

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ciencias y Sistemas  
Estructuras de Datos  
Sección A  
Ing. Edgar René Ornélyz  
Aux. Esvin Armando González Monzón



# Agenda Total

## Práctica 1 - Manejo de estructuras de datos simples

### Objetivo general

Con la realización de esta práctica se busca que el estudiante aplique e incremente sus conocimientos de programación con la creación de una solución de software que implemente estructuras de datos lineales como listas, pilas y/o colas para resolver problemas de gestión de la información.

### Objetivos específicos

Durante el desarrollo de esta práctica se requiere que el estudiante logre:

- Conocer y aplicar buenas prácticas de programación en el desarrollo de cualquier solución de software.
- Diseñar e implementar un sistema de carga masiva de registros con respaldo en archivos de texto físicos.
- Diseñar e implementar diferentes estructuras de datos haciendo uso óptimo de los recursos del computador.

### Descripción general

Para poder mantener organizados los contactos, lugares y eventos de un usuario común de PC existen varias aplicaciones, pero la empresa Toditos, S.A. desea crear la aplicación definitiva para mantener centralizada toda esta información, por lo que se le ha solicitado a usted, estudiante del 5to semestre del curso de Estructuras de Datos que elabore una aplicación utilizando el lenguaje de programación C++, en particular se desea una aplicación de escritorio.

Esta aplicación debe ser rápida, manejar altos volúmenes de información y hacer una gestión responsable de los recursos físicos del computador, por lo que una correcta implementación de sus propias estructuras de datos se hace algo fundamental para el éxito o el fracaso de esta labor. **TODAS las estructuras tendrán que trabajar con tipos genéricos.**

Se busca que, además de cumplir con los requerimientos funcionales, la aplicación cumpla con ciertos requisitos no funcionales presentes en cualquier sistema informático moderno, tales como ser intuitiva y amigable con el usuario, el correcto uso de la paleta de colores, una correcta gestión y presentación de errores, entre otros.



**Imagen 1:** Diagrama del funcionamiento general de la aplicación, se tendrán que gestionar contactos, eventos y lugares, con módulos dedicados a reportes y a importar/exportar la información a archivos de texto plano.

