

UNIVERSIDAD SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISION DE CIENCIAS DE LA INGENIERIA

LAB. LENGUAJES FORMALES Y DE PROGRAMACION

**LUIS FERNANDO RODRIGUEZ LIMA**

**201730879**



## “MANUAL TECNICO”

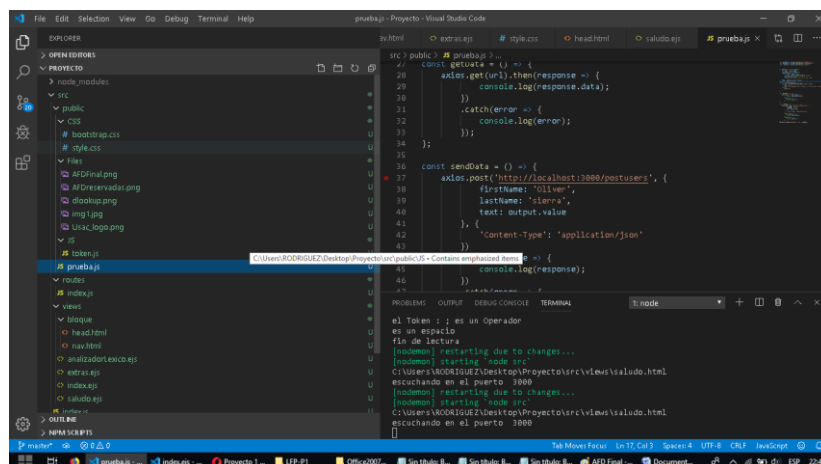
Este manual tecnico pretende explicar las funciones mas relevantes del proyecto, indicando asi la importancia del funcionamiento del mismo.

El proyecto pretende demostrar el funcionamiento de un analizador lexico, tendiendo en si un alfabeto aceptado por el analizador.

El proyecto se llevo a cabo usando tecnologia Node JS conjuntamente con JavaScript, Html y CSS.

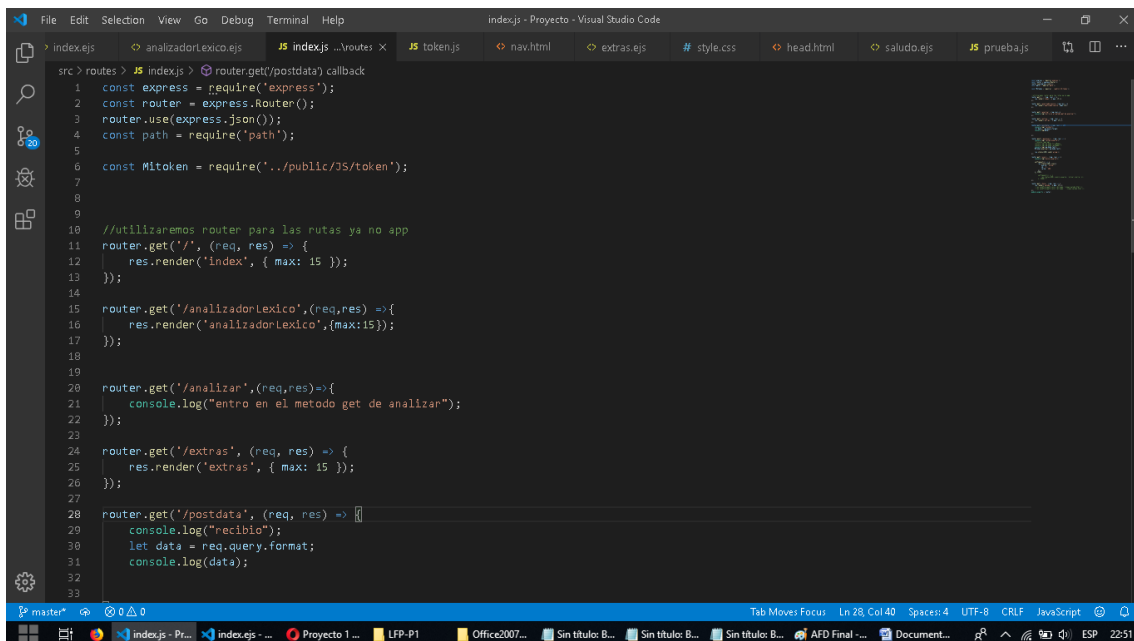
- **Organizacion:**

Se utilizo el orden siguiente:

