Manual Técnico

Sistema de ANALIZADOR de torneos

Ricardo Emanuel Sagui Miranda

26 de septiembre de 2025

Índice general

Introducción

Este **Manual Técnico** documenta la estructura interna del sistema de *Analizador de torneo*. Su propósito es describir los archivos de código, explicar su función en el sistema y mostrar el contenido de cada uno para referencia técnica.

Capítulo 1

Requerimientos

1.1. Software

- Node.js v18 o superior.
- Editor de texto o IDE (Visual Studio Code recomendado).

1.2. Hardware

- CPU: Intel i3 o superior.
- RAM: mínimo 4 GB.
- Espacio en disco: 200 MB libres.

Capítulo 2

Archivo Equipo.js

Este archivo define la clase `Cliente`, que representa a cada cliente registrado en el sistema, con sus respectivos atributos (ID y nombre) y sus métodos *getters* y *setters*.

Código

```
export class Equipo{
    constructor(nombreEquipo){

        this.nombreEquipo=nombreEquipo;

        this.jugadoresDelEquipo=[];

        //datos basicos de los equipos:

        this.cantPartidosJugados=0;
        this.victorias=0;
        this.derrotas=0;
        this.golesAFavor=0;
        this.golesEnContra=0;
        this.faseAlcanzada=null;// "Cuartos", "Semifinal", "Final
           "

    }

    agregarUnJugadorAlEquipo(jugador){ //un objeto jugador
       claramente
        this.jugadoresDelEquipo.push(jugador)

    }
}
```

Capítulo 3

Archivo Gol.js

Este archivo implementa la clase `gol`, que modela un objeto que nos ayudara a describir una lista de posibles anotaciones que tengan los jugadores

Código

```
export class Gol{
    constructor(jugadorNombre,equipoDelJugador,minutoGol){
        this.jugadorNombre=jugadorNombre; //el que metio el gol
        this.equipoDelJugador=equipoDelJugador;
        this.minutoGol=minutoGol;
    }


}
```

Capítulo 4

Archivo Jugador.js

Los que describen a base de listas, forman los equipos...Estos tienen atributos que describle sus anotaciones, minutos de gol, posicion, equipos asociado.

Código

```
export class Jugador{
    constructor(nombre, posicion, numero, edad){

        this.nombre=nombre;
        this.posicion=posicion;
        this.numero=Number(numero);
        this.edad=Number(edad);
        this.cantidadGoles=0; //si meten gol en algun juego se
           deberia actualizar
        this.minutosDeGol=[] // [15,45] metio gol en el 15 y en
           el 45

    }
}
```

Capítulo 5

Archivo Partido.js

Un partido relaciona 2 equipos, define un ganador, la fase que representa y los goles anotados.

Código

```
export class Partido{
    constructor(
        fase,
        equipoLocal,
        equipoVisitante,
        golesEquipoLocal, //es el equipo local
        golesEquipoVisitante//es el equipo visitante
    ){

        this.fase=fase;
        this.equipoLocal=equipoLocal;
        this.equipoVisitante=equipoVisitante;
        this.golesEquipoLocal=Number(golesEquipoLocal);
        this.golesEquipoVisitante=Number(golesEquipoVisitante);
        this.infoGoles=[];//una lista que almcaena objetos gol
        //Los objetos gol me dicen->el que marco "jugador", su
           equipo y el minuto

        this.pendiente=false //pueden haber partidos por jugar
    }

    agregarInfoGol(infoGol){ //aca se mete un objeto gol
        this.infoGoles.push(infoGol);

    }


}
```

Capítulo 6

Archivo Torneo.js

Es el orquestador principal, manejando las clasess previamente mencionadas, para recopilar partidos, equipos asociados, sede y claramente su nombre.

Código

```
import { Equipo } from "./Equipo.js";

export class Torneo{
    constructor(nombreTorneo,sede,cantEquiposEsperados){

        this.nombreTorneo=nombreTorneo;
        this.sede=sede;
        this.cantEquiposEsperados=Number(cantEquiposEsperados);
        this.equiposDelTorneo=new Map(); //aca iraon objetos
           equipos
        this.partidosDelTorneo=[]; //objeto partido

        /**
         Atencion, nosotros usaremos map porque:
         El Map es la estructura ideal para este caso donde
            necesitas:

        Buscar equipos frecuentemente por nombre

        Mantener una colecci n sin duplicados

        qAcceso r pido a los elementos

         */


    }

    asegurarEquipo(nombreDelEquipoABuscar){

        if(!this.equiposDelTorneo.has(nombreDelEquipoABuscar)){

```

```
            //Si no existe un equipo, lo creamos y si no,
               nosotros no permitimos que exista duplicadoss
            this.equiposDelTorneo.set(nombreDelEquipoABuscar, new
                Equipo(nombreDelEquipoABuscar));

        }

        //aseguro que si se guardo
        return this.equiposDelTorneo.get(nombreDelEquipoABuscar);

    }

    //Esta parte va a registrar un partido con las estadisticass

    aplicarPartido(partido){ //supongo que es un objeto partido

        //caso en donde el partido aparezca pendiente:
        if(partido.pendiente){
            this.partidosDelTorneo.push(partido);
            return
        }



        const equi1=this.asegurarEquipo(partido.equipoLocal);
        const equi2=this.asegurarEquipo(partido.equipoVisitante);


        //si hubo aplicamos un partido es porque paso, entonces
           se suma para ambos equipos

        equi1.cantPartidosJugados++; equi2.cantPartidosJugados++;

        //estadisticas iniciales equipo local
        equi1.golesAFavor+=partido.golesEquipoLocal;
        equi1.golesEnContra+=partido.golesEquipoVisitante;

        //estadisticas iniciales equipo visitante

        equi2.golesAFavor+=partido.golesEquipoVisitante;
        equi2.golesEnContra+=partido.golesEquipoLocal;

        //establece las victorias:
        if(partido.golesEquipoLocal>partido.golesEquipoVisitante)
            equi1.victorias++;
        else if(partido.golesEquipoLocal<partido.
           golesEquipoVisitante) equi2.victorias++;
        else{// en caso de haber empates, aunque creo que no
           pasaa}
        }

