# http://upload.wikimedia.org/wikipedia/commons/thumb/5/5f/Escudo_UdeG.svg/1000px-Escudo_UdeG.svg.png

# UNIVERSIDAD DE GUADALAJARA

## CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

## DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN

## DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

## SEMINARIO DE SOLUCIÓN DE PROBLEMAS DE ESTRUCTURAS DE DATOS II

#### Mtro. Sergio Manuel Bolaños Gutiérrez

### PROYECTO FINAL

### Simulador de Lector de Correo Electrónico

##### DESCRIPCIÓN

Un lector de correo electrónico, también llamado cliente de correo electrónico, es un software que administra la información almacenada en un servidor de correo electrónico para hacerla accesible al usuario. Existen clientes basados en Web y autónomos, que se ejecutan de forma nativa en la plataforma de cómputo, ya sea un dispositivo móvil o de escritorio.

El número de correos electrónicos almacenados en una cuenta de un usuario promedio alcanza las decenas de miles; ésta información, resguardada en almacenamiento secundario, se traduce por lo general en decenas de gigabytes de información organizada en archivos.

Los requerimientos de tiempo de respuesta para la recuperación de esta información son altos: un usuario típico deseará poder encontrar y leer cualquier correo electrónico en cuestión de segundos. Para poder satisfacer requerimientos de esta clase para los usuarios finales es necesario implementar estructuras de datos avanzadas en combinación con el almacenamiento secundario.

Adicionalmente, existen requerimientos de seguridad, puesto que el correo electrónico es utilizado comúnmente para el manejo de información confidencial, un usuario espera poder mantener la información sensible bajo protección, lo que se logra mediante el cifrado, además, debido a que estos archivos necesitan un gran espacio en disco, generalmente es necesario comprimirlos.

# ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES

Diseñe y codifique un programa simulador de lector de correo electrónico que cumpla con los siguientes requerimientos funcionales:

1. El programa no se conectará a un servidor de correo electrónico real, sino que trabajará en modo fuera de línea.
2. El software almacenará la información de los correos electrónicos en archivos de acceso directo con registros de longitud fija en formato binario.
3. Escribir Correo Electrónico: el programa debe permitir al usuario escribir un correo electrónico nuevo y almacenarlo en un archivo binario de datos.

Cada mensaje de correo electrónico será almacenado como un registro de longitud fija con los siguientes campos:

1. Identificador de registro de 10 dígitos (asignado por el usuario).
   1. Fecha de Envío.
   2. Hora de Envío.
   3. Remitente (From: ).
   4. Destinatario (To: ).
   5. Copia Carbón (CC: ).
   6. Copia Carbón Ciega (BCC: ).
   7. Asunto (Subject: ).
   8. Contenido (debe soportar saltos de línea, es decir, diferentes párrafos).
2. Leer Correo Electrónico: el software debe proveer un mecanismo para que el usuario pueda leer el correo electrónico almacenado.
   * 1. Acceso Directo: el usuario puede buscar y leer un correo electrónico introduciendo directamente el identificador de 10 dígitos del registro asignado al correo electrónico por el sistema.
     2. Acceso Secuencial: en caso de que el usuario no conozca el identificador del correo electrónico, el programa proveerá una forma de buscar correos electrónicos por el remitente (búsqueda secuencial).
3. Modificar Correo Electrónico: el programa debe admitir la modificación de cualquier correo electrónico que el usuario solicite.
   1. Para seleccionar el correo electrónico a modificar, el usuario podrá:
      1. Buscar el correo electrónico a modificar por identificador (**§** 4.1.1).
      2. Buscar el correo electrónico a modificar por remitente (**§** 4.1.2).
   2. Se podrá actualizar cualquier campo excepto el identificador del registro del correo electrónico.
4. Eliminación de Correos Electrónicos: el programa debe permitir al usuario eliminar cualquier correo electrónico que el usuario indique.
   1. Para seleccionar el correo electrónico a modificar el usuario podrá:
      1. Buscar el correo electrónico a modificar por identificador (**§** 4.1.1).
      2. Buscar el correo electrónico a modificar por remitente (**§** 4.1.2).
5. Módulo de Copias de Seguridad
   1. El sistema proveerá al usuario un mecanismo a través del cual podrá hacer copias de seguridad de los correos electrónicos.
   2. Las copias de seguridad se harán en un formato de archivos común para el intercambio de información: Valores Separados por Comas o CSV (*Comma-Separated Values*).
   3. Los archivos CSV usados por el módulo de copias de seguridad deben cumplir con el RFC 4180: <http://tools.ietf.org/html/rfc4180>, definido por la *Internet Engineering Task Force*.
   4. Exportar Correos Electrónicos: el usuario podrá solicitar exportar los correos electrónicos a un archivo CSV para su respaldo.
      1. El sistema exportará todos los correos electrónicos almacenados en el archivo de datos principal.
      2. Todos los campos del registro de correo electrónico deben ser exportados, en el orden en que están definidos en el requerimiento §3.
      3. El archivo tendrá un encabezado con el nombre de los campos en el orden en el cual fueron exportados.
   5. Importar Correos Electrónicos: el usuario podrá importar los correos electrónicos previamente exportados en un archivo CSV con el formato del requerimiento §7.4.2.
   6. Administración de la Copia de Seguridad: el sistema permitirá al usuario administrar la información contenida en las copias de seguridad.
      1. Eliminar Correo de Copia de Seguridad: El sistema debe permitir al usuario eliminar cualquier correo electrónico de un archivo de correos exportados en CSV. El usuario deberá para tal efecto introducir el identificador del correo electrónico (§ 4.1.1).
      2. Modificar un Correo Electrónico: El sistema debe permitir al usuario modificar un correo electrónico directamente del archivo de correos ya exportados en formato CSV. El usuario deberá introducir el identificador del correo electrónico (§ 4.1.1).
   7. Exportar Correos Electrónicos en Formato Propietario:
      1. Por compatibilidad con un sistema heredado, el sistema debe además exportar los correos electrónicos en un formato de longitud variable utilizando campos de dimensión en lugar de delimitadores.
      2. El sistema también permitirá al usuario eliminar (§ 7.6.1) y modificar (§ 7.6.2) correos exportados en formato propietario directamente en el archivo exportado.
6. Módulo de Búsquedas Eficientes
   1. El sistema proveerá al usuario de diferentes mecanismos de búsqueda eficiente de tal forma que el usuario pueda encontrar cualquier correo electrónico en un tiempo corto.
   2. Búsqueda en Memoria
      1. El sistema debe proveer una opción al usuario para habilitar el modo de búsqueda en memoria.
      2. El sistema cargará en una estructura de datos eficiente en memoria principal (RAM) el contenido de todo el archivo de datos ordenados por el remitente.
      3. El usuario podrá hacer búsquedas por remitente como se especifica en el requerimiento §4.1.2, el sistema debe implementar una búsqueda eficiente en el archivo ya cargado en almacenamiento primario (RAM) sin consultar el almacenamiento secundario. El contenido del archivo no cambiará durante la ejecución.
   3. Búsqueda Rápida con Índices
      1. El sistema proveerá al usuario un mecanismo para realizar búsquedas utilizando índices:
         1. Índices Primarios: el usuario puede hacer una búsqueda indizada utilizando como llave primaria el identificador del correo electrónico (§4.1.1).
         2. Índices Secundarios: el usuario también tendrá la posibilidad de hacer una búsqueda indizada utilizando como llave secundaria el remitente del correo electrónico (§4.1.2) o el destinatario.
      2. Los índices se almacenarán en todo momento en memoria principal en árboles auto balanceados tipo AVL (Adelson-Velskii-Landis) para proveer un tiempo de acceso promedio a todas las entradas del índice.
      3. Los índices deben ser almacenados en archivos para evitar el costo de generar el índice cada vez que el sistema se inicia.
      4. El sistema debe implementar las siguientes operaciones sobre el índice:
7. Creación del índice a partir del archivo de datos.
8. Consulta del índice mediante llave (primaria o secundaria).
9. Inserción de un nuevo elemento en el índice.
10. Eliminación de un elemento en el índice.
11. Almacenamiento del índice en un archivo.
12. Carga de un índice desde un archivo a memoria.
    * + 1. Todas las operaciones sobre el índice: consulta, inserción y eliminación, se realizan en la estructura de datos en memoria.
        2. La inserción de registros se hace directamente al archivo de datos, y se deberá crear una entrada de índice para el nuevo registro; el índice se modifica en memoria y el archivo de índices queda desactualizado.
        3. La eliminación de registros se hace directamente al archivo de datos y en el índice primario inmediatamente, sin embargo, la eliminación en el índice secundario se postergará hasta que el usuario lo solicite.
        4. Por rendimiento el archivo de índices únicamente se actualizará al salir del programa. Si el programa termina anormalmente antes de terminar la actualización exitosamente el archivo de índices no será actualizado correctamente.
        5. Para resolver este problema implemente un mecanismo de tolerancia a fallas en su programa que al ser iniciado:
           1. Utilice el encabezado del archivo de índices como registro de bandera, el campo de llave con valor ‘0’ (ASCII) indica que el archivo de índices está actualizado, el valor ‘1’ (ASCII) indica que el archivo de índices está desactualizado.
           2. Detecte si el archivo de índices está desactualizado respecto a la información contenida en el archivo de datos.
           3. Reconstruya el índice a partir del archivo de datos, si éste se encuentra desactualizado.
      1. Índices Paginados: En caso de que el tamaño de los índices sea grande los requerimientos de memoria del sistema pueden incrementarse hasta causar problemas de rendimiento al usuario.
         1. Para evitar tener un impacto negativo en el rendimiento general del sistema, se debe utilizar un esquema de paginación de índices que deben estar implementados en árboles AVL con un búfer de 10 páginas.
         2. El algoritmo de reemplazo de páginas debe ser el menormente utilizado recientemente o LRU (Least Recently Used).
    1. Búsqueda Rápida con Dispersión
       1. El sistema debe permitir al usuario hacer una búsqueda rápida a través de una tabla de dispersión de 211 posiciones con la función de dispersión PJW.
       2. El método de resolución de colisiones utilizado debe ser con cubetas (listas ligadas).
       3. El usuario utilizará el remitente como llave para la búsqueda en la tabla de dispersión.
       4. Serialización:
          1. El sistema debe permitir al usuario almacenar la tabla de dispersión en un archivo binario para evitar crearla cada vez que el sistema se inicia.
          2. El sistema también debe soportar cargar una tabla de dispersión desde un archivo binario.
13. Cifrado/ Descifrado
    1. El sistema debe proveer un mecanismo para proteger el contenido de correos electrónicos que contengan información confidencial para el usuario.
    2. Cifrado de Archivos: el sistema debe soportar el cifrado de archivos mediante el Cifrado César.
    3. Descifrado de Archivos: el sistema debe soportar el descifrado de archivos cifrados con el Cifrado César.
    4. El usuario debe especificar una contraseña al momento de cifrar el archivo, la cual debe ser almacenada en el archivo cifrado y debe ser requerida al momento de descifrar el archivo.
14. Compresión de Datos
    1. El sistema debe permitir al usuario comprimir la copia de seguridad de los correos electrónicos de tal forma que estos ocupen menos espacio en disco.
    2. El sistema debe soportar el algoritmo de compresión RLE para la compresión de archivos de texto plano.

# ESPECIFICACIÓN DE REQUERIMIENTOS NO FUNCIONALES

1. El programa deberá contar con un menú para acceder a todas las operaciones.
2. El sistema debe estar desarrollado con una arquitectura orientada a objetos e implementado en un lenguaje orientado a objetos completo, como lo es C++.