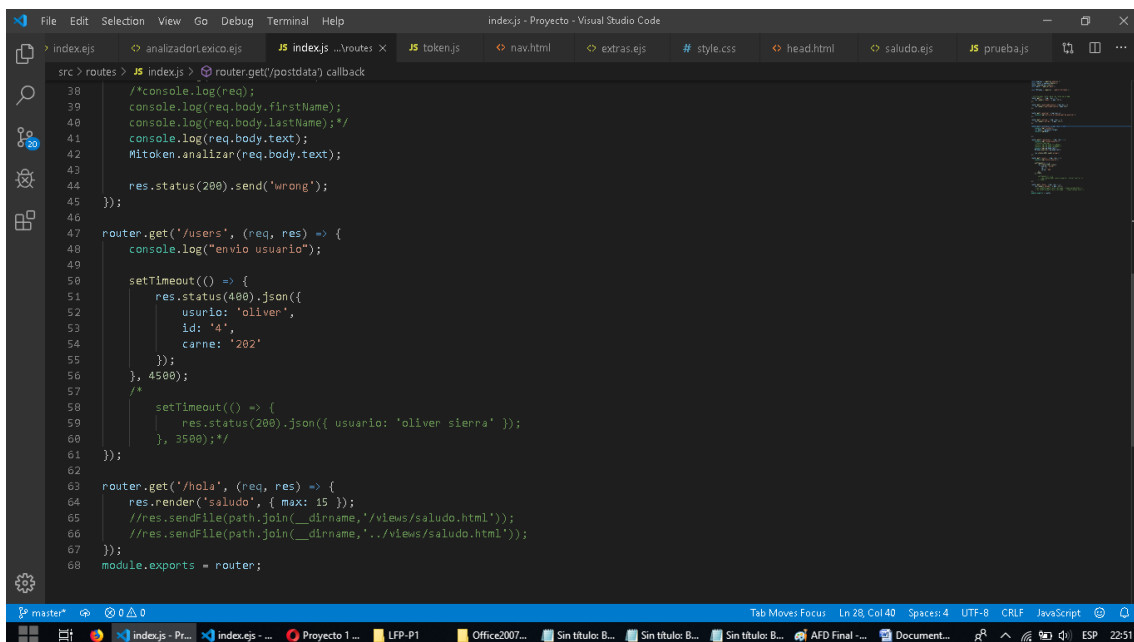


- Rutas:

## Index.ejs



```
src > routes > JS indexjs > router.get('/postdata') callback
1  const express = require('express');
2  const router = express.Router();
3  router.use(express.json());
4  const path = require('path');
5
6  const Mitoken = require('../public/35/token');
7
8
9
10 //utilizaremos router para las rutas ya no app
11 router.get('/', (req, res) => {
12   res.render('index', { max: 15 });
13 });
14
15 router.get('/analizadorLexico', (req, res) => {
16   res.render('analizadorLexico', { max: 15 });
17 });
18
19
20 router.get('/analizar', (req, res) => {
21   console.log("entro en el metodo get de analizar");
22 });
23
24 router.get('/extras', (req, res) => {
25   res.render('extras', { max: 15 });
26 });
27
28 router.get('/postdata', (req, res) => {
29   console.log("recibio");
30   let data = req.query.format;
31   console.log(data);
32 }
33 }
```

