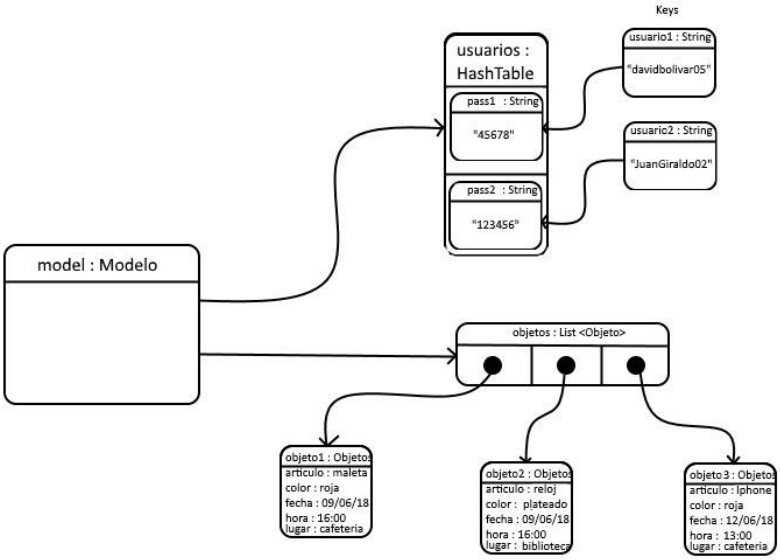
Reporte de artículos				
Criterio **Propuesta**	**Implementación de la solución**	**Sencillez de la solución**	**Eficacia de la solución**	**Totales**
Separar los datos de cada registro del archivo de texto y mostrarlos en un DataGridView	**3**	**4**	**5**	**12**
Mostrar los registros en un Textbox en forma de lista y separar cada dato con algún carácter o espacio.	**5**	**5**	**5**	**15**
Mostrar los registros en los controles donde se registra la información (textbox, combobox, etc) y agregar botones de navegación (adelante y atrás) para poder acceder a todos	**2**	**4**	**5**	**11**

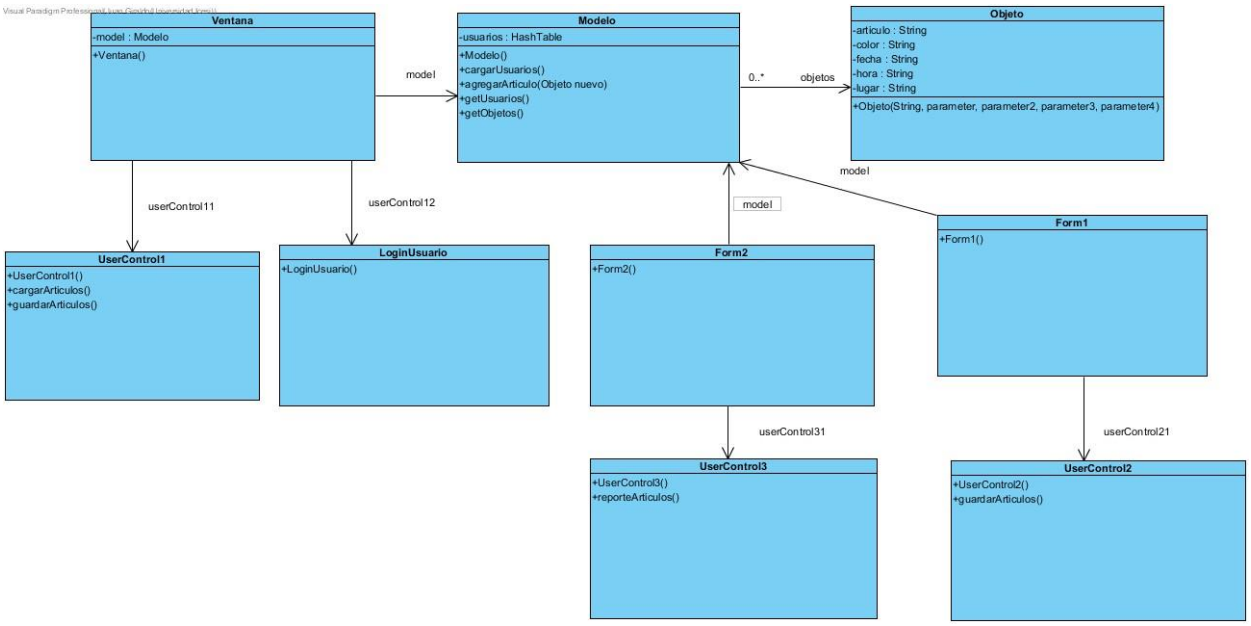
los registros.				
Utilizar el lenguaje nativo de consulta LinQ para facilitar el reporte de información.	**4**	**5**	**5**	**14**

