# MANUAL TÉCNICO

**PROYECTO #2 LFP**

# Tabla de contenido

# Objetivos

- **Específicos:**
  - Análisis léxico y sintáctico de texto mediante un lenguaje definido
- **Generales:**
  - Simulación de Bot
  - Cargar archivos

Dirigido a cualquier usuario que desee saber sobre resultados del ambiento del futbol a partir de comandos que tengan la estructura del lenguaje definido.

# Especificación Técnica

**Requisitos de Hardware:**

- Mouse
- Teclado

**Requisitos de Software:**

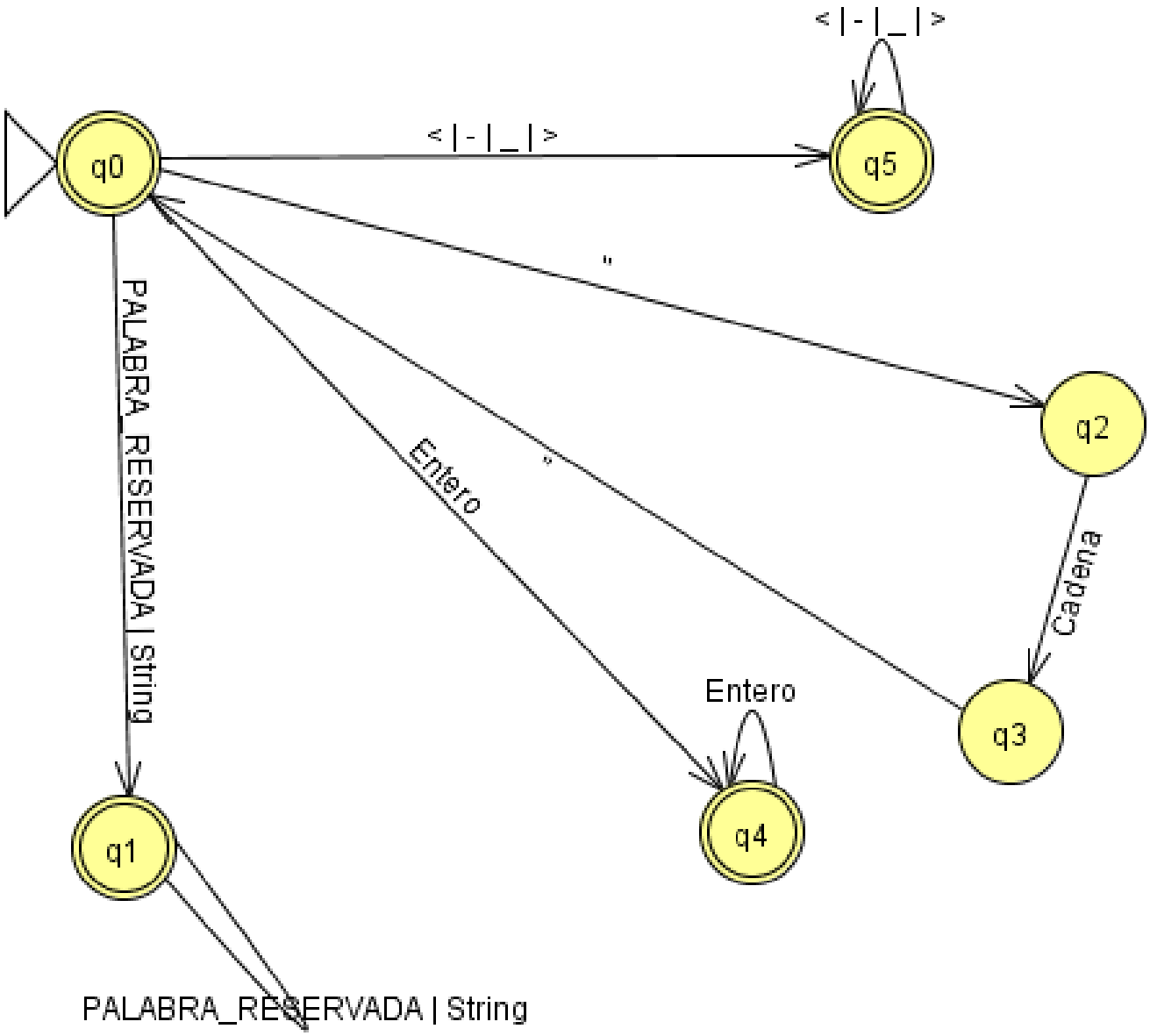
- **Sistema Operativo:** Windows 10, Linux, MAC OS
- **Herramientas:** Visual Studio Code, IDE Python
- **Lenguaje de Programación:** Python