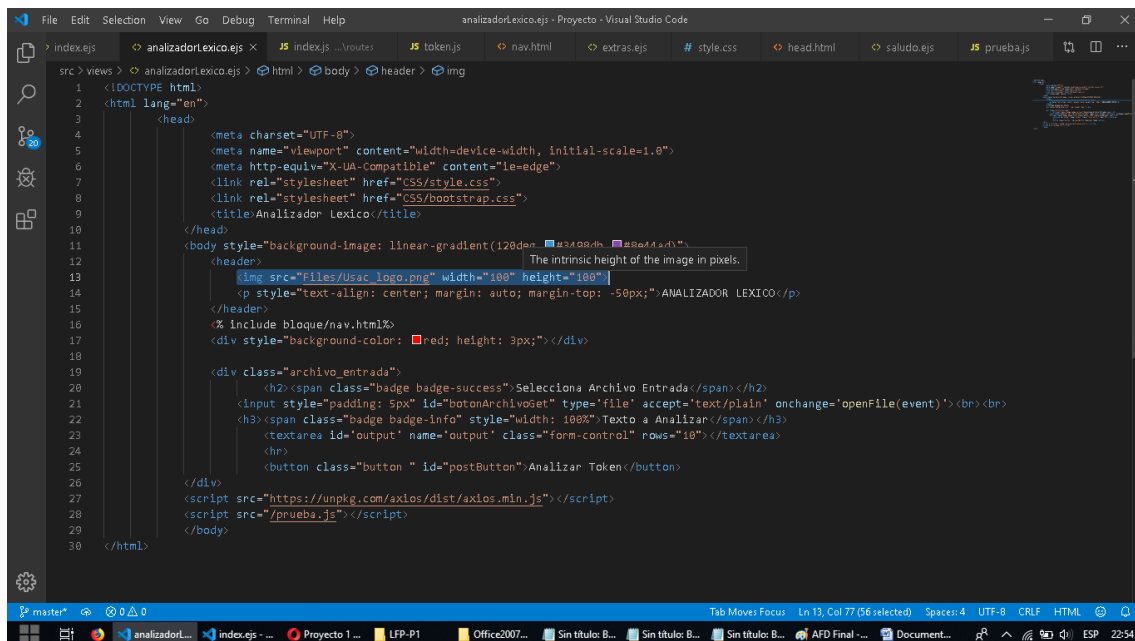


```
src > routes > JS indexjs > router.get('/postdata') callback
38  /*console.log(req);
39  console.log(req.body.firstName);
40  console.log(req.body.lastName);*/
41  console.log(req.body.text);
42  Mitoken.analizar(req.body.text);
43
44  res.status(200).send('wrong');
45 });
46
47 router.get('/users', (req, res) => {
48   console.log("envio usuario");
49
50   setTimeout(() => {
51     res.status(400).json({
52       usuario: 'oliver',
53       id: '4',
54       carne: '202'
55     });
56   }, 4500);
57   /*
58   setTimeout(() => {
59     res.status(200).json({ usuario: 'oliver sierra' });
60   }, 3500);*/
61 });
62
63 router.get('/hola', (req, res) => {
64   res.render('saludo', { max: 15 });
65   //res.sendFile(path.join(__dirname, './views/saludo.html'));
66   //res.sendFile(path.join(__dirname, './views/saludo.html'));
67 });
68 module.exports = router;
```

En esta clase se capturan todos los metodos GET y POST enviados desde el usuario por medio del navegador, esta clase caputra la informacion enviada y segun sea la necesidad envia la informacion requerida.

