Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Estructura de Datos
Ing. Edgar Rene Ornelis
Auxiliar: Steven Facundo Mejía

Manual Técnico

Jorge Ivan Samayoa Sian
202307506
Guatemala 06 de septiembre de 2025

Introducción

EDDMail es un sistema de simulación de correos electrónicos desarrollado en Object Pascal y usando la librería GTK para el desarrollo de la interfaz. Este sistema permite a los usuarios enviar y recibir correos, así como gestionar su lista de contactos. EDDMail también hace uso de Graphviz y un sistema de generación de archivos .dot para la generación de reportes.

Especificación Técnica

- **Lenguaje de desarrollo:** Objetc Pascal
- **IDE:** Libre (En este caso se usó Visual Studio Code)
- **SO:** Linux, distribución libre (En este caso se usó Debian 13)
- **Librerías**: GTK2, GLIB2 y GDK2
- **Herramienta gráfica:** Graphviz

Estructura del proyecto

- **Usuarios:** Maneja la información de cada uno de los usuarios (id, nombre, usuario, contraseña, correos y teléfono). Son almacenados en una lista simple.

```
PNodo = ^TNodo;
TNodo = record
    id: String;
    nombre: String;
    usuario: String;
    password: String;
    email: String;
    telefono: String;

    //Estructuras para cada usuario
    contactos: PContacto;
    correos: PCorreo;
    papelera: PPila;
    correosProgramados: PProgramados;
    siguiente: PNodo;
end;
```

Se declaran variables en el mismo usuario que albergarán las diferentes estructuras manejadas en nuestro proyecto. Esto se hace con la finalidad de que cada usuario creado tenga sus propias listas evitando una globalización.

Funciones y procesos de la lista simple para el manejo de los usuarios y generación del archivo .dot para el reporte:

```
procedure Insertar(id, nombre, usuario, email, telefono, password: String);
procedure imprimir;
function generarDotLS: string;
function validarCredenciales(email, password: string): boolean;
function buscarPorEmail(const email: string): TDatos;
function obtenerNodoPorEmail(const email: string): PNodo;
```

- **<u>Correos:</u>** Maneja la información de cada correo (id, remitente, estado (L/NL), programado, asunto, fecha y mensaje). Estos son almacenados en una lista doblemente enlazada.

```
//Lista doble de correos
PCorreo = ^TCorreo;
TCorreo = record
    idCorreo: String;
    remitente: String;
    estado: String;
    programado: Boolean;
    asunto: String;
    fecha: TDateTime;
    mensaje: String;
    siguiente, anterior: PCorreo;
end;
```

Su función es almacenar los correos que envía el remitente pero en la lista del destinatario, esto se logra validando primero si el destinatario existe y si lo encuentra llama a la función insertarDoble e inserta el correo en la lista del destinatario.

```
    Exit;
end;

// Buscar al usuario REAL (destinatario) por email
contactoDestino := buscarUsuarioPorEmail(listaUsuarios, correoDestinatario);

if contactoDestino = nil then
begin
    mostrarMensajeError(enviarCorreoWindow, 'Error', 'No se encontró el usuario destino.');
    Exit;
end;

// Insertar el correo en la lista de correos del usuario destino
insertarDoble(contactoDestino^.correos, idCorreo, remitente, estado, programado, asunto, fechaEnvio, mensaje);
mostrarMensajeLogin(enviarCorreoWindow, 'Correo Enviado', 'El correo ha sido enviado exitosamente.');
imprimirListaDoble(contactoDestino^.correos);
```

Funciones de la lista doble:

```
procedure insertarDoble(var lista: PCorreo; idCorreo, remitente, estado: string;
    programado: Boolean; asunto: string; fecha: TDateTime; mensaje: string);
function EscapeDotString(const S: string): string;
function generarDotLD(lista: PCorreo): string;
procedure imprimirListaDoble(lista: PCorreo);
```

- Papelera: Almacena los correos parcialmente eliminados por el usuario remitente. Lo hace mediante una pila

```
//Pila de correos
PPila = ^TPila;
TPila = record
    idCorreo: String;
    remitente: String;
    estado: String;
    programado: Boolean;
    asunto: String;
    fechaEnvio: TDateTime;
    mensaje: String;
    siguiente: PPila;
end;
```

Funciones principales de la pila:

```
procedure insertarPila(var pila: PPila; idCorreo, remitente, estado: String;
    programado: Boolean; asunto: String; fechaEnvio: TDateTime; mensaje: String);
procedure eliminarPila(var pila: PPila);
procedure restablecerPila(var pila: PPila; var bandeja: PCorreo);
function EscapeDotString(const S: string): string;
function generarDotPila (pila: PPila): string;
procedure imprimirPila(pila: PPila);
```

- Correos Programados: Usa una cola para almacenar los correos que han sido programados para enviarse, sin embargo, a diferencia de los correos que se insertan en la lista doble del destinatario, estos correos si se insertan en la cola del remitente.

```
//Cola de correos programados
PProgramados = ^TProgramados;
TProgramados = record
    idCorreo: String;
    remitente: String;
    estado: String;
    programado: Boolean;
    asunto: String;
    fechaEnvio: TDateTime;
    mensaje: String;
    destinatario: String;
    siguiente: PProgramados;
end;
PPProgramados = ^PProgramados;
```

Funciones principales de la cola:

```
procedure insertarCola(var cola: PProgramados; idCorreo, remitente, estado: String; programado: Boolean;
    asunto: String; fechaEnvio: TDateTime; mensaje, destinatario: String);
procedure eliminarCola(var cola: PProgramados);
function EscapeDotString(const S: string): string;
function generarDotCola (cola: PProgramados): string;
procedure imprimirCola(cola: PProgramados);
```

- Contactos: Tiene como función almacenar los contactos, que el actual usuario logueado agrega, en una lista circular.

