

Universidad de San Carlos de Guatemala

Facultad de ingeniería

Escuela de ciencias y sistemas

Manual técnico del programa EDD mail

Victor Hugo Velasquez

202100054

Estudiante de Ingeniería en ciencias y Sistemas

Estructura de datos sección C

Introducción

Introducción al Manual Técnico

Este manual técnico se adentra en la arquitectura y el funcionamiento interno del sistema de gestión de correos electrónicos

EDDMail, una aplicación desarrollada en Object Pascal. El objetivo es documentar el código fuente, las estructuras de datos dinámicas y las librerías utilizadas para construir un sistema que simula un entorno de correo electrónico real.

Arquitectura y Estructuras de Datos

El sistema se basa en una variedad de estructuras de datos para gestionar eficientemente la información. A continuación, se detallan las principales:

- **Usuarios:** Se implementa una **lista simple** para almacenar y administrar los usuarios registrados
- **Bandeja de Entrada:** Para gestionar los correos recibidos por cada usuario, se utiliza una **lista doblemente enlazada**.
- **Contactos:** La representación de los contactos de cada usuario se logra con una **lista circular**.
- **Correos Programados:** Los correos destinados a ser enviados automáticamente se gestionan mediante una **cola**, procesándolos en un orden FIFO (primero en entrar, primero en salir).
- **Papelera:** Los correos eliminados se almacenan en una **pila** para su posterior gestión
- **Comunidades:** Se implementa una **lista de listas** para manejar las comunidades y los usuarios que pertenecen a cada una de ellas
- **Matriz Dispersa:** Para visualizar las relaciones de envío y recepción de correos entre usuarios, se utiliza una **matriz dispersa**.

Especificaciones de las estructuras con código pascal

En esta parte del código, hace referencia a la funcionalidad que tiene el botón numero 1 que en este caso es el de ingresar

```
1  procedure TForm1.Button1Click(Sender: TObject);
2  var U: PUsuario;
3  begin
4      if (Trim(Edit1.Text) = '') or (Edit2.Text = '') then
5      begin
6          ShowMessage('Por favor, ingrese su email y contraseña.');
```

```
7          Exit;
8      end;
9
10     U := BuscarUsuarioPorEmail(Edit1.Text);
11     if (U <> nil) and (U^.Password = Edit2.Text) then
12     begin
13         CurrentUser := U;
14         if EqualCI(U^.Email, 'root@edd.com') then
15         begin
16             ShowMessage('¡Bienvenido, Administrador!');
```

```
17             Hide;
18             root.Form2 := root.TForm2.Create(Application);
19             root.Form2.Show;
20         end
21     else
22     begin
23         ShowMessage('¡Bienvenido, ' + U^.Email + '!');
```

```
24         Hide;
25         user.Form3 := user.TForm3.Create(Application);
26         user.Form3.Show;
27     end;
28 end
29 else
30     ShowMessage('Credenciales incorrectas. Intente de nuevo.');
```

```
31 end;
```

En este botón que es el numero 2 podemos ver la lógica en cuanto el usuario le de click , si los campos están vacíos; dará un mensaje de advertencia que en este caso es una ventana emergente en donde le indique que no se puede pasar a proceder nada porque no hay datos dentro de los campos

```
1  procedure TForm1.Button2Click(Sender: TObject);
2  var email, password: string;
3  begin
4      email := Trim(Edit1.Text);
5      password := Edit2.Text;
6
7      if (email = '') or (password = '') then
8      begin
9          ShowMessage('Ingrese email y contraseña para registrarse.');
```

```
10         Exit;
11     end;
12
13     if not EmailValido(email) then
14     begin
15         ShowMessage('El formato de email no es válido.');
```

```
16         Exit;
17     end;
18
19     if EqualCI(email, 'root@edd.com') then
20     begin
21         ShowMessage('El usuario root ya existe y no puede registrarse nuevamente.');
```

```
22         Exit;
23     end;
24
25     if BuscarUsuarioPorEmail(email) <> nil then
26     begin
27         ShowMessage('El usuario ya existe dentro del sistema.');
```

```
28         Exit;
29     end;
30
31     AgregarUsuario('', '', email, '', password);
32     ShowMessage('Usuario registrado con éxito. Ahora puede iniciar sesión.');
```

