# Proyecto II – Text Finder

Tecnológico de Costa Rica
Área de Ingeniería en Computadores
CE-1103 Algoritmos y Estructuras de Datos I
Segundo Semestre 2022
Valor 20%

---

## Objetivo General

➔ Diseñar e implementar una aplicación para realizar búsquedas de texto en documentos.

## Objetivos Específicos

➔ Implementar listas enlazadas y algunas variaciones
➔ Implementar árboles y algoritmos de ordenamiento
➔ Investigar y desarrollar una aplicación en el lenguaje de programación Java
➔ Investigar acerca de programación orientada a objetos en Java.
➔ Aplicar **patrones de diseño** en la solución de un problema.
➔ Utilizar UML para modelar la solución de un problema.

## Descripción del Problema

Text Finder es una aplicación de escritorio escrita en Java que permite buscar texto en archivos .txt, .pdf y .docx. Una vez que la aplicación ha indizado los documentos, el usuario puede realizar búsqueda de texto. La aplicación muestra los resultados de la búsqueda en todos los documentos y muestra al usuario extractos del documento donde aparece la palabra. El usuario puede ordenar los resultados por Nombre del archivo (Quicksort), fecha de creación (bubblesort) y tamaño-cantidad de palabras (Radixsort)

Las características principales de la aplicación son las siguientes:

1. **Administración de la biblioteca de documentos.** Desde acá el usuario podría:
    a. Agregar documentos a la biblioteca.
    b. Puede especificar archivos individuales o carpetas con documentos. De igual forma puede eliminar o actualizar documentos previamente agregados a la biblioteca.
2. **Indización de la biblioteca.** La aplicación provee una opción para ejecutar la indización de los documentos. La indización consiste en lo siguiente:
    a. Se parsea cada documento en la biblioteca.
    b. Por cada palabra del documento, se inserta en un árbol binario de búsqueda y un AVL. Cada nodo del árbol tiene dos elementos: la palabra y una lista de ocurrencias. La lista de ocurrencias debe guardar el documento en el que aparece y alguna posición dentro del documento, de forma tal que cuando se busque la palabra, la aplicación pueda mostrar la palabra junto con texto alrededor.
3. **Búsquedas de texto.** Esto involucra:
    a. El usuario puede ingresar un texto para buscar. Puede ingresar una frase y seleccionar buscar cada palabra individual o en frase

    b. La aplicación muestra la palabra o la frase resaltada de un color junto con el texto alrededor. Muestra todas las apariciones en todos los documentos indizados.
    c. Para el resultado debe mostrar cuántas comparaciones se realizaron usando el índice del Árbol Binario de Búsqueda y cuántas comparaciones usando el índice del Árbol AVL.
4. **Abrir documento.** El usuario puede abrir el documento en la posición donde aparecen las ocurrencias desde la aplicación.

Este proyecto implica una arquitectura cliente-servidor, donde la comunicación debe hacerse mediante sockets. El cliente es solamente una interfaz donde el usuario interactúa con el sistema, mientras que el servidor es el encargado de la gestión de las estructuras de datos, los algoritmos y el procesamiento de las solicitudes de los usuarios. El cliente solo captura los requerimientos de los usuarios y muestra el resultado de las solicitudes.

**Documentación requerida**

1. Internamente, el código se debe documentar utilizando Javadoc. Se debe generar el HTML.
2. La documentación externa se hará en un documento que incluya lo siguiente (deberá entregarse un PDF):
    a. Breve descripción del problema.
    b. Diagrama de clases.
    c. Descripción de las estructuras de datos desarrolladas.
    d. Descripción de los algoritmos desarrollados.
    e. Problemas encontrados en forma de bugs de *github*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
3. **Planificación y administración del proyecto:** se utilizará Azure DevOps para la administración del proyecto. Debe incluir:
    a. Lista de features e historias de usuario identificados de la especificación.
    b. Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
    c. Descomposición de cada user story en tareas.
    d. Asignación de tareas entre los integrantes del grupo.

**Aspectos operativos y evaluación**:

1. **Fecha de entrega: De acuerdo con el cronograma del curso y lo establecido en el TEC Digital**
2. El proyecto tiene un valor de 20% de la nota del curso.
3. El trabajo es **en parejas.**
4. Deben entregar en el TEC Digital un documento con el link del repositorio de GitHub, Azure DevOps y el PDF de la documentación. Para ambas herramientas deben dar acceso al correo del profesor.
5. Es obligatorio utilizar un Git y GitHUB para el control de versiones del código fuente y evidenciar el uso de Commits frecuentes.
6. Es obligatorio integrar toda la solución.
7. El código tendrá un valor total de 70%, la documentación externa 20% y la defensa un 10%.
8. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.

9. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado. Se recomienda que realicen la documentación conforme se implementa el código.
10. La nota de la documentación externa es proporcional a la completitud del proyecto.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la primera cita de revisión oficial.
13. Aún cuando el código y la documentación externa tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación externa, automáticamente se obtiene una nota de cero en la nota final del proyecto.
    b. Si no se utiliza un manejador de código se obtiene una nota de cero en la nota final del proyecto.
    c. Si la documentación externa no se entrega en la fecha indicada se obtiene una nota de cero en la nota final del proyecto.
    d. Sí el código no compila se obtendrá una nota de cero en la nota final del proyecto, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en Java (Windows), en caso contrario se obtendrá una nota de cero en la nota final del proyecto.
    f. Si alguna persona integrante del proyecto no se presenta a la revisión se asignará una nota de cero en la nota final del proyecto.
14. La revisión de la documentación podrá ser realizada por parte del profesora antes, durante o después de la revisión del proyecto.
15. Cada estudiante tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
16. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
17. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión al profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.