```

```
76          let rem1 = Number(partido.golesEquipoLocal)  || 0; //
               goles a n no justificados en E1
77          let rem2 = Number(partido.golesEquipoVisitante) || 0;
78
79          for (const infoGol of partido.infoGoles) {
80          const enE1 = equi1.jugadoresDelEquipo.some(j => j.nombre
               === infoGol.jugadorNombre);
81          const enE2 = equi2.jugadoresDelEquipo.some(j => j.nombre
               === infoGol.jugadorNombre);
82
83          if (enE1 && !enE2) {
84              const j = equi1.jugadoresDelEquipo.find(j => j.nombre
                    === infoGol.jugadorNombre);
85              j.cantidadGoles++; j.minutosDeGol.push(infoGol.
                   minutoGol); rem1--;
86          } else if (!enE1 && enE2) {
87              const j = equi2.jugadoresDelEquipo.find(j => j.nombre
                    === infoGol.jugadorNombre);
88              j.cantidadGoles++; j.minutosDeGol.push(infoGol.
                   minutoGol); rem2--;
89          } else if (!enE1 && !enE2) {
90              // Heur stica: asigna al equipo que a n tiene goles
                    por justificar
91              if (rem1 > 0 && rem2 === 0) {
92              rem1--;
93              // opcional: crear     placeholder     para que
                   aparezca en tops
94              let j = equi1.jugadoresDelEquipo.find(j => j.nombre
                   === infoGol.jugadorNombre);
95              if (!j) {
96                  j = { nombre: infoGol.jugadorNombre, posicion: "
                       DESCONOCIDA", numero: 0, edad: 0, cantidadGoles
                       : 0, minutosDeGol: [] };
97                  equi1.jugadoresDelEquipo.push(j);
98              }
99              j.cantidadGoles++; j.minutosDeGol.push(infoGol.
                   minutoGol);
100             } else if (rem2 > 0 && rem1 === 0) {
101             rem2--;
102             let j = equi2.jugadoresDelEquipo.find(j => j.nombre
                   === infoGol.jugadorNombre);
103             if (!j) {
104                 j = { nombre: infoGol.jugadorNombre, posicion: "
                       DESCONOCIDA", numero: 0, edad: 0, cantidadGoles
                       : 0, minutosDeGol: [] };
105                 equi2.jugadoresDelEquipo.push(j);
106             }
107             j.cantidadGoles++; j.minutosDeGol.push(infoGol.
                   minutoGol);
108             } else {
```

```
109             // Ambig edad: a n no sabemos a qui n asignarlo
                   sin arriesgar incoherencia
110             console.warn(`Gol no asignado por falta de plantel y
                   ambig edad: ${infoGol.jugadorNombre} (min ${
                   infoGol.minutoGol}) en ${partido.equipoLocal} vs ${
                   partido.equipoVisitante}`);
111             }
112         } else {
113             // enE1 && enE2: mismo nombre en ambos equipos
                   se al de datos sucios
114             throw new Error(`Ambig edad: ${infoGol.jugadorNombre
                   } aparece en ambos equipos`);
115         }
116         }
117         //sin importar lo que pase, alguno de los equipos se
               queda  eliminado en esta fase
118         equi1.faseAlcanzada=capitalizar(partido.fase);
119         equi2.faseAlcanzada=capitalizar(partido.fase);
120
121         this.partidosDelTorneo.push(partido);
122     }
123 }
124
125 //la entradaa deben se ser cadenas que signifiquen una fase de
       eliminacion que llego algun equipo
126 export function capitalizar(cadenaDeFase){
127
128     //si tiene contenido dentro la cadena fase, tenemos 2
           opciones
129     //1) separar la primer letra, volverla mayucula y concatenar
           con lo restante de la cadena y enviar la cadenaFase vacia
130     return cadenaDeFase? cadenaDeFase.charAt(0).toUpperCase()+
           cadenaDeFase.slice(1): cadenaDeFase;
131
132 }
133
134 function normalizaFase(fase) {
135   return (fase || "").toLowerCase();
136 }
137
138 function faseSiguiente(fase) {
139   switch (normalizaFase(fase)) {
140     case "cuartos": return "Semifinal";
141     case "semifinal": return "Final";
142     case "final": return "Campe n"; // opcional
143     default: return "-";
144   }
145 }
146
147 function actualizarFaseAlcanzada(equipo, fase, gano) {
148   if (!equipo) return;
```

```
149   if (gano) {
150     equipo.faseAlcanzada = faseSiguiente(fase);
151   } else {
152     // solo marca la fase actual si no se le hab a asignado algo
          mayor
153     if (!equipo.faseAlcanzada) {
154       equipo.faseAlcanzada = capitalizar(fase);
155     }
156   }
157 }
158 //para levantar el sistema en un servidor local
159 // "npx http-server ./src -p 5173 -c-1"
160 // "http://localhost:5173/UI/index.html"
```

Capítulo 7

Archivo token.js

Definimos que tokens vamos a devolver segun la existencia de algun lexema. Además los ubicamos según su posición en una fila y columna.

Código

```
//Conjunto base que para pertenecer se deben cumplir una serie de
    filtros
export class Token{
    constructor(lexema,tipo,fila,columna){

        this.lexema=lexema;
        this.tipo=tipo;
        this.fila=fila;
        this.columna=columna;
    }
}

//Palabras reservadas:
/*
Atenccion,falta reconer un marcadot tipo 3-1 como un token:
   marcador
En este ejemplo que me dio mi auxiliar lo considero una cadena

*/
export const PALABRAS_RESERVADAS=[

    "TORNEO",
    "EQUIPOS",
    "ELIMINACION",
    "equipo",
    "jugador",
    "resultado",
    "partido",
    "goleador",
    "cuartos",
    "semifinal",
    "final",
```

```
    "vs",
    "goleadores",
    "minuto"
];

export const ATRIBUTOS_VALIDOS=[
    "nombre",
    "sede",
    "equipos",
    "posicion",
    "numero",
    "edad",

];

export const TIPOS_DE_SIMBOLOS={
    "{":"Llave izquierda",
    "}":"Llave derecha",
    "[": "Corchete Izquierdo",
    "]":"Corchete Derecho",
    ":": "Dos puntos",
    ",": "Coma"


}
```

Capítulo 8

lexer.js

Esto es nuestro automata finito determinista, hacemos cambios de estados segun los diferentes lexemas encontrados.