- Views

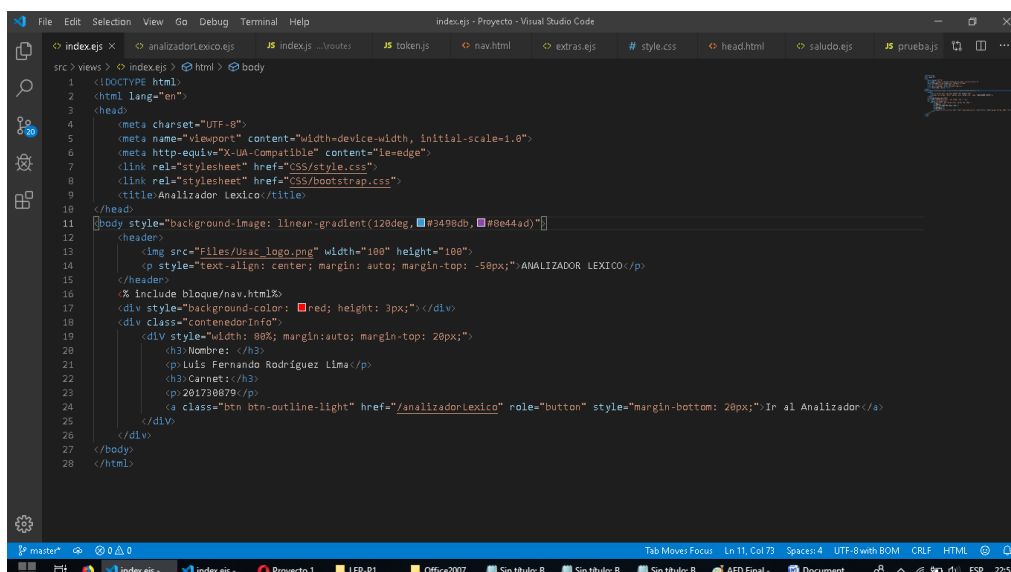
## analizadorLexico.ejs



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="CSS/style.css">
8   <link rel="stylesheet" href="CSS/bootstrap.css">
9   <title>Analizador Lexico</title>
10 </head>
11 <body style="background-image: linear-gradient(120deg, #3498db, #9b59b6, #34495e);">
12   <header>
13     
14     <p style="text-align: center; margin: auto; margin-top: -50px;">ANALIZADOR LEXICO</p>
15   </header>
16   <% include bloque/nav.html%>
17   <div style="background-color: #f2f2f2; height: 3px;"></div>
18   <div class="archivo_entrada">
19     <h2><span class="badge badge-success">Selecciona Archivo Entrada</span></h2>
20     <input style="padding: 5px;" id="botonArchivoGet" type="file" accept="text/plain" onchange="openFile(event)"><br><br>
21     <h3><span class="badge badge-info" style="width: 100%;">Texto a Analizar</span></h3>
22     <div id="output" name="output" class="form-control" rows="10"></div>
23     <hr>
24     <button class="button" id="postButton">Analizar Token</button>
25   </div>
26   <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