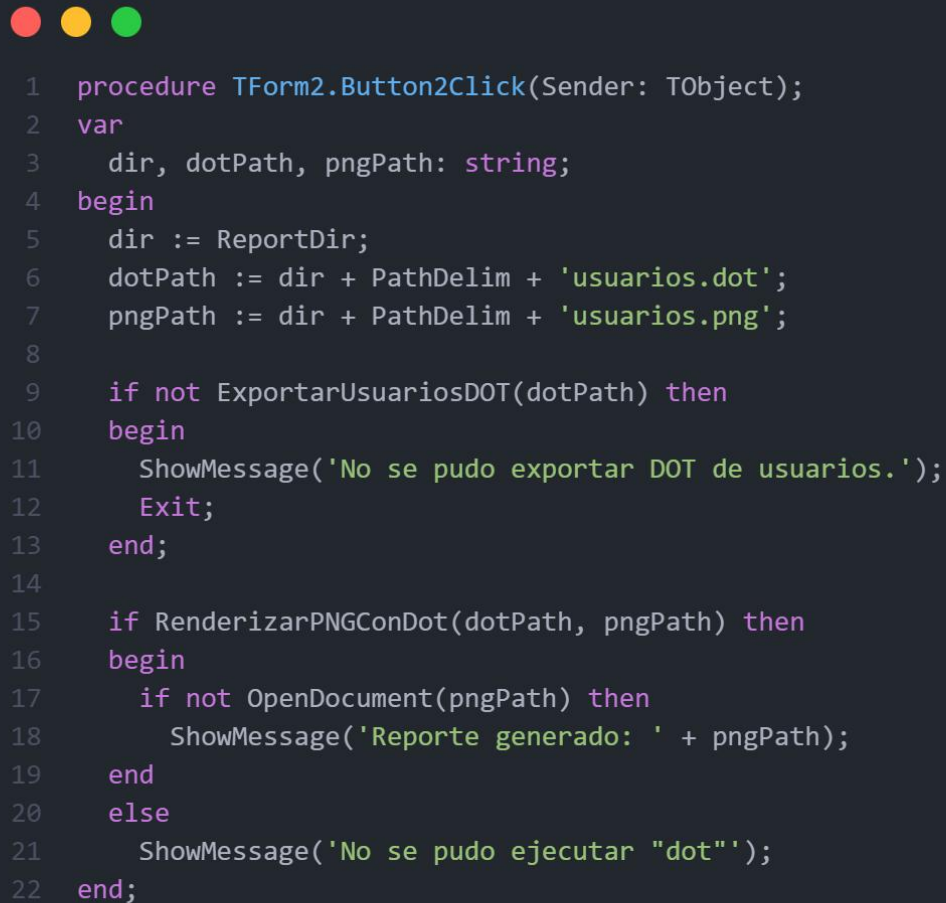
```
33 end;
```

Esto se toma en cuenta para el código que tiene el Form de logeo.pas

Botón del root.pas para cargar JSON

```
1  procedure TForm2.Button1Click(Sender: TObject);
2  var
3      OD: TOpenDialog;
4      J: TJSONData;
5      Obj: TJSONObject;
6      Arr: TJSONArray;
7      I: Integer;
8      U: TJSONObject;
9      sl: TStringList;
10     email, password, nombre, usuario, telefono: string;
11
12     function SDef(O: TJSONObject; const Key: string): string;
13     begin
14         if (O = nil) or (O.Find(Key) = nil) then Exit('');
15         Result := O.Get(Key, '');
16     end;
17 begin
18     // CARGA MASIVA de usuarios
19     OD := TOpenDialog.Create(Self);
20     try
21         OD.Filter := 'JSON|*.json';
22         if not OD.Execute then Exit;
23
24         sl := TStringList.Create;
25         try
26             sl.LoadFromFile(OD.FileName);
27             J := GetJSON(sl.Text);
28         finally
29             sl.Free;
30         end;
31
32         try
33             Obj := TJSONObject(J);
34             Arr := TJSONArray(Obj.Find('usuarios'));
35             if Arr = nil then begin ShowMessage('JSON sin "usuarios".'); Exit; end;
36
37             for I := 0 to Arr.Count-1 do
38                 begin
39                     if not (Arr.Items[I] is TJSONObject) then Continue;
40                     U := TJSONObject(Arr.Items[I]);
41                     nombre := SDef(U, 'nombre');
42                     usuario := SDef(U, 'usuario');
43                     email := SDef(U, 'email');
44                     telefono := SDef(U, 'telefono');
45                     password := SDef(U, 'password');
46
47                     if (email <> '') and (BuscarUsuarioPorEmail(email) = nil) then
48                         AgregarUsuario(nombre, usuario, email, telefono, password);
49                 end;
50
51                 ShowMessage('Carga masiva completada.');
```

El botón numero 2 del Form root.pas lo que le indica al usuario es que a la hora de que se presione el botón genera la grafica de lo que es la lista enlazada de los usuarios cargados en el JSON