Código

```
import { Token, PALABRAS_RESERVADAS, ATRIBUTOS_VALIDOS,
   TIPOS_DE_SIMBOLOS } from "./token.js";

export class Lexer{
    constructor(entradaDeTexto){

        this.entradaDeTexto=entradaDeTexto; //entradaDeeTexto=ET
        this.posicionActualEnLaET=0;
        this.filaEnLaET=1;
        this.columnaEnLaET=1;

        this.listaDeTokensEncontrados=[];
        this.listaDeErroresEncontrados=[];

        /*

        Todos los automatas infinitos van a iniciar en el estado
           INCIAL

        */
        this.estadoActualDelAutomata="INICIAL";
        this.buffer="";
        this.columnaDeInicio=1;
    }

    //An lisis lexico
    analizarEntradaDeTexto(){
        //Me detengo hasta que la posicion actual sea igual  al
           tama o de la entrada de texto
        while(this.posicionActualEnLaET<this.entradaDeTexto.
           length){

```

```
29              //Recorro caracter por caracter
30              const charActual=this.entradaDeTexto[this.
                   posicionActualEnLaET];
31
32              if(this.estadoActualDelAutomata==="INICIAL"){
33
34                  if(charActual===" "||charActual==="\t"||
                       charActual==="\r"){ //Ignoramos espacios,tabs y
                        retornos al inicio de la linea
35
36                      //A pesar que las ignoramos, debemos avanzar
                           de posicion
37                      this.avanzarDePosicion();
38                      continue; //Se sigue iterando no puedo parar
39                  }
40                  //salto de linea
41                  if(charActual==="\n"){
42                      this.filaEnLaET++; //Un salto de linea
                           aumenta la fila
43                      this.columnaEnLaET=1; //Y la columna se
                           reinicia
44                      this.posicionActualEnLaET++; //Avanzo de
                           posicion en la entrada de texto
45                      continue; //Se sigue iterando no puedo parar
46                  }
47                  // Letra -> IDENT (inicializa buffer con el
                       primer char y AVANZA)
48                  if(this.esLetraELCaracter(charActual)){
49                      this.estadoActualDelAutomata="IDENT" //cambia
                            el estado a IDENT
50                      this.buffer=charActual; // teorimacembnte el
                           buffer me permite almacenar los caracteres
51                      this.columnaDeInicio=this.columnaEnLaET //
                           guardo la columna de inicio
52                      this.avanzarDePosicion();
53                      continue;
54                  }
55
56                  //Digito->num
57                  if(this.esDigitoElCaracter(charActual)){
58                      this.estadoActualDelAutomata="NUM"; //
                           cambiamos el estado a numero
59                      this.buffer=charActual;
60                      this.columnaDeInicio=this.columnaEnLaET;
61                      this.avanzarDePosicion();
62                      continue;
63                  }
64                  // Comilla -> CADENA (no guardo la comilla en el
                       buffer)
65                  if(charActual==='"'){ // o sea si encontramos una
                       comilla significa que se viene un string
```

```
66
67                  this.estadoActualDelAutomata="CADENA";
68                  this.buffer=""; //aca se va a almcenar la
                        cadena
69                  this.columnaDeInicio=this.columnaEnLaET;
70                  this.avanzarDePosicion(); //No quiero guardar
                        la comilla inicial
71                  continue;
72
73              }
74
75              //esto es para anlizar los simbolos {},[]-;-:
76              const tipoSimbolo=this.esTipoDeSimbolo(charActual
                    );
77              if(tipoSimbolo!=null){
78
79                  this.listaDeTokensEncontrados.push(new Token(
80                      charActual, //Lexema
81                      tipoSimbolo, //tipo
82                      this.filaEnLaET,
83                      this.columnaEnLaET
84                  ));
85                  this.avanzarDePosicion();
86                  continue;
87              }
88
89              //Ya puse todos los posibles inicios del automata
                    , sillego hasta aca es porque existe un erorr:
90
91              this.agregarError(charActual,"Token invalido","
                    Este lexer no reconoce este caracter");
92              this.avanzarDePosicion();
93              continue;
94          }
95
96          //======Estado IDENT========:
97          if(this.estadoActualDelAutomata==="IDENT"){
98              if(this.posicionActualEnLaET<this.entradaDeTexto.
                    length
99                  && this.esLetraELCaracter(this.entradaDeTexto
                        [this.posicionActualEnLaET])){
100                 this.buffer+=this.entradaDeTexto[this.
                        posicionActualEnLaET];//Es decir el budder
                        recibe el caracter actual, asi
                        retroalimentandose
101                 this.avanzarDePosicion();
102
103             } else{
104
105                 //vamos a clasificar el lexema, es decir el
                        buffer est  listo para cargar un token
```

```
106
107                     //le paso un posible lexema y a donde deberia
                            pertenecer
108                     if(this.estaEnLista(this.buffer,
                            PALABRAS_RESERVADAS)){
109
110                         this.listaDeTokensEncontrados.push(new
                                Token(
111                             this.buffer,
112                             "Palabra Reservada",
113                             this.filaEnLaET,
114                             this.columnaDeInicio //a mi me
                                    interesa en donde comienza esta
                                    palabra reservada
115                         ));
116                     }
117                     else if(this.estaEnLista(this.buffer,
                            ATRIBUTOS_VALIDOS)){
118
119                         this.listaDeTokensEncontrados.push(
120                             new Token(
121                                 this.buffer,
122                                 "Atributo valido",
123                                 this.filaEnLaET,
124                                 this.columnaDeInicio
125                             )
126                         );
127                     }
128                     //si no es ni una, entonces cae en un error
129                     else{
130                         this.agregarError(this.buffer,"Token no
                                valido","Identificador No permitido",
                                this.columnaDeInicio);
131                     }
132                     //si se capto un identifaco, vuelvo al estado
                            iniclal para seguir reconociendo
133                     this.estadoActualDelAutomata="INICIAL" //
                            vuelvo para reconocer otros estados
134
135                 }
136                 continue;
137             }
138
139             //Estado para NUM
140             if(this.estadoActualDelAutomata==="NUM"){
141                 if(this.posicionActualEnLaET< this.entradaDeTexto
                        .length
142                     && this.esDigitoElCaracter(this.
                            entradaDeTexto[this.posicionActualEnLaET]))
                            {
```

```
143                     this.buffer+=this.entradaDeTexto[this.
                            posicionActualEnLaET];
144                     this.avanzarDePosicion();
145 
146                 }else{
147                     //Al compppletar el buffer con un patron
                            asociado a un token numero
148                     this.listaDeTokensEncontrados.push(
149                         new Token(
150                             this.buffer,
151                             "N mero",
152                             this.filaEnLaET,
153                             this.columnaDeInicio));
154                     this.estadoActualDelAutomata="INICIAL";
155 
156                 }
157                 continue;
158             }
159 
160             if(this.estadoActualDelAutomata==="CADENA"){
161 
162                 //No se encontro una comilla de cierre:
163                 if(this.posicionActualEnLaET>=this.entradaDeTexto
                        .length){
164 
165                     this.agregarError(this.buffer,
166                         "Token invalido",
167                         "No existe comilla de cierre",this.
                                columnaDeInicio);
168 
169                     this.estadoActualDelAutomata="INICIAL";
170                     continue;
171                 }
172 
173                 //si lo anterior no pasa es porque seguimos
                        almacenando en el buffer
174 
175                 if(charActual==='"'){
176 
177                     const tipoDeCadena=this.esMarcadorCadena(this
                            .buffer)? "Marcador":"Cadena"; //la funcion
                             anterior me da un true o false, si es t->
                            Marcador, si es f->Cadena
178                     this.listaDeTokensEncontrados.push(new Token(
179 
180                         this.buffer, //lexema
181                         tipoDeCadena, //tipo
182                         this.filaEnLaET,
183                         this.columnaDeInicio
184                         )
185                     )
```

```
                    this.avanzarDePosicion();
                    this.estadoActualDelAutomata="INICIAL" //si
                       se cierra una cadena, el token esta
                       terminado, entonces volvemos al principio
                    continue;
                }

                //soportar saltos de linea en las cadenas

                if(charActual==="\n"){
                    this.filaEnLaET++;
                    this.columnaEnLaET=1;
                    this.buffer+="\n"
                    this.posicionActualEnLaET++;
                    continue

                }

                //en caso que no haya cierre, concatenamos lo
                   nuevo captado por el buffer

                this.buffer+=charActual;
                this.avanzarDePosicion();
                continue;
            }
        }

        return {tokens:this.listaDeTokensEncontrados,
            errores:this.listaDeErroresEncontrados};
    }



    //Enlita errores en la tabla de errores
    agregarError(lexema,tipo,descripcion, col=this.columnaEnLaET)
       {

        this.listaDeErroresEncontrados.push({
            error:lexema,
            tipo,
            descripcion,
            filaDeError:this.filaEnLaET,
            colError:col });
        //La nomenclatura anterior me permite no siempre mandar
           esos argumentos a la funcion
    }


    //============= Helpers para la validacion de info=====

    esLetraELCaracter(char){
```

```
232
233         return (char>="A" && char<="Z") || (char>="a" && char<="z
                ") || char==="_";
234
235     }
236
237     esDigitoElCaracter(char){
238
239         return char>="0" && char<="9";
240
241     }
242
243     esTipoDeSimbolo(char){
244         //Este diccionario me daba sus nombres respeccto al signp
245         if(char==="{") return TIPOS_DE_SIMBOLOS["{"];
246         if(char==="}") return TIPOS_DE_SIMBOLOS["}"];
247         if(char==="[") return TIPOS_DE_SIMBOLOS["["];
248         if(char==="]") return TIPOS_DE_SIMBOLOS["]"];
249         if(char===":") return TIPOS_DE_SIMBOLOS[":"];
250         if(char===",") return TIPOS_DE_SIMBOLOS[","];
251         return null;
252
253     }
254
255     //Para ver las listas de tokens y errores
256     estaEnLista(lexemaABuscar,lista){
257
258         for(let i=0; i<lista.length; i++){
259             if(lista[i]===lexemaABuscar) return true;
260
261         }
262         return false;
263     }
264
265     //Especificamente nos dijeron que los resultados los
            reconocieramos como un token y no como cadenas
266
267     esMarcadorCadena(marcador){
268         let pos=0;
269
270         let tieneIzq=false,tieneDere=false; //izquierda-derecha=
                numeros del marcador
271
272         while(pos<marcador.length && this.esDigitoElCaracter(
                marcador[pos])){tieneIzq=true;pos++;};
273
274         if(!tieneIzq) return false; //Si no hay izquierda, no se
                cumple el formato
275
276
```

```
277         if(pos>=marcador.length || marcador[pos]!="-") return
               false;
278
279         pos+=1;
280
281         while(pos<marcador.length && this.esDigitoElCaracter(
               marcador[pos])){tieneDere=true;pos++}
282         return tieneDere && pos===marcador.length; //Significa
               que se cumple el formato y se recorrio toda la cadena
283     }
284     //Avamzar es simplemente cambiar posicion y columna por
           concecuente
285     avanzarDePosicion(){
286         this.posicionActualEnLaET++;
287         this.columnaEnLaET++;
288     }
289
290 }
```

