Universidad San Carlos de Guatemala

Facultad de ingeniería

Ingeniería en Ciencias y Sistemas

Ing. Otto Amílcar Rodríguez Costa

Aux. Danny Hugo Bryan Tejaxún Pichiyá

Manual técnico

Proyecto 1

Josue Geovany Yahir Perez Avendaño

202300814

21/09/2025

Arquitectura del Sistema

Patrón Arquitectónico

El sistema utiliza una arquitectura **MVC (Modelo-Vista-Controlador)** adaptada para JavaScript vanilla:

- **Modelo**: Clases de dominio (Tournament, Team, Player, Match)
- **Vista**: Interfaz HTML/CSS con manipulación directa del DOM
- **Controlador**: TourneyInterface que orquesta el análisis y la presentación

Análisis Léxico

Autómata Finito Determinista (AFD)

El analizador léxico implementa un AFD manual que procesa el archivo carácter por carácter.

Estados del Autómata:

- Estado 0: INICIAL
- Estado 1: IDENTIFICADOR
- Estado 2: NÚMERO
- Estado 3: CADENA
- Estado 4: SÍMBOLO
- Estado 5: ERROR

Implementacion del Scanner

```
class TourneyScanner {
  #input;         // Texto de entrada
  #position;      // Posición actual
  #line;          // Línea actual
  #column;        // Columna actual
  #currentChar;   // Carácter actual
  #tokens;        // Array de tokens generados
  #reservedWords;  // Mapa de palabras reservadas
```

```
  scan() {
    while (this.#currentChar !== null) {
      this.#skipWhitespace();

      if (this.#isLetter(this.#currentChar)) {
        this.#readIdentifier();
      } else if (this.#isDigit(this.#currentChar)) {
        this.#readNumber();
      } else if (this.#currentChar === '"') {
        this.#readString();
      } else {
        this.#readSymbol();
      }
    }
  }
}
```