```
1 procedure TForm2.Button2Click(Sender: TObject);
2 var
3     dir, dotPath, pngPath: string;
4 begin
5     dir := ReportDir;
6     dotPath := dir + PathDelim + 'usuarios.dot';
7     pngPath := dir + PathDelim + 'usuarios.png';
8
9     if not ExportarUsuariosDOT(dotPath) then
10    begin
11        ShowMessage('No se pudo exportar DOT de usuarios.');
```

The image shows a screenshot of a code editor with a dark background. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The code is written in Pascal and is a procedure named TForm2.Button2Click. It declares three string variables: dir, dotPath, and pngPath. It then sets dir to ReportDir, dotPath to dir plus a path delimiter and 'usuarios.dot', and pngPath to dir plus a path delimiter and 'usuarios.png'. The procedure then checks if ExportarUsuariosDOT(dotPath) returns false. If so, it shows a message box 'No se pudo exportar DOT de usuarios.' and calls Exit. If not, it checks if RenderizarPNGConDot(dotPath, pngPath) returns true. If so, it shows a message box 'Reporte generado: ' + pngPath. If not, it shows a message box 'No se pudo ejecutar "dot"'. The procedure ends with end;.

```
12     Exit;
13 end;
14
15 if RenderizarPNGConDot(dotPath, pngPath) then
16 begin
17     if not OpenDocument(pngPath) then
18         ShowMessage('Reporte generado: ' + pngPath);
19 end
20 else
21     ShowMessage('No se pudo ejecutar "dot"');
22 end;
```

En el botón numero 3 del root.pas se le indica que cuando le de click genere un archivo .dot y un .png en donde pueda mostrar la relación de la matriz dispersa entre los correos enviados por los usuarios

```
1  procedure TForm2.Button3Click(Sender: TObject);
2  var
3      dir, dotPath, pngPath: string;
4  begin
5      // REPORTE DE RELACIONES (MATRIZ DISPERSA)
6      dir := ReportDir;
7      dotPath := dir + PathDelim + 'relaciones.dot';
8      pngPath := dir + PathDelim + 'relaciones.png';
9
10     if not ExportarRelacionesDOT(dotPath) then
11     begin
12         ShowMessage('No se pudo exportar el DOT de relaciones.');
```