Capítulo 9

flujoDeTokens.js

Tomando una lista de tokens, hacemos funciones capaces de ver que tokens siguen despues, esto nos sirve para poder hacer un buen consumo de tokens. De igual forma realizamos la lógica de consumo.

Código

```
/*

Recordemos que nosotrs al guardar un token en un objeto o en algo
hace que lo considermos un dato,que nos permite manipular info
A esto lo llamamos consumo de tokens

Este es un flujo seguro para iterar tokens y terminar sus
   validaciones para los posteriores reportes

*/


//demas imports

export class TokenStream{
    constructor(listaDeTokens){ //desde el lexer nosotros
       almacenamos los tokens en una lista
        this.listaDeTokens=listaDeTokens;
        this.idx=0; //esto nos permite movernos en la listas
    }

    //peek=ojeada, es ver si consumir:

    verQueHayEnLaLista(posPorDefecto=0){ return this.
       listaDeTokens[this.idx+posPorDefecto]|| null;} //estoy
       viendp mas adelante, que tanto? Pues lo que me pasen en la
       funcion, o por defecto solo con idx
    //OJO si me paso del tama o de la lista->null


    //ahora esto si sirve para consumir y avanzar:
```

```
27
28      siguienteEnLaLista(){return this.listaDeTokens[this.idx++]||
           null;} //devuvo el token actual y despues incremeto el idx
           uno m s
29
30      //ver si ya terminamos:
31      seraQueTerminamosLaLista(){return this.idx>=this.
           listaDeTokens.length;}
32
33      lexemaEsperado(lex){
34
35          const token=this.siguienteEnLaLista();//consumo pero
               devuelve el token actual en donde me encontraba
36          if(!token || token.lexema!==lex){ //atributo lexema de un
               token
37              this.throwError(`Se esperaba '${lex}'`, token);
38          }
39          return token;
40
41      }
42
43      // es un atributo
44      tipoEsperado(tipoEsperado){
45
46          const token=this.siguienteEnLaLista();
47          if(!token || token.tipo!==tipoEsperado){
48              this.throwError(`Se esperaba '${tipoEsperado}'`,
                   token);
49
50          }
51          return token;
52
53      }
54
55      //Yo considero a los atributos lexemas al final de cuentas
56      atributoEsperado(nombreAtributo){
57
58          const token=this.siguienteEnLaLista();
59
60          if(!token|| token.lexema!==nombreAtributo){
61
62              this.throwError(`Se esperaba atributo '${
                   nombreAtributo}'`, token);
63
64          }
65          return token;
66
67      }
68
69      //Para palabras reservadas:
70
```

```
71      palabraReservadaEsperada(PReser){
72
73          return this.lexemaEsperado(PReser);
74      }
75
76      throwError(msg,token){
77          if (typeof msg !== "string") msg = String(msg);
78          //Si existe el token no esta vacio y no e cumple los
               estandares esperados
79          //Caemos en este metodo, donde se accede al token,su fila
               , su lexema  y lo del jason me permite que sea visto en
                consola de forma amena, con comillas detro
80          //Si el token esta vacio, esta variable estara vacia
81          const dondeFueElError=token? ` (l nea ${token.fila},
               columna ${token.columna}, lexema=${JSON.stringify(token
               .lexema)})` : "";
82
83          throw new Error(`${msg}${dondeFueElError}`);
84      }
85
86  }
```

