

## Objetivos

### Unidad 6: Estructuras y Algoritmos Recursivos

OE6.1 Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.

OE6.2 Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.

OE6.3 Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.

OE6.4 Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

## Enunciado

La [IV Copa Panamericana de Voleibol Masculino Sub-21](#) se celebrará del 5 al 11 de mayo de 2019 en Lima, (Perú) y la federación tiene la necesidad de registrar algunos datos de los espectadores que desean asistir al evento y por motivo de algunas labores de logística durante el evento, el personal de registro y control estará permanentemente realizando búsquedas de espectadores y participantes en el software, utilizando como criterio el id de registro del evento.

Usted ha sido elegido para desarrollar un programa que permita leer la información de los posibles asistentes al evento desde un archivo de texto delimitado por comas y cargar la información de manera que se puedan hacer búsquedas a través del identificador del registro del evento.

El programa debe permitir leer el archivo de texto con la información de todos los posibles participantes. Esta información se cargará en una estructura de datos de tipo árbol binario de búsqueda (ABB) cuya relación de orden estará dada por el id de registro del evento. Una vez cargada esta información en el árbol de espectadores, se seleccionará aleatoriamente el 50% de esos espectadores para agregarlos a una lista doblemente enlazada, que será la estructura de datos que almacenará los inscritos al evento o participantes al evento.

Luego de la lectura del archivo y la creación de las dos estructuras de datos: una para los posibles participantes y otra para los inscritos al evento, se debe permitir hacer búsquedas por id, para cada una de las estructuras creadas. Su programa debe indicar, en cualquier caso, si el espectador se encuentra o no entre los buscados así como el tiempo que tardó la búsqueda. Si el espectador fue encontrado, deben visualizarse sus datos en un panel a la izquierda.

Los datos de entrada al programa pueden ser generados utilizando la herramienta web <https://www.mockaroo.com> que permite generar datos aleatorios de forma personalizada. La configuración y orden de los datos generados debe ser la que se muestra a continuación:

Field Name	Type	Options
id	EIN	blank: 0 % fx ×
first_name	First Name	blank: 0 % fx ×
last_name	Last Name	blank: 0 % fx ×
email	Email Address	blank: 0 % fx ×
gender	Gender	blank: 0 % fx ×
country	Country	restrict countries... blank: 0 % fx ×
photo	Avatar	size: 50 × 50 format: random blank: 0 % fx ×
birthday	Date	5/10/1960 to 1/1/2016 in m/d/yyyy blank: 0 % fx ×

Se debe generar el máximo de filas posible para hacer buenas pruebas y el formato en que se debe exportar debe ser CSV el cual delimita los archivos de texto con coma por defecto.

# Rows:  Format:  Line Ending:  Include: ☒ header ☐ BOM

[Download Data](#) [Preview](#) [More](#) Want to save this for later? [Sign up for free.](#)

Ya que el máximo es 1000, realice pruebas del tiempo de la consulta para archivos de entrada mas grande como 100000, por ejemplo (generando 100 veces 1000 y uniéndolos). Compare los tiempos de las búsquedas cuando se realizan sobre todos los programadores y los participantes.

La interfaz gráfica del programa debe cumplir con los lineamientos que se muestran en el siguiente mockup:

IV Copa Panamericana de Voleibol Masculino Sub-21

Tab 1 Tab 2

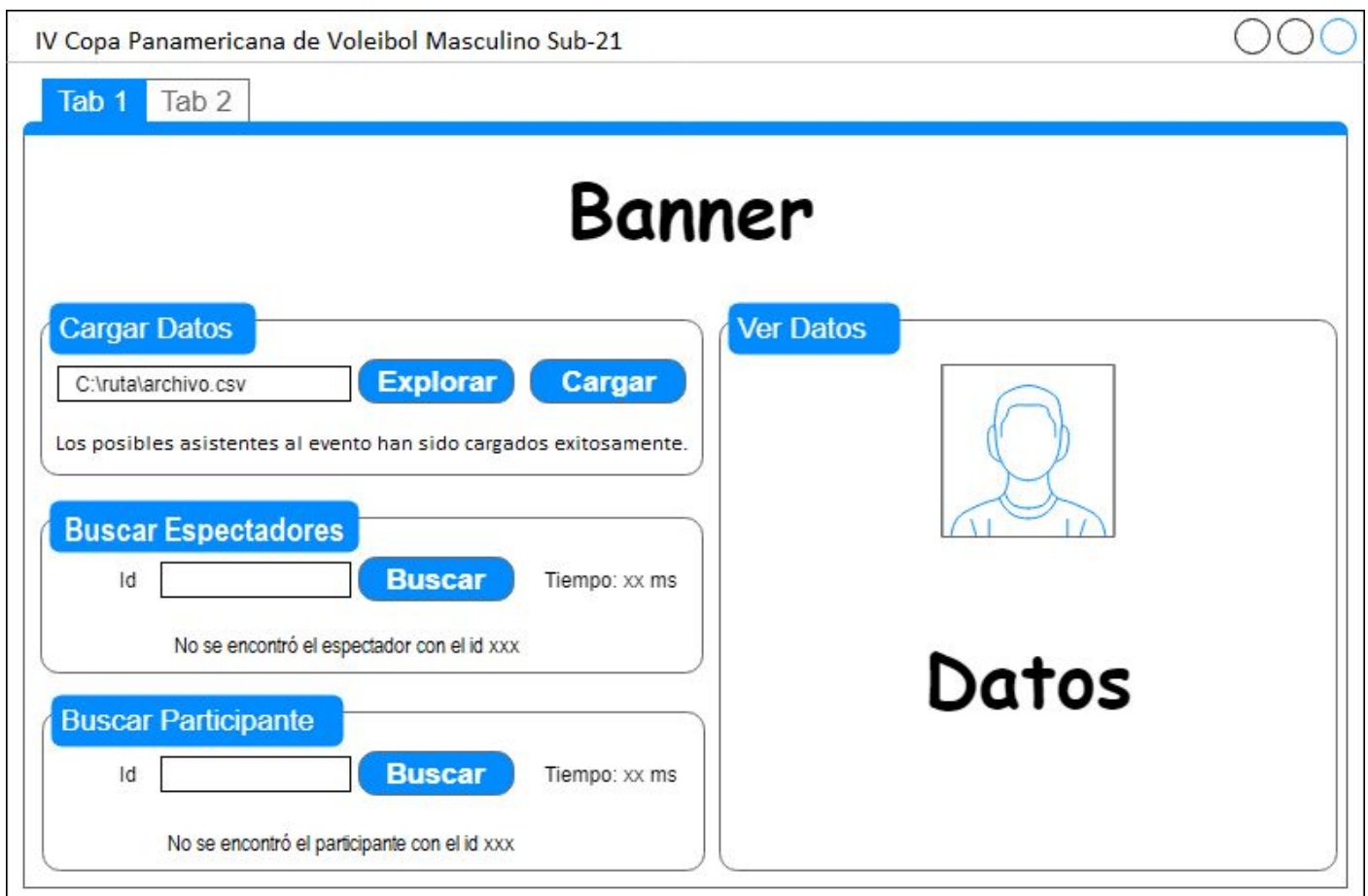
# Banner

**Cargar Datos**

**Explorar** **Cargar**

Los posibles asistentes al evento han sido cargados exitosamente.