Teniendo en cuenta que las ganadoras tienen su resultado marcado con verde, como se puede observar no hay una única ganadora con excepción de la primera evaluación. Lo que se puede concluir es que combinar estas soluciones es la mejor manera de dar solución al problema. Dado que las escogidas son sencillas de implementar, combinarlas no es una tarea difícil.



Diseños preliminares


Keys
usuarios : HashTable
pass1 : String
"45678"
pass2 : String
"123456"
usuario1 : String
"davidbolivar05"
usuario2 : String
"JuanGiraldo02"
model : Modelo
objetos : List <Objeto>
objeto1 : Objeto
articulo : maleta
color : roja
fecha : 09/06/18
hora : 16:00
lugar : cafeteria
objeto2 : Objeto
articulo : reloj
color : plateado
fecha : 09/06/18
hora : 16:00
lugar : biblioteca
objeto3 : Objeto
articulo : Iphone
color : roja
fecha : 12/06/18
hora : 13:00
lugar : cafeteria


Ventana
-model : Modelo
+Ventana()
model
Modelo
-usuarios : HashTable
+Modelo()
+cargarUsuarios()
+agregarArticulo(Objeto nuevo)
+getUsuarios()
+getObjetos()
0..*
objetos
Objeto
-articulo : String
-color : String
-fecha : String
-hora : String
-lugar : String
+Objeto(String, parameter, parameter2, parameter3, parameter4)
model
userControl11
userControl12
model
UserControl1
+UserControl1()
+cargarArticulos()
+guardarArticulos()
LoginUsuario
+LoginUsuario()
Form2
+Form2()
Form1
+Form1()
userControl31
userControl21
UserControl3
+UserControl3()
+reporteArticulos()
UserControl2
+UserControl2()
+guardarArticulos()

6. **Preparación de informes y especificaciones**

   Especificación del problema:

   Problema: Registro de los datos de objetos abandonados es poco adecuado.

   Entradas: Nombre del objeto recuperado, hora, fecha, zona, detalles.

   Salida: Registro del reporte del objeto en el archivo de texto plano.

7. **Implementación del diseño**

   Tareas que implementar:

   - Cargar los usuarios registrados

Nombre	cargarUsuarios()
Descripción	Permite cargar los usuarios que esten registrados previamente:
Entrada	Archivo .txt con todos los datos
Salida	Los usuarios han sido cargados a una HashTable

Código:

```
public void cargarUsuarios() {

            String pathUsuarios = "..\\..\\Usuarios.txt";
            List<String> lines =
File.ReadAllLines(pathUsuarios).ToList();
            foreach (string item in lines) {
                String[] split = item.Split('-');
                usuarios.Add(split[0],split[1]);
                Console.WriteLine("ok");
            }

        }
```

   - Cargar los artículos guardados

Nombre	cargarArticulos()
Descripción	Permite cargar los artículos guardados previamente
Entrada	Archivo .txt con todos los datos de los articulos
Salida	Los articulos han sido cargados a una lista de articulos

Código:

```
public void cargarArticulos() {
            StreamWriter file = new
StreamWriter("..\\..\\Objetos.txt");
            Modelo model = Ventana.darModelo();
            List<Objeto> listaObjetos = model.getObjetos();
            foreach (Objeto item in listaObjetos) {
                file.WriteLine((item.getArticulo()+"-
"+item.getColor()+"-"+item.getFecha()+"-"+item.getHora()+"-
"+item.getLugar()).Trim());
            }

            file.Close();

        }
```