Capítulo 10

consumididorBloqueElimacion.js

A base de la clase de tokens, se van haciendo validadciones para poder predecir un buen formato del archivo de entrada, centrado en el bloque que me habla del desarrollo de los partidos.

Código

```
// ------------- ELIMINACION { fases: [ partidos ] ... } ---------------------
import {Partido} from "../Modelos/Partido.js"
import { consumirBloqueEquipos, consumirNumero } from "./consumidorBloqueEquipos.js";
import {Gol} from "../Modelos/Gol.js"

//TStream es un onjeto flujo de tokes y torneo es un objeto torneo
export function consumirBloqueEliminacion(TStream,torneo){

    //Esto es lo que cierra los bloques de elimiaciones
    while(!TStream.seraQueTerminamosLaLista() && TStream.verQueHayEnLaLista().lexema!=="}"){

        const token=TStream.verQueHayEnLaLista();

        //aca solo estoy viendo que lo que siguie en la lista tiene coherencia respecto al formato
        if(token.lexema==="cuartos"|| token.lexema==="semifinal"|| token.lexema==="final"){
            const fase=TStream.siguienteEnLaLista().lexema;//consume la fase de la que me esta hablando

            TStream.lexemaEsperado(":");
            TStream.lexemaEsperado("[");

            //ahora sigue la laista de partidos:

            consumirListaDePartidos(TStream,torneo,fase);
            TStream.lexemaEsperado("]")
```

```
25
26              //aca se consumen las comas que separan las fases:
27
28              if(TStream.verQueHayEnLaLista() && TStream.
                   verQueHayEnLaLista().lexema===","){
29                  TStream.siguienteEnLaLista();//se consume la coma
30
31              }
32          }
33          //en caso en que estemos en la coma
34
35          else if(token.lexema===","){
36              TStream.siguienteEnLaLista();
37          }
38          else{
39              TStream.throwError("Se esperaba una fase (cuartos|
                   semifinal|final) o '}'",token);
40          }
41      }
42  }
43
44
45  //TStream=flujoDeTokens y torneo son objetos, fase es una
       variable que me servira para la construccion de otro objeto
46  function consumirListaDePartidos(TStream,torneo,fase){
47
48      while(!TStream.seraQueTerminamosLaLista() && TStream.
           verQueHayEnLaLista().lexema!=="]"){
49
50          TStream.palabraReservadaEsperada("partido");//se consume
               el token y se avanza
51          TStream.lexemaEsperado(":");
52
53          //parseo de los equipos que jugaron
54
55          const equi1=TStream.tipoEsperado("Cadena").lexema; //me
               devuelde el lexema del token con tipo cadena, que debe
               ser el nombre del equipo
56          TStream.palabraReservadaEsperada("vs");//se consume el
               token con el lexema de la palabra reservadad y se
               avanzaa
57          const equi2=TStream.tipoEsperado("Cadena").lexema;
58
59          //debe seguir el parseo del resultado:
60
61          TStream.lexemaEsperado("[");
62          TStream.palabraReservadaEsperada("resultado");
63          TStream.lexemaEsperado(":");
64          //En mi lexeer yo creaba objetos tokens con la
               posibilidad de tener como tipo de lexema:marcador
```

```
65          //hay que tomar en cuenta que el marcador puede ser 3-2,
               pendiente o empate
66          //un resultado empate lo desarollaremos des es
67
68          let golesE1=null,golesE2=null,esPendiente=false;
69
70          const TRes=TStream.verQueHayEnLaLista();
71
72          //caso donde si exista un marcador
73          if(TRes && TRes.tipo==="Marcador"){
74
75              const marcador = TStream.tipoEsperado("Marcador").
                   lexema;
76              [golesE1, golesE2] = marcador.split("-").map(n =>
                   Number(n));
77          }
78
79          else if(TRes && TRes.tipo==="Cadena" && TRes.lexema==="
               Pendiente"){
80              TStream.tipoEsperado("Cadena"); // consume "Pendiente
                   "
81              esPendiente=true;
82          }
83
84          else{
85              TStream.throwError('Se esperaba Marcador o "Pendiente
                   ',TRes)
86          }
87
88
89          //ahora en el parseo toca los registros de gol:
90
91          let goles=[];
92
93          //esto es propio del bloque que es sobre los goleadores
94          //lo que separa los goleadores de los resultados, es una
               coma.
95          //Pero si es penddiente no va a existir.
96          if(!esPendiente && TStream.verQueHayEnLaLista() &&
               TStream.verQueHayEnLaLista().lexema===","){
97              TStream.siguienteEnLaLista();//consumo la coma
                   despues del resultado
98              TStream.palabraReservadaEsperada("goleadores");//se
                   consume el token y avanzo
99              TStream.lexemaEsperado(":");
100             //inicio goleadores
101             TStream.lexemaEsperado("[");
102             goles=consumidorDeBloqueGoleadores(TStream,equi1,
                   equi2); //funcion explicada abajo pero es para todo
                    el bloque que habla de los goleadores
103             //fin goleadores
```

```
104             TStream.lexemaEsperado("]");
105         }
106         //signo quue indica que la info del partido termina
107 
108         TStream.lexemaEsperado("]");
109 
110         //si existen mas partidos, son separado por comas
111 
112         if(TStream.verQueHayEnLaLista() && TStream.
               verQueHayEnLaLista().lexema===","){
113             TStream.siguienteEnLaLista();//se consume esa coma
114         }
115 
116         //Se registra un partido
117         const partidoNuevo=new Partido(fase, equi1,equi2,golesE1,
               golesE2);
118         partidoNuevo.pendiente=esPendiente;
119         //los partidos tienen un atributo que es una lista para
               guardar info de goles
120 
121         //Entonces voy a iterar sobre la lista que hice
122         for(const gol of goles) partidoNuevo.agregarInfoGol(gol);
123         //Esto es un metodo de torneo que aplica un partido a una
                lista y hace una serie de validaciones y registros
124         torneo.aplicarPartido(partidoNuevo);
125 
126     }
127 }
128 
129 ///dedicado al apartado de quines metieron gol
130 function consumidorDeBloqueGoleadores(TStream,equi1,equi2){
131     const goles=[];
132 
133     while(!TStream.seraQueTerminamosLaLista() && TStream.
           verQueHayEnLaLista().lexema!=="]"){
134         TStream.palabraReservadaEsperada("goleador");
135         TStream.lexemaEsperado(":");
136         //tomamos el lexema si el tipo del token en donde estamos
                es una cadena
137         const nombreJugador=TStream.tipoEsperado("Cadena").lexema
               ;
138 
139         //despues de haber consumido tal token,llegamos cuadno
               nos dicen el minuto
140 
141         TStream.lexemaEsperado("[");
142         //Este metodo tambien icluye el salto del token de ":"
143         const minutoDeGol=consumirNumero(TStream,"minuto"); //me
               devuele el lexema, o sea el minuto de cuando fue el gol
144         TStream.lexemaEsperado("]");
145 
```

```
146         //asignacion del gol:
147         const gol=new Gol(nombreJugador,null, Number(minutoDeGol)
               ); //este null es porque voy a asginarle su equipo
               despues, para no estar mezclando operaciones
148         gol.equipoDelJugador=null;
149         goles.push(gol);
150 
151         //cada registro de gol esta separado por una coma
152 
153         if(TStream.verQueHayEnLaLista() && TStream.
               verQueHayEnLaLista().lexema===",") TStream.
               siguienteEnLaLista();
154     }
155      // Nota: la asociaci n  jugadorequipo  exacta requiere
           conocer planteles.
156     // Si quieres resolver aqu  , pasa un callback o el objeto
           torneo, y busca por nombre del jugador.
157 
158     return goles;
159 
160 
161 }
```