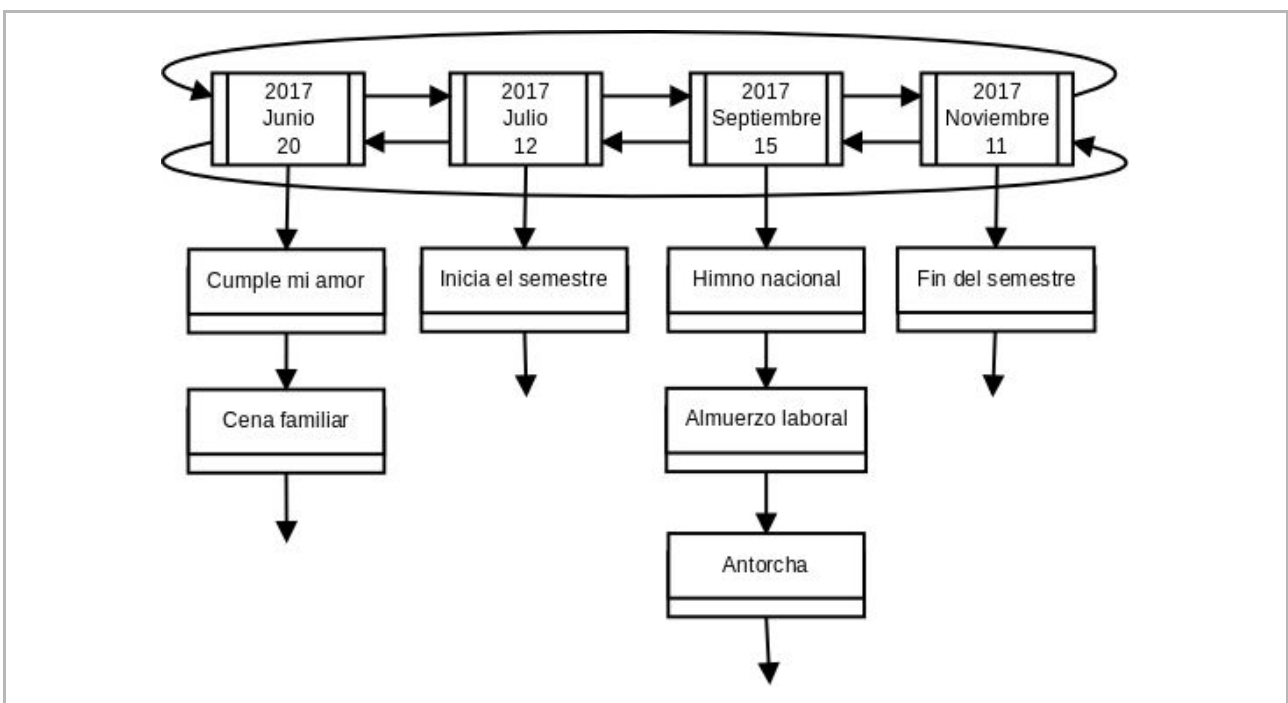
## Módulos del proyecto

De manera general la aplicación estará dividida en varios módulos que trabajarán de forma conjunta para cumplir los requisitos funcionales de la aplicación.

### Calendario

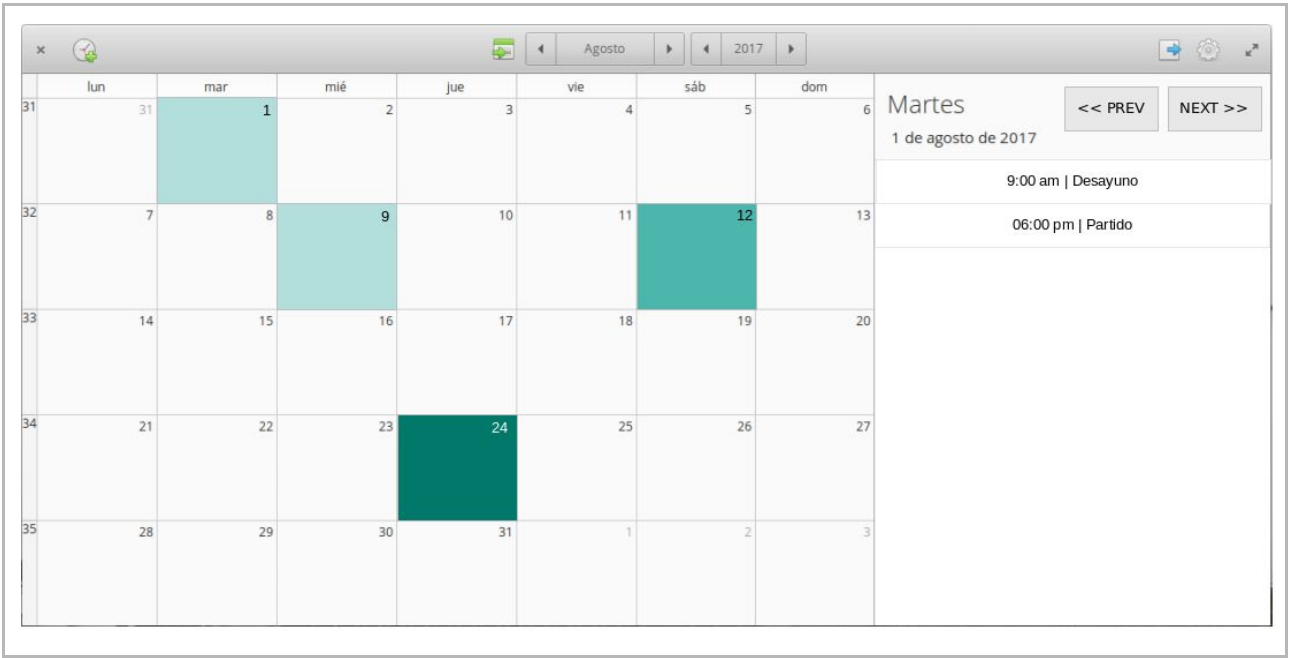
Este módulo permitirá al usuario gestionar los eventos que componen su agenda, la estructura que representará el calendario será una **lista doblemente enlazada circular** (lista de días) en donde cada nodo representará un día. Cada día tendrá vinculada una **lista simplemente enlazada ordenada** que almacenará los eventos, dicha lista deberá de estar ordenada con respecto a la hora de inicio de cada evento. Un evento tendrá los siguientes datos:

- Título
- Descripción
- Fecha (viene de la lista de días)
- Hora de inicio
- Duración
- Lugar de realización (ver [Lugares](#))
- Una lista de participantes (ver [Contactos](#))



**Imagen 2:** Lista doblemente enlazada circular que gestionará los días del calendario en donde cada nodo tendrá una lista simple que almacenará los eventos de la agenda.

