Introducción

El presente manual técnico describe el diseño, implementación y funcionamiento de un sistema desarrollado en **JavaScript** que tiene como objetivo analizar archivos de entrada con información relacionada a torneos deportivos, procesar los datos y generar diversos reportes que permitan entender el desempeño de equipos, jugadores y la evolución general del torneo.

En el ámbito de la informática, el análisis de datos provenientes de archivos estructurados es una tarea fundamental. Muchos sistemas, tanto académicos como industriales, requieren leer archivos con reglas específicas y transformarlos en información útil y presentable. En este caso particular, se busca aplicar los fundamentos del **análisis léxico** para convertir un texto plano en una secuencia de **tokens** significativos, que posteriormente alimentan la generación de reportes dinámicos.

Este proyecto tiene además un objetivo pedagógico: enseñar cómo funcionan los **autómatas finitos deterministas (AFD)** y la construcción de analizadores léxicos sin recurrir a herramientas externas. Se hace especial énfasis en cumplir con la restricción de no utilizar librerías de parsing, expresiones regulares avanzadas, ni generadores de analizadores, lo cual garantiza que el estudiante comprenda el proceso de construcción de un **scanner manual** que recorra carácter por carácter el archivo de entrada.

La importancia de un sistema como este radica en que simula la labor que realizan compiladores, intérpretes y sistemas de procesamiento de datos en general. La diferencia es que aquí se aplica a un contexto cercano y práctico: la organización de un torneo deportivo. De esta manera, se puede comprender cómo conceptos teóricos de lenguajes formales y autómatas se aplican en problemas reales.

Además, el proyecto contribuye al desarrollo de competencias en varias áreas:

- **Programación estructurada y orientada a eventos.**
- **Gestión de archivos y entrada/salida de datos.**
- **Diseño de interfaces gráficas dinámicas en la web.**
- **Generación y presentación de reportes con información procesada.**
- **Manejo de errores y robustez en sistemas informáticos.**

Con ello, el sistema no solo cumple un objetivo académico, sino que también constituye una base para proyectos más grandes relacionados con el análisis de datos, el desarrollo de sistemas administrativos y la construcción de software de gestión en entornos deportivos o educativos.

Objetivos

ObjetivoGeneral

Desarrollar un sistema que realice el análisis léxico de un archivo de torneo, detectando tokens válidos y errores, y genere reportes detallados sobre equipos, jugadores y estadísticas del evento deportivo.

Objetivos Específicos

1. Implementar un **scanner manual** en JavaScript, basado en un autómata, para identificar tokens en el archivo de entrada.
2. Diseñar una interfaz gráfica sencilla que permita cargar archivos, mostrar tokens y errores, y generar reportes dinámicos.
3. Elaborar diferentes reportes: **bracket de eliminación, estadísticas por equipo, tabla de goleadores e información general del torneo**.
4. Garantizar el manejo de errores léxicos y de procesamiento, de manera que el sistema sea tolerante a fallos.
5. Cumplir con las restricciones académicas: no usar expresiones regulares ni librerías de análisis léxico o sintáctico.

Alcances

- Permite analizar un archivo con información de torneos siguiendo una sintaxis predefinida.
- Genera tokens válidos y registra errores léxicos.
- Produce cuatro reportes dinámicos que pueden visualizarse en la interfaz web.
- Se desarrolla íntegramente en **JavaScript nativo** (sin librerías prohibidas).
- El código está documentado y modularizado para futuras ampliaciones.

Limitaciones

- Solo reconoce la sintaxis definida en el archivo de entrada (no es flexible a otros formatos).
- No realiza análisis sintáctico profundo, únicamente léxico.
- El sistema funciona en navegadores modernos, no es una aplicación de escritorio.
- No persiste la información en bases de datos; los reportes existen únicamente en memoria.

Desarrollo Técnico

- Flujo General del Sistema

1. El usuario selecciona un archivo desde la interfaz.
2. El sistema lo lee con FileReader.
3. El **scanner** procesa el texto carácter por carácter y devuelve dos listas: tokens y errores.
4. Se muestran tablas con los resultados iniciales.
5. El usuario puede generar los cuatro reportes a partir de los tokens.
6. Finalmente, los reportes se muestran en tablas HTML dinámicas.