Capítulo 11

consumidorBloqueEquipos.js

A base de la clase de tokens, se van haciendo validadciones para poder predecir un buen formato del archivo de entrada, centrado en el bloque que me habla de los equipos participantes .

Código

```
// -------- EQUIPOS { (equipo: "Nombre" [ jugadores... ]) , ... } ----------


import { Jugador } from "../Modelos/Jugador.js";



//Este recibe un objeto torneo y uno de flujo de tokens
export function consumirBloqueEquipos(TStream,torneo){


    //repetimos hasta encontrar la lla ve de cierre
    //acordate que el metodo ver,no consume, solo mira que sigue
    while(!TStream.seraQueTerminamosLaLista() && TStream.verQueHayEnLaLista()?.lexema!=="}"){

        const token=TStream.verQueHayEnLaLista();

        //si lo que sigue tiene xomo lexema equipo

        if(token.lexema==="equipo"){
            TStream.palabraReservadaEsperada("equipo"); //se consume el token del flujo de tokens
            TStream.lexemaEsperado(":");

            const nombreEquipo=TStream.tipoEsperado("Cadena").lexema;

            //crea un objeto equipo con el nombre y se queda en una lista del torneo
```

```
27              const equipoNuevo=torneo.asegurarEquipo(nombreEquipo)
                    ;
28
29              TStream.lexemaEsperado("[");//se consume el token y
                    se pasa al siguuiente
30
31              consumirListaJugadores(TStream,equipoNuevo); //
                    funcion creada luego
32
33              TStream.lexemaEsperado("]");
34
35              //si revisas lo archivos de entrada, entre casa
                    equipo existen comas
36
37              //Entonces eso lo consumimos
38              if(TStream.verQueHayEnLaLista() && TStream.
                    verQueHayEnLaLista().lexema===",") TStream.
                    siguienteEnLaLista();
39
40              //caso en donde literalmente no existe siguiente,
                    pero estamos en la posicion donde este la coma
41          }
42          else if(token.lexema===","){
43              TStream.siguienteEnLaLista(); //DE Todas formas es
                    consumido
44              }
45          //caso donde no se cumole elformato
46          else{
47              TStream.throwError("Se esperaba 'equipo' o '}' en
                    EQUIPOS", token);
48              }
49      }
50
51      if(torneo.equiposDelTorneo.size!==torneo.cantEquiposEsperados
            ){
52          //se al de falta de coherencia
53          console.warn(`Advertencia: se declararon ${torneo.
                equiposDelTorneo.size} equipos, se esperaban ${torneo.
                cantEquiposEsperados}.`);
54      }
55  }
56
57  //Funcion para consumir una lista de jugadores:
58
59  export function consumirListaJugadores(TStream,equipo){
60
61      while(!TStream.seraQueTerminamosLaLista() && TStream.
            verQueHayEnLaLista().lexema!=="]"){
62          //Todo esto sigue la forma del archivo de texto
63          TStream.palabraReservadaEsperada("jugador"); //si
                coincide con el lexema del token, seguimos
```

```
64          TStream .lexemaEsperado(":");
65
66          //si el tipo del token es una cadena, me devuelve le
               token y accedo a su lexema para tener su nombre
67          const nombreJugador = TStream.tipoEsperado("Cadena").
               lexema;
68
69
70          TStream.lexemaEsperado("[");
71
72          //Este metodo me devuelve un lexema, en este caso el
               texto que me indica en que posicion juega el don
73          const posicionDelJugador=consumirCadena(TStream,"posicion
               ")//Esta funcin la creamos abajo
74          TStream.lexemaEsperado(","); //estas comas seperan los
               atributos en el archivo
75
76          //Ya no hago lo del analizar los 2 puntos y eso, porque
               ya lo hace el metodo{}
77
78          //--------------------
79          const numeroDelJugador=consumirNumero(TStream,"numero");
80          TStream.lexemaEsperado(","); //estas comas seperan los
               atributos en el archivo
81
82          const edad=consumirNumero(TStream,"edad");
83          TStream.lexemaEsperado("]")//lllave de cierre
84
85
86          //con estos datos podemos crear un jugador y agregarlo a
               un equipo
87
88          equipo.agregarUnJugadorAlEquipo(new Jugador(nombreJugador
               ,posicionDelJugador,numeroDelJugador,edad));
89
90          //entre cada info de los jugadores hay comas que seperan:
91          if(TStream.verQueHayEnLaLista() && TStream.
               verQueHayEnLaLista().lexema===",") TStream.
               siguienteEnLaLista();
92      }
93  }
94
95  //funciones de apoyo para el parseo de los jugadores
96
97  export function consumirCadena(TStream,atributoAEvaluar){
98
99      TStream.atributoEsperado(atributoAEvaluar); //si no pasa nada
            solo se consume y pasamos al next
100
101     TStream.lexemaEsperado(":");//se consum y vamos al next
102
```

```
103     return TStream.tipoEsperado("Cadena").lexema; //si se cumple
            que el tipo del token que viene luego es una cadena, pues
            accedo al lexema del token y lo devuelvo
104 }
105
106 //el numero a evaluar puede ser un dorsal o una edads
107 export function consumirNumero(TStream,numeroAEvaluar){
108
109     TStream.atributoEsperado(numeroAEvaluar); //si el atributo
            esperado "el que pase como argumento" coincide con el
            lexema del token, se consume y vamos la siguiente
110
111     TStream.lexemaEsperado(":");
112
113     return TStream.tipoEsperado("N mero").lexema; //pongo asi
            N mero, porque asi guardaba el tipo de los tokens que
            tuviera como lexema algun numero
114
115 }
```