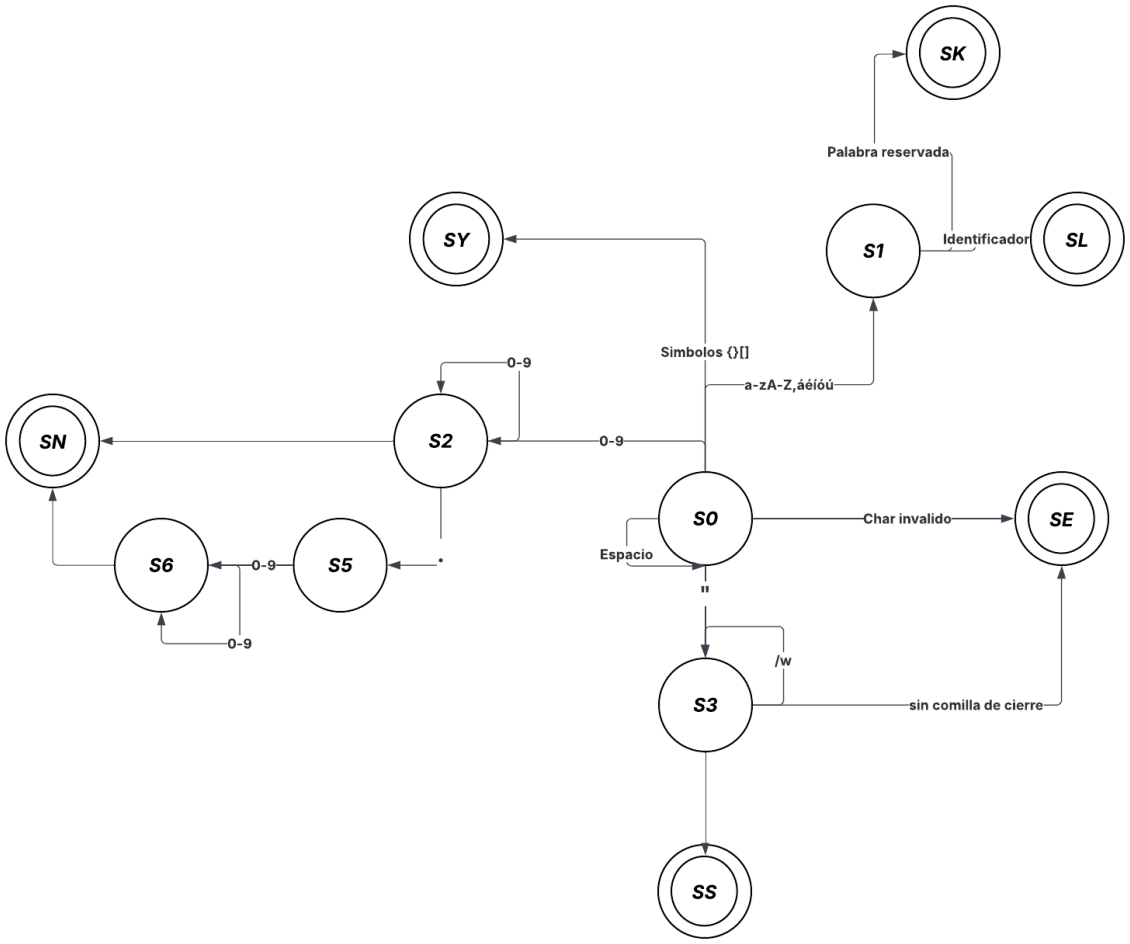
Tokens Reconocidos

Elemento	Descripción	Estructura y Valores Permitidos	Ejemplo
TORNEO	Define la información general del torneo de eliminación	Debe comenzar con TORNEO { y terminar con }. Debe incluir nombre y cantidad de equipos	TORNEO { nombre: "Copa Mundial", equipos: 8 }
EQUIPOS	Define la sección donde se listarán los equipos participantes	Debe comenzar con EQUIPOS { y terminar con }	EQUIPOS { equipo: ... }
equipo	Define un equipo dentro de la sección EQUIPOS	equipo: "Nombre" [jugadores: ...] donde se listan todos los jugadores	equipo: "Brasil" [jugador: "Pelé", jugador: "Ronaldinho"]
jugador	Define un jugador dentro de un equipo	jugador: "Nombre" [posicion: "POSICION", numero: NUMERO, edad: EDAD]	jugador: "Messi" [posicion: "DELANTERO", numero: 10, edad: 35]
ELIMINACION	Define la estructura del torneo de eliminación directa	Debe comenzar con ELIMINACION { y contener las fases en orden	ELIMINACION { cuartos: {...}, semifinal: {...}, final: {...} }
fase	Define una fase específica del torneo	fase: [partidos de esa fase]	cuartos: [partido: ..., partido: ...]
partido	Define un partido entre dos equipos con resultado	partido: "Equipo1" vs "Equipo2" [resultado: "X-Y", goleadores: [...]]	partido: "Brasil" vs "Argentina" [resultado: "2-1", goleadores: ["Pelé", "Messi"]]
goleador	Define quién anotó goles en un partido	goleador: "NombreJugador" [minuto: X]	goleador: "Pelé" [minuto: 23]
Posiciones	Valores permitidos para posiciones de jugadores	"PORTERO", "DEFENSA", "MEDIOCAMPO", "DELANTERO"	posicion: "DELANTERO"

Algoritmo de Análisis

El parser implementa **análisis sintáctico descendente recursivo** con las siguientes características:

- **Predictivo**: Usa lookahead para determinar producciones
- **Recursivo**: Cada símbolo no terminal tiene su método
- **Con recuperación de errores**: Continúa el análisis tras encontrar errores



Implementación del AFD en TourneyJS

1. Estructura del Autómata Finito Determinista

El AFD se implementó manualmente en TourneyScanner.js procesando carácter por carácter:

Estados principales: - Estado inicial: Lectura de caracteres - Estado identificador: Reconocimiento de palabras reservadas/identificadores - Estado número: Reconocimiento de valores numéricos - Estado cadena: Reconocimiento de strings entre comillas - Estado símbolo: Reconocimiento de delimitadores

2. Transiciones de Estados

Desde estado inicial:

- isLetter() → Estado identificador
- isDigit() → Estado número

- " → Estado cadena
- {, }, [,], :, ,, ; → Estado símbolo
- isWhitespace() → Permanecer en estado inicial
- Carácter inválido → Estado error

3. Reconocimiento de Tokens

Estado identificador:

- Procesa secuencias alfanuméricas
- Consulta tabla de palabras reservadas (Map con tokens como TORNEO, EQUIPOS, etc.)
- Si no es palabra reservada → TK_identificador

Estado número:

- Procesa dígitos consecutivos
- Maneja números decimales con punto
- Genera token TK_numero

Estado cadena:

- Procesa contenido entre comillas
- Maneja escape de comillas (\")
- Genera token TK_cadena

4. Manejo de Errores Léxicos

El AFD detecta errores mediante:

- Caracteres no reconocidos (símbolos inválidos como @, #, %, etc.)
- Cadenas sin cerrar (EOF antes de ")
- Caracteres específicamente problemáticos

5. Integración con el Parser

El scanner genera una lista de tokens que el parser consume:

- next_token(): Avanza al siguiente token

- look_ahead(): Examina token actual sin consumir
- El parser implementa recuperación de errores sintácticos

6. Tolerancia a Errores

Scanner: Continúa procesando después de encontrar errores léxicos **Parser:** Implementa sincronización para recuperarse de errores sintácticos, incluyendo tolerancia al punto y coma mediante skipOptionalSemicolons()

Esta implementación sigue los principios clásicos de construcción de compiladores, donde el AFD procesa la entrada secuencialmente y utiliza una tabla de transiciones implícita (mediante estructuras condicionales) para reconocer los diferentes tipos de tokens del lenguaje de torneos deportivos.