Es necesario que esta lista cuente con operaciones básicas como agregar, modificar y eliminar eventos, así mismo su contenido se debe de poder visualizar en un calendario gráfico en donde [la tonalidad o el color](#) del día cambie según la cantidad de eventos programados para dicho día.



**Imagen 3:** Ejemplo de la interfaz deseada para el calendario (ventana principal) los días con un color más intenso indican que la cantidad de eventos en ellos es mayor a otros según lo indica la siguiente tabla.

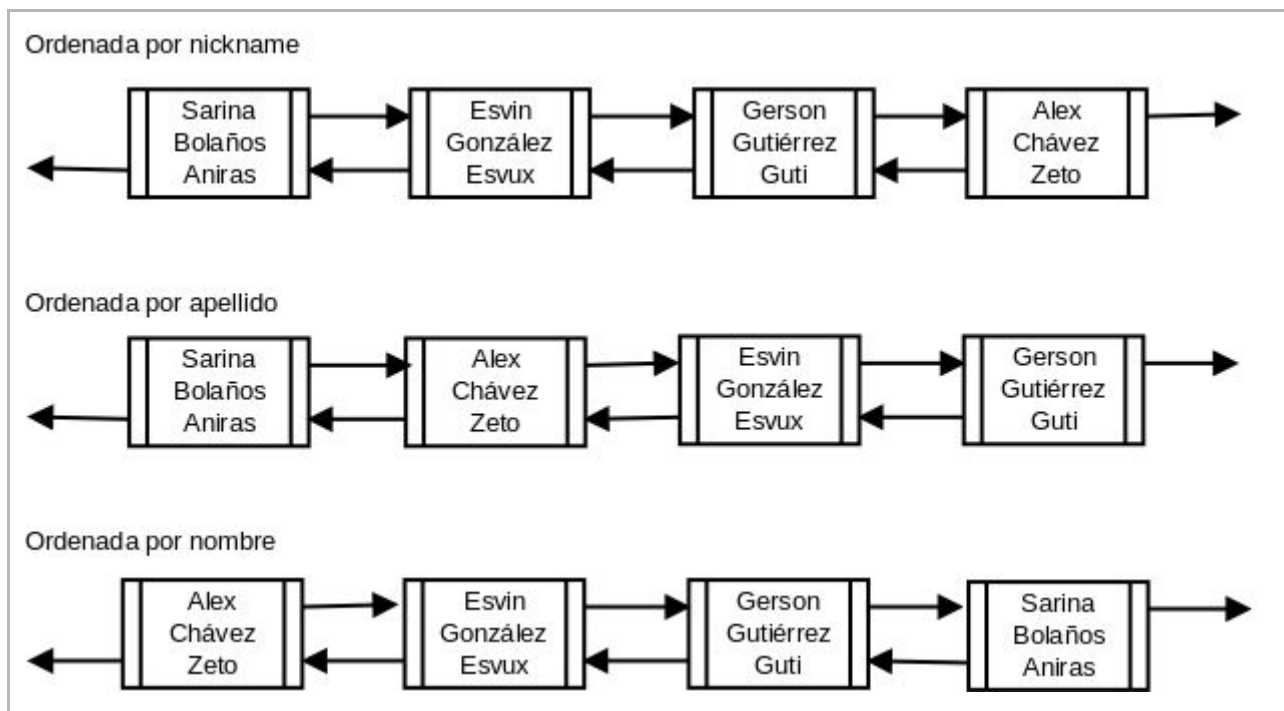
Eventos por día	Color
Ninguno	Transparente o base
Menos de 3	Claro
3 o menos de 5	Regular
5 o más	Fuerte

Para la navegación entre los eventos se tendrán dos controles principales **[NEXT]** y **[PREV]** que serán dos botones cuya función será ir al día siguiente (o anterior) de la lista de días (doble-circular), mostrando el listado de eventos vinculados al día en cuestión. Además existirá una vista de calendario que mostrará un panorama global de los eventos por mes. La navegación sobre por el calendario se podrá realizar cambiando de mes en mes o cambiando el año que se visualiza.