Capítulo 12

consumidorBloqueTorneo.js

A base de la clase de tokens, se van haciendo validadciones para poder predecir un buen formato del archivo de entrada, centrado en el bloque que me habla los datos iniciales de un torneo .

Código

```
// ---------- TORNEO { nombre: "str", equipos: num, sede: "str" } ----------
//TStream=onjeto flujo de tokens



export function consumirBloqueTorneo(TStream){

    let nombreTorneo=null,equiposEsperados=null, sede=null;

    //finalizamos hasta encontrar una llave de ciere:}

    //Mientrad sigan existiendo objetos en la lista de tokens y el lexema del token siguiente no sea una llave, podemos seguir
    while(!TStream.seraQueTerminamosLaLista() && TStream.verQueHayEnLaLista().lexema!=="}"){

        const token=TStream.verQueHayEnLaLista();

        //en caso que el lexema del tokem sea un nombre
        if(token.lexema==="nombre"){

            TStream.atributoEsperado("nombre"); //En este metodo se consume un token y se pasa el siguiente
            //entonces caemos ac

            TStream.lexemaEsperado(":");//Se cosume el token y paso al siguiente

```

```
25              nombreTorneo=TStream.tipoEsperado("Cadena").lexema;
                   //estos metodos me devuelven tokens, por eso puedo
                   acceder al lexema
26              //De igual forma lo anterior hace que se consuma el
                   token y pase al siguiente
27          }
28
29          //va a lelgar un punto donde el token que sera visto como
                el que sigue
30          //Esto gracias a const token=TStream.verQueHayEnLaLista()
               ;
31          //Cumpla con alguna de las siguintes condiciones
32          //Porque ojo, yo no me estoy quedando quieto, siempre
               estoy viendo que viene luego
33
34
35          else if(token.lexema==="equipos"){
36              TStream.atributoEsperado("equipos");//en este casp si
                    consumo y paso al siguiente
37              TStream.lexemaEsperado(":");
38              equiposEsperados=TStream.tipoEsperado("N mero").
                   lexema; //el token que me es devuelto,lo uso para
                   acceder a numero
39          }
40
41          else if(token.lexema==="sede"){
42              TStream.atributoEsperado("sede");
43              TStream.lexemaEsperado(":");
44              sede=TStream.tipoEsperado("Cadena").lexema;
45
46          }
47
48          else if(token.lexema===","){
49              TStream.siguienteEnLaLista(); //lo que pasa que los
                   atributos, son separasos por comas que consumimos
                   asi sin mas
50
51          }
52
53          else{
54              TStream.throwError("Atributo no valid en el torneo",
                   token);
55
56          }
57      }
58
59      //Evaluo la info que obtuve de los datos que guarde
60      if (!nombreTorneo || !equiposEsperados || !sede) {
61          throw new Error("TORNEO incompleto: faltan nombre,
               equipos o sede.");
62      }
```

```
63      return { nombreTorneo, equiposEsperados, sede };
64  }
```

Capítulo 13

conjuntoDeConsumidores.js

Junta todos los consumidores y los pone a trabajar uno detrás del otro.

Código

```
import { TokenStream } from "./flujoDeTokens.js";
import { consumirBloqueTorneo } from "./consumidorBloqueTorneo.js
   ";
import { consumirBloqueEquipos } from "./consumidorBloqueEquipos.
   js";
import { consumirBloqueEliminacion } from "./
   consumidorBloqueEliminacion.js";
import { Torneo } from "../Modelos/Torneo.js";

export function conjuntoDeConsumidores(listaTokens){
    const TStream=new TokenStream(listaTokens)

    TStream.lexemaEsperado("TORNEO");//se consume y se avanza al
       siguinte token
    //signo de apartura de info
    TStream.lexemaEsperado("{");
    //todos los consumides comparte el mismo flujo de tokens
    const infoTorneo=consumirBloqueTorneo(TStream); //estp me
       devuevle 3 datos que voy a usar luego

    const torneoNuevo=new Torneo(infoTorneo.nombreTorneo,
       infoTorneo.sede,infoTorneo.equiposEsperados);
    TStream.lexemaEsperado("}"); //signo de cierre info

    //Equipos{..}
    TStream.palabraReservadaEsperada("EQUIPOS");
    TStream.lexemaEsperado("{");
    consumirBloqueEquipos(TStream,torneoNuevo);
    TStream.lexemaEsperado("}")

    //Elimacion{..}
    TStream.palabraReservadaEsperada("ELIMINACION");
    TStream.lexemaEsperado("{");
```

```
28      consumirBloqueEliminacion(TStream,torneoNuevo);
29      TStream.lexemaEsperado("}")
30
31      //voy a retornar el objeto torneos, pero ojo. Al pasar por
            todos los consumidores
32      //Se le a adieron estadisticasd
33
34      return torneoNuevo;
35
36
37  }
```

Capítulo 14

common.js

Encargado de las peticiones con el servidor web para poder mantener dinámico el html y que el usurio experimente una situación amena, todas las demás archivos relacionados a la actualización constante de la página repite patrones demasiado similares.