Ahora empezamos con el código para lo que es el Form user.pass

Por acá tenemos el botón que envia correos a distintos tipos de usuarios registrados



```
1 procedure TForm3.Button1Click(Sender: TObject);
2 var F: TInboxWin;
3 begin
4     if CurrentUser = nil then begin SafeMsg('Inicie sesión.');
```

```
Exit; end;
5     F := TInboxWin.CreateForUser(Self, CurrentUser);
6     try F.ShowModal; finally F.Free; end;
7 end;
```



```
1 {--- Enviar ---}
2
3 constructor TSendWin.CreateSimple(AOwner: TComponent);
4 var
5     lbl: TLabel;
6 begin
7     inherited CreateNew(AOwner, 1);
8     Caption := 'Enviar correo';
9     Position := posScreenCenter; Width := 720; Height := 520;
10
11     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Para (email)'; lbl.Left := 16; lbl.Top := 20;
12     edtPara := TEdit.Create(Self); edtPara.Parent := Self; edtPara.Left := 120; edtPara.Top := 16; edtPara.Width := 360;
13
14     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Asunto'; lbl.Left := 16; lbl.Top := 56;
15     edtAsunto := TEdit.Create(Self); edtAsunto.Parent := Self; edtAsunto.Left := 120; edtAsunto.Top := 52; edtAsunto.Width := 560;
16
17     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Mensaje'; lbl.Left := 16; lbl.Top := 92;
18     memoMsg := TMemo.Create(Self); memoMsg.Parent := Self; memoMsg.Left := 120; memoMsg.Top := 92; memoMsg.Width := 560; memoMsg.Height := 320;
19
20     btnEnviar := TButton.Create(Self); btnEnviar.Parent := Self; btnEnviar.Caption := 'Enviar'; btnEnviar.Left := 120; btnEnviar.Top := 424; btnEnviar.Width := 120; btnEnviar.OnClick := @SendDo;
21     btnCerrar := TButton.Create(Self); btnCerrar.Parent := Self; btnCerrar.Caption := 'Cerrar'; btnCerrar.Left := 560; btnCerrar.Top := 424; btnCerrar.Width := 120; btnCerrar.ModalResult := mrClose;
22 end;
```


Este es el código que reconoce la lógica para poder programar un correo con fecha y destinatario designado

```
1  {--- Programar ---}
2
3  constructor TProgWin.CreateSimple(AOwner: TComponent);
4  var
5      lbl: TLabel;
6  begin
7      inherited CreateNew(AOwner, 1);
8      Caption := 'Programar envío';
9      Position := poScreenCenter; Width := 720; Height := 520;
10
11     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Para (email)'; lbl.Left := 16; lbl.Top := 20;
12     edtPara := TEdit.Create(Self); edtPara.Parent := Self; edtPara.Left := 120; edtPara.Top := 16; edtPara.Width := 360;
13
14     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Asunto'; lbl.Left := 16; lbl.Top := 56;
15     edtAsunto := TEdit.Create(Self); edtAsunto.Parent := Self; edtAsunto.Left := 120; edtAsunto.Top := 52; edtAsunto.Width := 560;
16
17     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Mensaje'; lbl.Left := 16; lbl.Top := 92;
18     memoMsg := TMemo.Create(Self); memoMsg.Parent := Self; memoMsg.Left := 120; memoMsg.Top := 92; memoMsg.Width := 560; memoMsg.Height := 320;
19
20     lbl := TLabel.Create(Self); lbl.Parent := Self; lbl.Caption := 'Fecha (YYYY-MM-DD o DD/MM/AAAA)'; lbl.Left := 16; lbl.Top := 424;
21     edtFecha := TEdit.Create(Self); edtFecha.Parent := Self; edtFecha.Left := 260; edtFecha.Top := 420; edtFecha.Width := 180;
22
23     btnProgramar := TButton.Create(Self); btnProgramar.Parent := Self; btnProgramar.Caption := 'Programar'; btnProgramar.Left := 460; btnProgramar.Top := 420; btnProgramar.Width := 120; btnProgramar.OnClick := @ProgEnqueue;
24     btnCerrar := TButton.Create(Self); btnCerrar.Parent := Self; btnCerrar.Caption := 'Cerrar'; btnCerrar.Left := 590; btnCerrar.Top := 420; btnCerrar.Width := 90; btnCerrar.ModalResult := mrClose;
25 end;
```