**Ver Datos**



# Datos

**Buscar Espectadores**

Id  **Buscar** Tiempo: xx ms

No se encontró el espectador con el id xxx

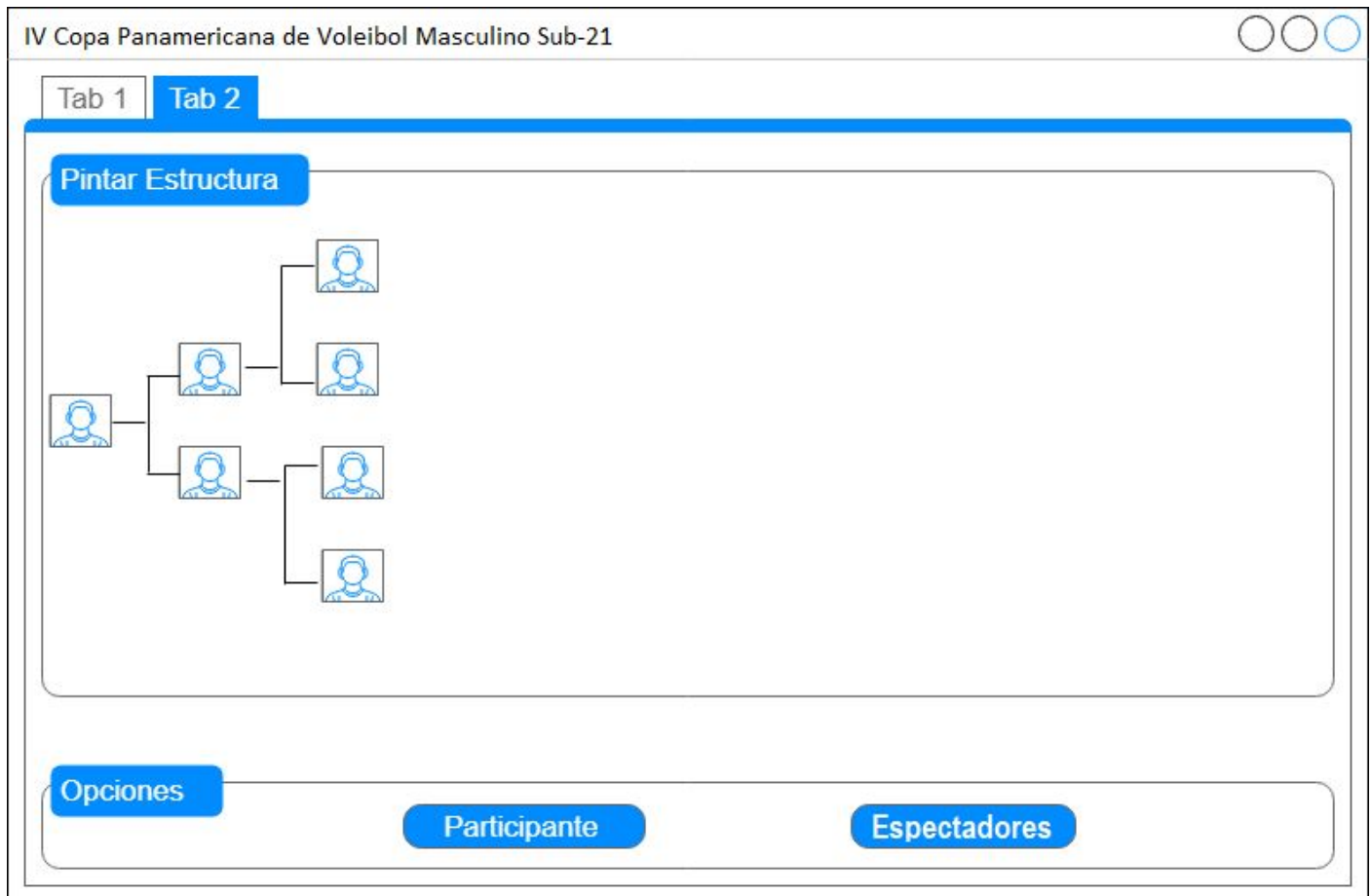
**Buscar Participante**

Id  **Buscar** Tiempo: xx ms

No se encontró el participante con el id xxx

Como sugerencia, puede usar `JFileChooser` el cual puede encontrar en el JDK de Java 8 en: <https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html>.

El programa también contará con una opción que podrá visualizar los datos de las dos estructuras de un país determinado, donde la estructura del árbol binario de búsqueda se verá como un árbol y la lista se visualizará en forma consecutiva. El ejemplo de la interfaz se presenta a continuación:



Para cada opción se visualizará sólo los datos del país seleccionado (puede ser el componente `JOptionPane` de Java), eso quiere decir, que para la estructura de espectadores el árbol pintado solo debe tener espectadores de la ciudad seleccionada e igual para la estructura de los participantes.

**Entregables.** 1. Requerimientos. 2. Diagrama de clases de modelo e interfaz (no generado automáticamente). 3. Implementación completa de todos los requerimientos en Java. 4. Diseño de casos de pruebas unitarias de todas las funcionalidades relacionadas con los árboles. 5. Implementación de todas las pruebas unitarias diseñadas y corriendo perfectamente. La estructura de los espectadores deben ser un árboles binarios de búsqueda y los participantes una lista doblemente enlazada, tal como se explicaron en el tema visto en clase y no podrán usar otro tipo de contenedor como array, `ArrayList`, etc. Otra restricción importante es que todos los métodos para manipular las dos estructuras (lista doblemente enlazada y árbol binario de búsqueda).

**Fecha de Entrega:** martes 21 de mayo de 2019 a las 11:55 p.m. a través de Moodle.

**Integrantes:** el laboratorio puede realizarse en parejas.