Código

```
import { Lexer } from "../../EstructuraDeScanner/lexer.js";
import {conjuntoDeConsumidores} from "../../Consumidor/
   ConjuntoDeConsumidores.js"

//agarro los elementos del html

const textArea=document.getElementById("vistaArchivo");
const btnCargar = document.getElementById("btnCargar");
const btnBorrar=document.getElementById("btnBorrar");

//vamos a guardar/leer el texto en un discoduro del navegador
   para poder compartir info entre paginas

export function guardarTextoEntrada(txt){

    //guardo localmente en el navegador y le asigno tal nomombre
    localStorage.setItem("textoEntrada",txt);

}

export function leerTextoEntrada(){
    //en caso de estar vacia ,devolvemos un string sin nada
    return localStorage.getItem("textoEntrada") || "";
}

//evitamos hacer cualquier cosa si no tenemos almacenado en el
   disco el texto
export function asegurarTexto(){
    const texto = leerTextoEntrada();
    if(!texto){ alert("Primero carga un archivo .txt en la
       p gina principal.");throw new Error("Sin texto"); }
```

```
28      return texto;
29  }
30
31  //logica para exportar pagina
32
33  export function descargarPagina(nombre="reporte.html"){
34      //tomo todo el archivo html existente
35      const html="<!doctype html>\n" + document.documentElement.
            outerHTML;
36
37      //archivo binario donde este la info
38      //Lo del type es para que el navegador sepa que este archvio
            es de tipo  gina  web
39      const blob=new Blob([html],{type:"text/html"});
40
41      //elace para descarga
42      //Es como la etiqueta de los html, dedicados descargas. EL
            href hace referencia a la direccion del objeto blob y en
            dowland le hago una sugerencia de nombre
43      const a=Object.assign(document.createElement("a"),{href:URL.
            createObjectURL(blob),download:nombre});
44      document.body.appendChild(a); //agrego al dom enlace
45      a.click(); // un click sobre esta forza la descar
46      a.remove(); //inmediatamente se elimina para no ensuxiar el
            dom
47
48  }
49
50  //boton para cargar ahivo
51
52  if(btnCargar){
53      //Escuchado de eventos, para cuando se haga click. se llame a
            esta funcion
54      btnCargar.addEventListener("click",()=>{
55          //se crea un elemento input y se xonfigura de una vez, es
                de tipo archivo y accept me dice que solo prioriza
               archivos de texto plano
56          const input = Object.assign(document.createElement("input
               "), {type:"file", accept:".txt,text/plain"});
57
58          //por experiencia de usuario esta funcion sera
               asincronica, sin pausar la ejecucion de la pagina
59          input.onchange=async()=>{
60              //accedo al primer archivo seleccionado
61              const file=input.files?.[0];
62              if(!file) return; //en casp de no selecionar nada
63              const text=await file.text(); //se sigue con el
                   programa hasta que se resuelva esta promesa,  o sea
                    que se cargue el archivo
64              textArea.value=text; //el valor del panel con el
                   texto, va a cambiar, ahora ser
```

```
65              guardarTextoEntrada(text); //guardamos el texto en un
                    objeto del navegador, acordate, le pone el nombte
                   textoEntrada
66          };
67          input.click(); //es el click invivsible para que ejecute
               todo
68      });
69  }
70
71  //boton borrar:
72
73  if(btnBorrar){
74      btnBorrar.addEventListener("click",()=>{
75          localStorage.removeItem("textoEntrada"); //asi le puso en
                el disco duro del navegador
76          textArea.value=""; //por lo tanto reescribimos el valor
               de ese panel
77      });
78  }
79
80  //en case de haber texto guardado, rellenamos el panel:
81
82  if(textArea){
83      const text=leerTextoEntrada()//leemos lo que guarde en
           guardarTextoEntrada(text);
84      if(text) textArea.value=text;
85  }
86
87  //Navegacion para los reporte
88  function go(pagina){window.open(`../UI/${pagina}`, "_blank")} //
       el argumento blank es para que se abra en una pagina en blanco
89
90  //acciones de botoens
91
92  document.getElementById("btnAnalizar")?.addEventListener("click
       ",()=>go("analizar.html"));
93  document.getElementById("btnBracket")?.addEventListener("click
       ",()=>go("bracket.html"));
94  document.getElementById("btnEquipos") ?.addEventListener("click",
        ()=> go("equipos.html"));
95  document.getElementById("btnGoleadores")?.addEventListener("click
       ", ()=> go("goleadores.html"));
96  document.getElementById("btnInfo")      ?.addEventListener("click
       ", ()=> go("info.html"));
97
98
99  export function obtenerTokensYErrores(texto){
100     const lexer=new Lexer(texto);
101     return lexer.analizarEntradaDeTexto(); //parte de los tokens
           y errores. esto me devuele los tokens y los errores
102
```

```
103 }
104
105 export function obtenerTorneo(texto){
106   const { tokens, errores } = obtenerTokensYErrores(texto);
107   let torneo = null;
108   let parseError = null;
109   try {
110     torneo = conjuntoDeConsumidores(tokens); // orquestador
          TORNEO->EQUIPOS->ELIMINACION
111   } catch (e) {
112     parseError = e instanceof Error ? e.message : String(e);
113   }
114   return { errores, torneo, parseError };
115 }
```

Capítulo 15

Recomendaciones para el Usuario

15.1. Antes de empezar

- Verifique que tenga instalado **Node.js 18+**.
- Guarde los archivos `.txt` en una carpeta accesible y recuerde su ruta.
- Ejecute el sistema desde una terminal ubicada en la carpeta del proyecto.

15.2. Ejecución básica

1. Abra una terminal en la carpeta del proyecto.
2. Inicie el programa: `npx http-server ./src -p 5173 -c-1`.
3. Abra en cualquier navegador esta ruta: http://localhost:5173/UI/index.html.