# Tabla de Tokens

| Nombre | Descripción del patrón | Expresión regular | Ejemplos |
|---|---|---|---|
| Cadena | Una cadena de caracteres encerrada en comillas dobles | \"[^.\"]*\" | "Real Madrid" "Barcelona" |
| Menor que | Signo menor que | '<' | < |
| Entero | Numero o numeros consecutivos | \d | 2022 1999 12 2 |
| Guion | Signo guion | '-' | - |
| Mayor que | Signo mayor que | '>' | > |
| RESULTADO | reservada RESULTADO | RESULTADO | RESULTADO |
| VS | reservada VS | VS | VS |
| TEMPORADA | reservada TEMPORADA | TEMPORADA | TEMPORADA |
| JORNADA | reservada JORNADA | JORNADA | JORNADA |
| GOLES | reservada GOLES | GOLES | GOLES |
| LOCAL | reservada LOCAL | LOCAL | LOCAL |
| VISITANTE | reservada VISITANTE | VISITANTE | VISITANTE |
| TOTAL | reservada TOTAL | TOTAL | TOTAL |
| TABLA | reservada TABLA | TABLA | TABLA |
| PARTIDOS | reservada PARTIDOS | PARTIDOS | PARTIDOS |
| TOP | reservada TOP | TOP | TOP |

| | | | |
|---|---|---|---|
| SUPERIOR | reservada SUPERIOR | SUPERIOR | SUPERIOR |
| INFERIOR | reservada INFERIOR | INFERIOR | INFERIOR |
| ADIOS | reservada ADIOS | ADIOS | ADIOS |
| -f | reservada BANDERA_f | -f | -f |
| -ji | reservada BANDERA_ji | -ji | -ji |
| -jf | reservada BANDERA_jf | -jf | -jf |
| -n | reservada BANDERA_n | -n | -n |

# Autómata Finito Determinista



# Gramática

| | | |
|---|---|---|
| S | ::= | INICIO |
| INICIO | ::= | RESULTADO |
| | ::= | JORNADA |
| | ::= | GOLES |
| | ::= | TABLA |
| | ::= | PARTIDOS |
| | ::= | TOP |
| | ::= | ADIOS |
| RESULTADO | ::= | pr_resultado cadena pr_vs cadena pr_temporada menorque entero guion entero mayorque |
| JORNADA | ::= | pr_jornada entero pr_temporada menorque entero guion entero mayorque guion pr_f string |
| GOLES | ::= | pr_goles CONDICION cadena pr_temporada menorque entero guion entero mayorque |
| TABLA | ::= | pr_tabla pr_temporada menorque entero guion entero mayorque guion pr_f string |
| PARTIDOS | ::= | pr_partidos cadena pr_temporada menorque entero guion entero mayorque LISTA |
| TOP | ::= | pr_top CONDICION pr_temporada menorque entero guion entero mayorque guion pr_n entero |
| ADIOS | ::= | pr_adios |
| CONDICION | ::= | pr_local |
| | ::= | \| pr_visitante |
| | ::= | \| pr_total |

| | | |
|---|---|---|
| | ::= | \| pr_superior |
| | ::= | \| pr_inferior |

# Lógica del Programa

**Clases Utilizadas:**

- **Main**
- **Token**
- **Error**
- **ErrorS**
- **Data**
- **Interfaz**
- **Gestor**
- **Lexico**
- **Sintactico**