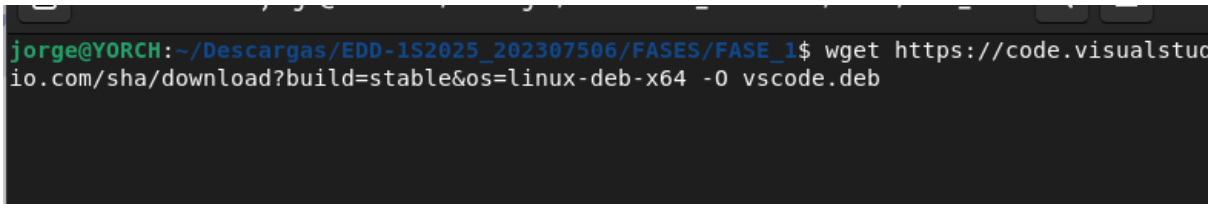
```
//Lista circular de contactos
PContacto = ^TContacto;
TContacto = record
    id: String;
    nombre: String;
    usuario: String;
    email: String;
    telefono: String;
    siguiente: PContacto;
end;
```

Sus funciones principales son:

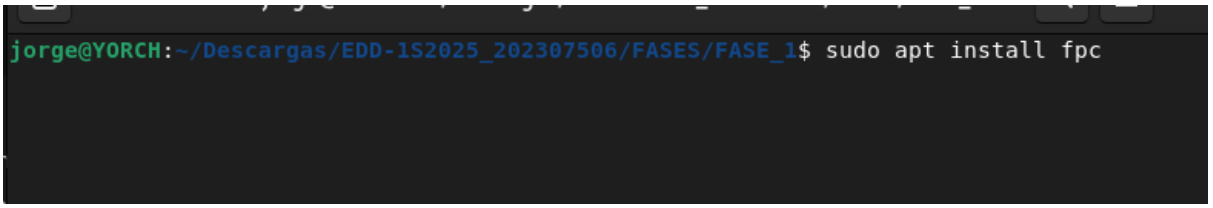
```
procedure InsertarCircular(var lista: PContacto; id, nombre, email, usuarioContacto, telefono: string);
function EscapeDotString(const S: string): string;
function generarDotLC (lista: PContacto): string;
function obtenerContactoPorID(lista: PContacto; id: string): TDatos;
function existeContacto(lista: PContacto; email: string): Boolean;
function buscarUsuarioPorEmail(listaUsuarios: PNodo; email: String): PNodo;
```

COMPILACIÓN Y COMANDOS

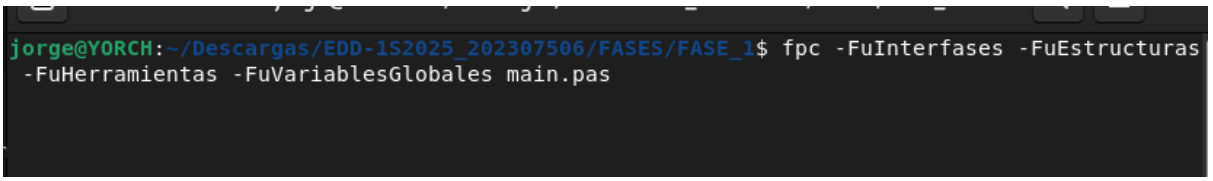
Comando para instalar VS:



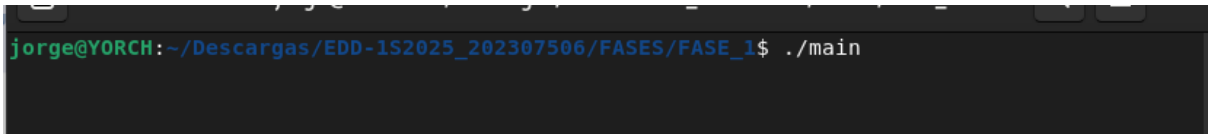
Comando para instalar Free Pascal:



Comando para compilar el proyecto:



Comando para ejecutar el proyecto:



Diferentes comandos para borrar los compilados:

```
# Eliminar los archivos generados para compilar
rm *.o
rm main

# Eliminar los archivos generados en la carpeta interfaces
rm interfaces/*.o;
rm interfaces/*.ppu;

# Eliminar los archivos generados en la carpeta structures
rm structures/*.o;
rm structures/*.ppu;

# Eliminar los archivos generados en la carpeta tools
rm tools/*.o;
rm tools/*.ppu;

# Eliminar los archivos generados en la carpeta data
rm data/*.o;
rm data/*.ppu;
```