- **El Scanner (Análisis Léxico)**

El Scanner implementa un **autómata de estados finitos** con la siguiente lógica:

- **Estado inicial**: comienza leyendo carácter por carácter.
- **Construcción de lexemas**:
  - Si el carácter es letra → se construye una palabra (puede ser reservada o identificador).
  - Si el carácter es número → se construye un número entero.
  - Si encuentra comillas → se abre una cadena de texto.
  - Si encuentra símbolos como {, }, :, ;, - → se reconocen como tokens únicos.
- **Finalización de lexemas**: cuando cambia la categoría de los caracteres, se cierra el token.
- **Errores léxicos**: cualquier carácter fuera del alfabeto esperado se almacena en una tabla de errores con línea y columna.

Ejemplo:
Entrada:

partido "Real Madrid" vs "Barcelona" resultado "2-1"

Tokens:

- palabra reservada: partido
- cadena: Real Madrid
- símbolo: vs
- cadena: Barcelona
- palabra reservada: resultado
- cadena: 2-1

-Módulo Principal

El archivo principal (script.js) contiene la lógica para:

- Cargar archivos.
- Invocar al scanner.
- Mostrar tokens y errores.
- Generar reportes.

Cada reporte se construye recorriendo la lista de tokens y extrayendo la información necesaria.

-Reportes Generados

1. **Reporte de Bracket de Eliminación**
    - Muestra partidos por fase (cuartos, semifinal, final).
    - Indica equipos enfrentados, resultado y ganador.
2. **Reporte de Estadísticas por Equipo**
    - Partidos jugados, ganados, perdidos.
    - Goles a favor, en contra y diferencia de goles.
3. **Reporte de Goleadores**
    - Lista de jugadores.
    - Número de goles anotados.
    - Minutos en los que anotaron.
4. **Reporte de Información General**
    - Nombre y sede del torneo.
    - Número de equipos participantes.
    - Total de partidos y partidos completados.
    - Promedio de goles.

    - Promedio de edad de jugadores.
    - Fase actual.

Pruebas y Validación

Para garantizar el correcto funcionamiento del sistema se realizaron diversas pruebas:

- **Prueba de carga de archivo correcto**: se validó que se reconocieran tokens y se generaran reportes completos.
- **Prueba con errores léxicos**: se introdujeron caracteres inválidos y se verificó que fueran detectados.
- **Prueba de reportes**: se confirmó que los cálculos de goles, partidos y estadísticas fueran correctos.
- **Prueba de robustez**: se cargaron archivos incompletos para validar el manejo de errores.

Los resultados confirmaron que el sistema cumple con los requerimientos establecidos.

Conclusión

El desarrollo de este analizador léxico y generador de reportes de torneo permitió aplicar en un caso práctico los conocimientos de **lenguajes formales y autómatas**, así como conceptos de **programación en JavaScript** orientada a la manipulación de archivos y la generación dinámica de interfaces web.

El sistema se diseñó respetando la restricción de **no utilizar librerías de análisis ni expresiones regulares complejas**, lo cual garantizó un aprendizaje profundo del proceso de construcción de un **scanner manual** y del manejo detallado de cada carácter del archivo de entrada.

Entre las principales ventajas del sistema se destacan:

- Su **robustez** al detectar errores léxicos.
- La **claridad** de sus reportes, que permiten al usuario visualizar de forma organizada la información del torneo.
- La **modularidad** del código, que facilita futuras ampliaciones.

No obstante, también se identifican ciertas limitaciones, como la falta de un análisis sintáctico profundo y la dependencia de un formato de entrada específico. Estas limitaciones abren la puerta a mejoras futuras, como la implementación de un analizador sintáctico manual, la integración de persistencia de datos o incluso la visualización gráfica del bracket en formato de diagrama.

En conclusión, este proyecto constituye un aporte significativo en la formación académica, pues combina teoría y práctica en un mismo ejercicio: desde la teoría de autómatas hasta la implementación de sistemas útiles en un contexto real. Con ello, el estudiante desarrolla no solo habilidades técnicas, sino también competencias en diseño de software, análisis de problemas y comunicación de resultados.