- Agregar articulo

Nombre	agregarArticulo()
Descripción	Permite agregar un articulo a la lista de artículos
Entrada	-Articulo -Color -Lugar -Hora

	-Fecha
Salida	El articulo ha sido agregado a la lista de articulos

Código:

```
public void agregarArticulo() {
Modelo model = Ventana.darModelo();
            String objeto = textBox1.Text;
            String lugar = textBox2.Text;
            String hora = textBox3.Text;
            String fecha = textBox4.Text;
            String color = textBox5.Text;
            Objeto nuevo = new Objeto(objeto,color,fecha,hora,lugar);
            model.agregarArticulo(nuevo);

            textBox1.Text = "Nombre";
            textBox2.Text = "lugar";
            textBox3.Text = "hora";
            textBox4.Text = "fecha";
            textBox5.Text = "color";
        }       }
```

- Guardar los artículos agregados

Nombre	guardarArticulos()
Descripción	Permite guardar los artículos agregados previamente
Entrada	-Lista de los artículos agregados
Salida	Los artículos han sido guardados en un archivo .txt

Código:

```
public void guardarArticulos() {
              String pathUsuarios = "..\\..\\Objetos.txt";
            List<String> lines =
File.ReadAllLines(pathUsuarios).ToList();
            Modelo model = Ventana.darModelo();
            List<Objeto> lista = model.getObjetos();
```

```
            foreach (string item in lines)
            {
                String[] split = item.Split('-');
                String articulo = split[0];
                String color = split[1];
                String fecha = split[2];
                String hora = split[3];
                String lugar = split[4];
                Objeto nuevo = new
Objeto(articulo,color,fecha,hora,lugar);
                lista.Add(nuevo);
            }
        }
```

- Hacer el reporte de los artículos

Nombre	reporteArticulos()
Descripción	Permite generar un reporte de los artículos previamente registrados
Entrada	-Reporte por nombre -Reporte por color -Reporte por lugar -Palabra clave
Salida	Los artículos encontrados con la palabra clave y la clasificación son mostrados en pantalla

Código:

```
public void reportarArticulos() {
               if (checkBox2.Checked)
            {
                var consulta = from s in lista
```

```
                                where
s.getArticulo().Contains(textBox1.Text)
                                select s;
                String msg = "";
                foreach (Objeto item in consulta)
                {
                    msg += "Articulo: " + item.getArticulo() + "-
Color: " + item.getColor() + "-Lugar: " + item.getLugar() + "-Fecha: "
+ item.getFecha() + "-Hora: " + item.getHora() + "\n";
                }
                richTextBox1.Text = msg;

            }

            else if (checkBox1.Checked)
            {
                var consulta = from s in lista
                                where
s.getColor().Contains(textBox1.Text)
                                select s;
                String msg = "";
                foreach (Objeto item in consulta)
                {
                    msg += "Articulo: " + item.getArticulo() + "-
Color: " + item.getColor() + "-Lugar: " + item.getLugar() + "-Fecha: "
+ item.getFecha() + "-Hora: " + item.getHora() + "\n" + "\n";
                }
                richTextBox1.Text = msg;

            }

            else if (checkBox3.Checked)
            {
                var consulta = from s in lista
                                where
s.getLugar().Contains(textBox1.Text)
                                select s;
                String msg = "";
                foreach (Objeto item in consulta)
                {
                    msg += "Articulo: " + item.getArticulo() + "-
Color: " + item.getColor() + "-Lugar: " + item.getLugar() + "-Fecha: "
+ item.getFecha() + "-Hora: " + item.getHora() + "\n" + "\n";
                }
                richTextBox1.Text = msg;

            }
            else {
                var consulta = from s in lista
                                select s;
                String msg = "";
                foreach (Objeto item in consulta)
                {
```

```
                    msg += "Articulo: " + item.getArticulo() + "-
Color: " + item.getColor() + "-Lugar: " + item.getLugar() + "-Fecha: "
+ item.getFecha() + "-Hora: " + item.getHora() + "\n"+"\n";
                }
                richTextBox1.Text = msg;

            }
        }
```