### Gestión de contactos

Los contactos representan personas con las que se organizan los eventos, éstos serán almacenados en una **lista doblemente enlazada ordenada**, la misma funcionará como un conjunto, ya que no permitirá insertar elementos con un nickname repetido (este elemento funcionará como identificador), esta lista será ordenable de acuerdo a tres criterios diferentes (por nombre, por apellido o por nickname) los cambios en el modo de ordenamiento se realizarán desde la aplicación. La lista de contactos permitirá las operaciones crear, editar, eliminar y ordenar. Cada contacto tendrá que contar como mínimo con la siguiente información:

- Correo electrónico
- Nombres
- Apellidos
- Nickname (alias)
- Edad
- Número de teléfono

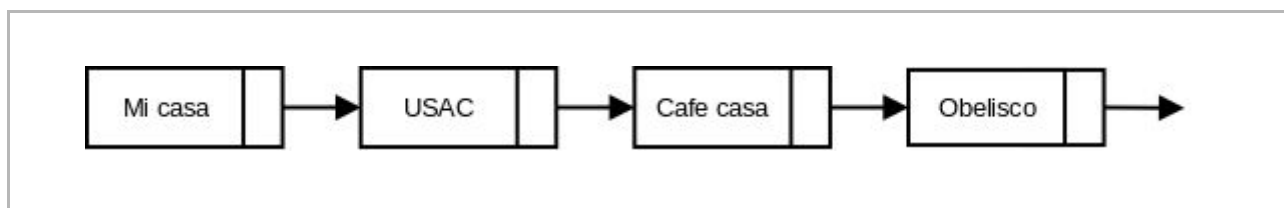


**Imagen 4:** Lista doblemente enlazada que gestionará los contactos del usuario con diferentes criterios de ordenamiento (nombres, apellidos o nickname).

## Gestión de lugares

Los lugares representan los sitios en donde se realizan los eventos, estos serán almacenados en una **lista simplemente enlazada**, sobre ésta estructura se podrán realizar operaciones de creación, edición y eliminación de elementos (en la ventana de lugares será necesario mostrar un mapa con la ubicación del lugar). Cada lugar debe tener como mínimo la siguiente información:

- Nombre
- Latitud (valor decimal)
- Longitud (valor decimal)
- Dirección (texto)



**Imagen 5:** Lista simplemente enlazada que almacenará los lugares.

## Carga masiva

Para agilizar los procesos de creación de información dentro de la plataforma creada se incorporará la función de importar o exportar la información de elementos (contactos, lugares y eventos) por medio de archivos en formato [JSON](#). Para esto se incorporan tres funciones adicionales al programa.

- **RESET:** Eliminará toda la información que se encuentra en la aplicación.
- **IMPORTAR:** Leerá un archivo en formato JSON y cargará los datos de dicho archivo en las diferentes estructuras de datos.
- **EXPORTAR:** Creará un archivo en formato JSON con la información que tienen las diversas estructuras de datos.

**NOTA:** Un ejemplo del formato JSON para los archivos a importar/exportar se encuentra en [este enlace](#).

## Reportes

Este módulo tendrá como objetivo la generación de dos reportes, **el primero** es un reporte gráfico que consiste en crear representaciones en imágenes del estado de las estructuras que almacenan los eventos, los contactos y los lugares (como ejemplo se pueden tomar las imágenes 2, 4 y 5 de este documento).


- Imagen de lista de todos los días junto con su respectiva lista de eventos
- Imagen de lista de contactos
- Imagen de lista de lugares








El segundo reporte consiste en comparar el rendimiento de las estructuras de datos creadas por el estudiante contra una de las estructuras de datos propias del lenguaje. Dicha comparación consistirá en realizar la inserción de 1,000, 50,000 y 1,000,000 registros tanto en las estructuras propias del estudiante como en las estructuras provistas por el lenguaje (por ejemplo un QList), en cada caso se deberá registrar los milisegundos en donde inicia el test y tener un registro de los milisegundos de finalizado el mismo, finalizado el test se proyectará un reporte de la siguiente forma (también podría ser una gráfica).

	Lista simple	Lista doble	Lista doble circular	Estructura del sistema
100 registros	100 ms	120ms	200ms	85ms
5k registros	2000 ms	2400 ms	2500 ms	900 ms
100k registros	10.1 s	12.0 s	20.5 s	2.4 s

## Funcionamiento del proyecto

Pantallas y funcionalidades esperadas en los diferentes módulos. Estos prototipos son una sugerencia del diseño que se pueda utilizar. La barra izquierda de todas las pantallas presentadas a continuación hace el papel de barra de navegación o control de pestañas.