➢ **Main**

Clase principal, la cual se tiene que ejecutar para utilizar el programa.

```
gestor = Gestor()

if __name__ == '__main__':
    #gestor.Print()
    gestor.cargarData()
    app = Interfaz(gestor)
    #Gestor.cargarData()
```

➢ **Token**

Clase donde se almacenarán los objetos para los tokens encontrados en el análisis léxico.

```
class Token:

    def __init__(self, lexema : str, linea :int, columna :int, tipo : str) -> None:
        self.lexema = lexema
        self.linea = linea
        self.columna = columna
        self.tipo = tipo

    def printTokens(self):
        print(self.lexema, self.linea, self.columna, self.tipo)
```

- **Error**

Clase donde se almacenarán los objetos para los errores encontrados en el análisis léxico.

```python
class Error:

    def __init__(self, descripcion : str, linea : int, columna : int) -> None:
        self.descripcion = descripcion
        self.linea = linea
        self.columna = columna

    def printError(self):
        print(self.descripcion, self.linea, self.columna)
```

- **ErrorS**

Clase donde se almacenarán los objetos para los errores encontrados en el análisis sintáctico.

```python
class ErrorS:

    def __init__(self, obtenido : str, esperado : str, columna : int) -> None:
        self.esperado = esperado
        self.obtenido = obtenido
        self.columna = columna

    def printError(self):
        print(self.obtenido, self.esperado, self.columna)
```

- **Data**

Clase donde se almacenarán los objetos para la data del csv.

```python
class Data:

    def __init__(self,fecha,temporada,jornada,equipo1,equipo2,goles1,goles2):
        self.fecha = fecha
        self.temporada = temporada
        self.jornada = jornada
        self.equipo1 = equipo1
        self.equipo2 = equipo2
        self.goles1 = goles1
        self.goles2 = goles2
```

- **Interfaz**

Clase que genera la interfaz del programa.

```
class Interfaz:
    def __init__(self,gestor):
        self.gestor = gestor
        raiz = Tk()
        raiz.title("La Liga Bot")
        raiz.resizable(0,0)
        raiz.iconbitmap("Icono.ico")
        miFrame = Frame()
        miFrame.pack()
        miFrame.config(width="750",height="650")
```

- **Gestor**

Clase donde se encuentran funciones que ejecutara el análisis sintáctico y que carga el archivo csv con toda la data.

```
class Gestor:

    def __init__(self):
        self.data = []
        self.aux = []
        self.puntos = []

    def crearData(self,fecha,temporada,jornada,equipo1,equipo2,goles1,goles2):
        self.data.append(Data(fecha,temporada,jornada,equipo1,equipo2,goles1,goles2))

    def cargarData(self):
```

- **Lexico**

Clase donde se establecen todos los métodos y funciones que se utilizan para codificar el autómata que permite la lectura del comando.

```
class Lexico:

    def __init__(self) -> None:
        self.listaTokens = []
        self.listaErrores = []
        self.tokensR = []
        self.erroresR = []
        self.linea = 1
        self.columna = 0
        self.buffer = ''
        self.estado = 0
        self.simbolo = ''
        self.i = 0

    def addToken(self, caracter, linea, columna, token):
        self.listaTokens.append(Token(caracter,linea, columna, token))
        self.buffer = ''
```

- **Sintactico**

Clase donde se establecen todos los métodos y funciones que se utilizan para codificar la gramática establecida para la lectura de los comandos.

```python
class Sintactico:

    def __init__(self, tokens : list, gestor) -> None:
        self.errores = []
        self.columna = 0
        self.tokens = tokens
        self.gestor = gestor

    def agregarError(self, obtenido, esperado, columna):
        self.errores.append(ErrorS(obtenido, esperado,columna))
```