# Complejidad Temporal, Estructuras de Datos y Algoritmos

## *Enunciado Trabajo Final*

### Consideraciones Generales

El objetivo de este trabajo es integrar los contenidos vistos en la materia. El trabajo deberá realizarse en forma individual.

La fecha límite para su defensa se encuentra publicada en el aula virtual en el enlace: “Plan de trabajo”.

El trabajo se presentará junto con un informe que debe incluir:

- Datos del autor del trabajo final.
- Un diagrama de clases UML describiendo la estructura del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre los objetos.
- Detalles de implementación, problemas encontrados y como fueron solucionados, condiciones de ejecución, etc.
- Las imágenes de todas las pantallas que componen el sistema codificado junto con una descripción completa de las mismas.
- Ideas o sugerencias para mejorarlo o realizar una versión avanzada del mismo.
- Una breve conclusión o reflexión de la experiencia adquirida a partir de la realización del trabajo final.

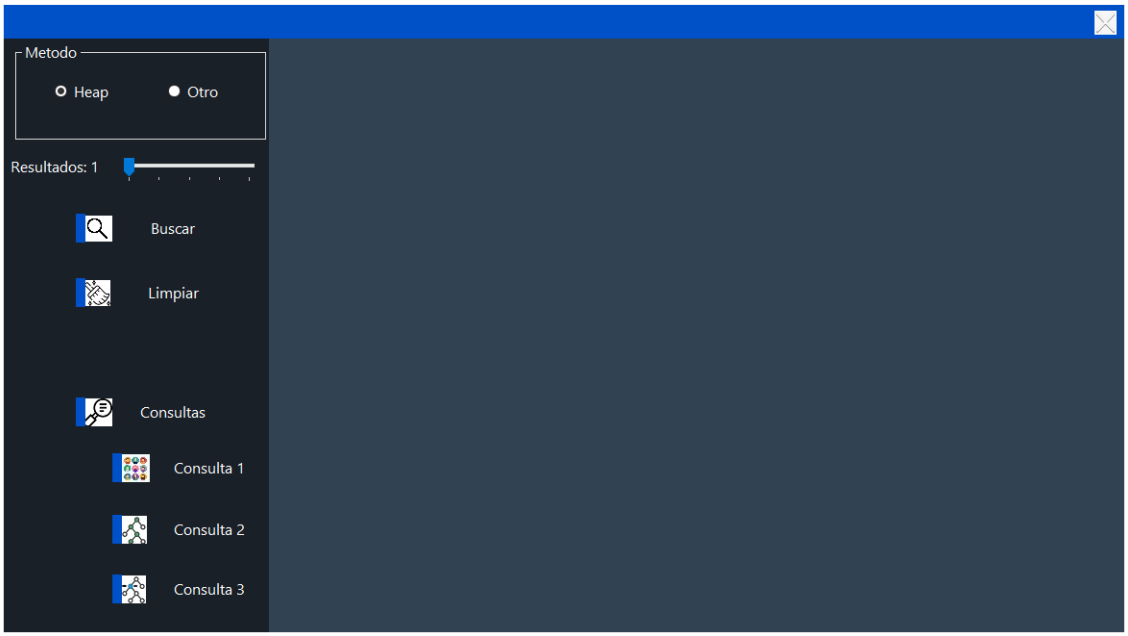
**Este informe se evaluará tanto en su contenido como en la forma en el que es presentado y tendrá una nota que afectará la nota final del trabajo.**

El trabajo deberá defenderse en un coloquio presencial.

## Enunciado

Una empresa informática está llevando a cabo un buscador de máximas ocurrencias que tiene por objetivo tomar conjuntos de datos almacenados en archivos *csv* y mostrarle al usuario cuales son los elementos que exhibe la mayor cantidad de apariciones en dichos *datasets*.

Al iniciar el aplicativo, éste solicita al usuario que seleccione el archivo *csv* donde se encuentran los recursos con los cuales se desea trabajar. Una vez seleccionado, la aplicación construirá una lista de *string* (`List<string>`) con los datos tomados del archivo *csv* para luego desplegar la siguiente vista a fin de realizar distintas búsquedas:




Para realizar una búsqueda es necesario:

1. Seleccionar el método con que se desea trabajar: usar una Heap o un método alternativo al primero.

2. Indicar la cantidad de resultados a mostrar utilizando la barra deslizante "*Resultados*".

3. Activar el botón "*Buscar*" para realizar efectivamente la búsqueda.

La empresa informática lo ha contratado a Ud. para completar el aplicativos antes descrito, su función será la de implementar los siguientes métodos pertenecientes a la clase **Estrategia**:

**1.** **BuscarConHeap**(List<string> datos, int cantidad, List<Dato> collected): Retorna en la variable ***collected*** los primeros elementos con mayor número de ocurrencias de la lista ***datos*** utilizando una Heap como estructura de datos soporte. El número de elementos a retornar es indicado por el parámetro ***cantidad***.

**2.** **BuscarConOtro**(List<string> datos, int cantidad, List<Dato> collected): Tiene la misma funcionalidad del método **BuscarConHeap()** pero debe implementarse utilizando un método alternativo ( de su preferencia) al uso de una Heap.

**3.** **Consulta1** (List<string> datos): Retorna un texto con los tiempos que insumen los métodos **BuscarConHeap()** y **BuscarConOtro()** en realizar la búsqueda de los 5 elementos de con mayor cantidad de ocurrencias.

4. **Consulta2** (List<string> datos): Retorna un texto con el camino a la hoja más izquierda de la Heap que se construye a partir de los datos de entrada cuando se utiliza el método **BuscarConHeap()**.

5. **Consulta3** (List<string> datos): Retorna un texto que contiene los datos de la Heap que se construye a partir de los datos de entrada cuando se utiliza el método **BuscarConHeap()**, explicitando en el texto resultado los niveles en los que se encuentran ubicados cada uno de los datos.

La cátedra proveerá métodos y clases utilitarias a fin de simplificar la construcción y prueba del aplicativo. A continuación, se describen los métodos más importantes y a que clase pertenecen:

| **Clase** | **Métodos** |
|---|---|
| **Dato** | **- Dato(int ocurrencias, string texto)**: Construye un objeto que almacena un texto y la cantidad de ocurrencias del mismo.<br>**- ToString():** Devuelve un texto que contiene el dato almacenado en formato *string* . |