<p><b>CALENDARIO (EVENTOS)</b></p> 	<p>Componente central: Calendario que muestra los días con colores de distinta intensidad según la cantidad de eventos por día. A la derecha se muestra una lista que contiene el detalle de cada evento según el día seleccionado en el calendario, dichos eventos podrán seleccionarse para su edición o eliminación.</p> <ul style="list-style-type: none"><li>➕ Muestra un formulario para agregar un nuevo evento</li><li>✎ Muestra un formulario para editar el evento seleccionado</li><li>✖ Elimina el evento seleccionado</li><li>📷 Genera la imagen de la lista de días + eventos por día</li></ul> <p><b>Extra:</b> Agregar controles para cambiar entre días <b>NEXT</b> y <b>PREV</b></p>
<p><b>CONTACTOS</b></p> 	<p>Componente central: Lista de contactos disponibles en el sistema, al seleccionar un contacto en específico su información debe pasar al formulario (celeste a la izquierda) para su edición.</p> <ul style="list-style-type: none"><li>➕ Crea un nuevo contacto con la información del formulario</li><li>✎ Edita el contacto selec. utilizando los datos del formulario</li><li>✖ Elimina el contacto seleccionado</li><li>📷 Genera la imagen de la lista de contactos</li></ul> <p><b>Extra:</b> Agregar un control para el ordenamiento de la lista</p>

<p style="text-align: center;"><b>LUGARES</b></p> 	<p>Componente central: Lista de lugares disponibles en el sistema, al seleccionar un lugar su información debe pasar al formulario (celeste a la izquierda) para su edición y se mostrará un mapa con la ubicación apuntada por los valores de latitud y longitud propios del lugar.</p> <ul style="list-style-type: none"> <li> Crea un nuevo lugar con la información del formulario</li> <li> Edita el lugar selec. utilizando los datos del formulario</li> <li> Elimina el lugar seleccionado</li> <li> Genera la imagen de la lista de lugares</li> </ul> <p><b>Extra:</b> No es necesario hacer el mapa sea interactivo (solo gráfico)</p>
<p style="text-align: center;"><b>UTILIDADES</b></p> 	<p>Componente central: Gráfica (o en su defecto tabla) con la comparación de las estructuras de datos descrita en la <a href="#">sección de reportes</a>. Además de unos controles para implementar la carga (importación) y exportación de los datos del programa.</p> <ul style="list-style-type: none"> <li> Muestra un cuadro de diálogo para abrir un archivo json y cargar la información que este contenga a la aplicación</li> <li> Muestra un cuadro de diálogo para guardar un archivo json cuyo contenido será la información que exista en ese momento dentro de la aplicación</li> </ul> <p><b>Extra:</b> Agregar un control (botón para la función de reset)</p>

## Consideraciones finales

Para el desarrollo de esta práctica se deben de tomar en cuenta los siguientes aspectos.

- La práctica debe ser realizada de manera individual.
- El lenguaje a utilizar será C++.
- El IDE a utilizar es libre (el estudiante puede elegir el IDE que desee, se recomienda QtCreator).
- La entrega será por medio de Dropbox, como una solicitud de archivo publicada por el auxiliar el día de entrega, no se recibirán prácticas fuera de tiempo o enviadas por correo electrónico.
- Los entregables de esta práctica consisten en:
  - Código fuente (sin dependencias rotas)
  - Archivo JAR de la aplicación (desde donde se realizará la calificación)
  - Manual técnico (descripción de las estructuras de datos realizadas por el estudiante)
- Para tener **derecho a calificación** es necesario que la solución creada cuente con:
  - Carga de archivo JSON
  - Reporte de rendimiento
  - Reportes gráficos
  - Estructuras de datos genéricas
- Todas las estructuras de datos deben ser desarrolladas por el estudiante (con excepción del reporte de rendimiento en donde se debe utilizar una estructura propia de C++).
- **Fecha de entrega: Martes 22 de agosto del 2017 antes de las 23:59 horas.**
- Copias totales o parciales serán penalizadas con nota de 0 puntos y reportadas ante escuela de ciencias y sistemas.