Acá tenemos el código para la lista de programados

```
1  {--- Lista de programados ---}
2
3  constructor TProgListWin.CreateSimple(AOwner: TComponent);
4  begin
5      inherited CreateNew(AOwner, 1);
6      Caption := 'Programados';
7      Position := poScreenCenter; Width := 820; Height := 520;
8
9      LV := TListView.Create(Self); LV.Parent := Self; LV.Align := alClient; LV.ViewStyle := vsReport; LV.ReadOnly := True; LV.RowSelect := True; LV.GridLines := True;
10     with LV.Columns.Add do begin Caption := 'Id'; Width := 60; end;
11     with LV.Columns.Add do begin Caption := 'Remitente'; Width := 180; end;
12     with LV.Columns.Add do begin Caption := 'Para'; Width := 200; end;
13     with LV.Columns.Add do begin Caption := 'Prog. Fecha'; Width := 180; end;
14     with LV.Columns.Add do begin Caption := 'Asunto'; Width := 180; end;
15
16     pnlBtns := TPanel.Create(Self); pnlBtns.Parent := Self; pnlBtns.Align := alBottom; pnlBtns.Height := 44;
17     btnProcVencidos := TButton.Create(Self); btnProcVencidos.Parent := pnlBtns; btnProcVencidos.Caption := 'Procesar vencidos (ahora)'; btnProcVencidos.Left := 8; btnProcVencidos.Top := 8; btnProcVencidos.Width := 200; btnProcVencidos.OnClick := @ProcessDue;
18     btnRefrescar := TButton.Create(Self); btnRefrescar.Parent := pnlBtns; btnRefrescar.Caption := 'Refrescar'; btnRefrescar.Left := 212; btnRefrescar.Top := 8; btnRefrescar.Width := 100; btnRefrescar.OnClick := @Refresh;
19     btnCerrar := TButton.Create(Self); btnCerrar.Parent := pnlBtns; btnCerrar.Caption := 'Cerrar'; btnCerrar.Left := 720; btnCerrar.Top := 8; btnCerrar.Width := 80; btnCerrar.ModalResult := mrClose;
20
21     LoadList;
22 end;
```

Este es el código para la papelera de los correos que el usuario desea eliminar

```
1  {--- Papelera ---}
2
3  constructor TrashMin.CreateSimple(Owner: TComponent);
4  begin
5      inherited CreateNew(Owner, 1);
6      Caption := 'Papelera';
7      Position := posScreenCenter; Width := 820; Height := 520;
8
9      LV := TListView.Create(Self); LV.Parent := Self; LV.Align := alClient; LV.ViewStyle := vsReport; LV.ReadOnly := True; LV.RowSelect := True; LV.GridLines := True;
10     with LV.Columns.Add do begin Caption := 'ID'; Width := 60; end;
11     with LV.Columns.Add do begin Caption := 'Remitente'; Width := 180; end;
12     with LV.Columns.Add do begin Caption := 'Fecha/Hora'; Width := 160; end;
13     with LV.Columns.Add do begin Caption := 'Asunto'; Width := 300; end;
14
15     pnlBtns := TPanel.Create(Self); pnlBtns.Parent := Self; pnlBtns.Align := alBottom; pnlBtns.Height := 44;
16     btnRestaurar := TButton.Create(Self); btnRestaurar.Parent := pnlBtns; btnRestaurar.Caption := 'Restaurar seleccionado'; btnRestaurar.Left := 8; btnRestaurar.Top := 8; btnRestaurar.Width := 180; btnRestaurar.OnClick := @RestoreSelected;
17     btnVaciar := TButton.Create(Self); btnVaciar.Parent := pnlBtns; btnVaciar.Caption := 'Vaciar Papelera'; btnVaciar.Left := 196; btnVaciar.Top := 8; btnVaciar.Width := 140; btnVaciar.OnClick := @EmptyAll;
18     btnCerrar := TButton.Create(Self); btnCerrar.Parent := pnlBtns; btnCerrar.Caption := 'Cerrar'; btnCerrar.Left := 720; btnCerrar.Top := 8; btnCerrar.Width := 80; btnCerrar.ModalResult := mrClose;
19
20     LoadList;
21 end;
```