27   <script src="/prueba.js"></script>
28 </body>
29 </html>
```

Esta clase por asi decirlo contiene codigo html en donde muestra visualmente al usuario las acciones que puede realizar como desplazarse dentro de la aplicacion con los botones y por supuesto realizar el analizador.

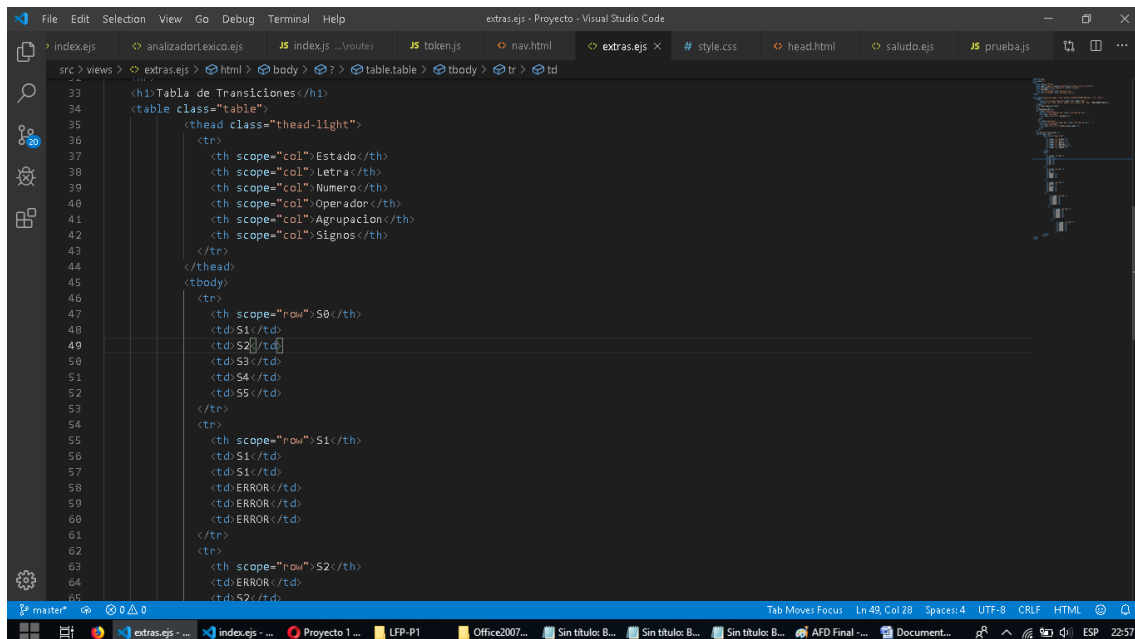
## Index.ejs



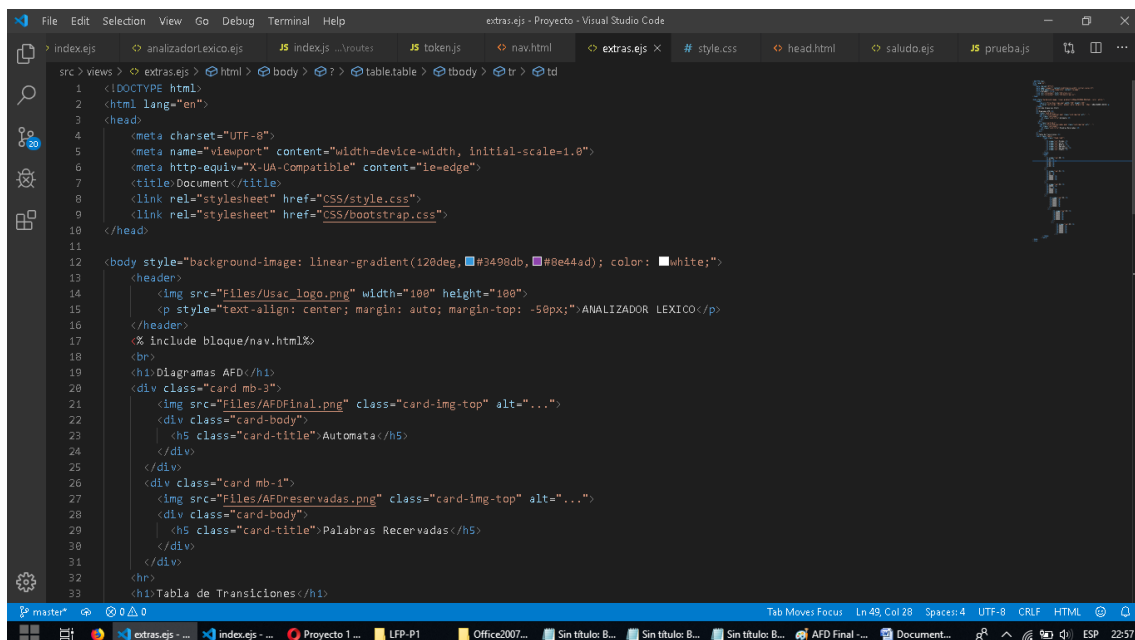
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <link rel="stylesheet" href="CSS/style.css">
8   <link rel="stylesheet" href="CSS/bootstrap.css">
9   <title>Analizador Lexico</title>
10 </head>
11 <body style="background-image: linear-gradient(120deg, #3498db, #9b59b6, #34495e);">
12   <header>
13     
14     <p style="text-align: center; margin: auto; margin-top: -50px;">ANALIZADOR LEXICO</p>
15   </header>
16   <% include bloque/nav.html%>
17   <div style="background-color: #f2f2f2; height: 3px;"></div>
18   <div class="contenidoInfo">
19     <div style="width: 80%; margin:auto; margin-top: 20px;">
20       <h3>Nombre: </h3>
21       <p>Luis Fernando Rodriguez Lima</p>
22       <h3>Carnet:</h3>
23       <p>201738879</p>
24       <a class="btn btn-outline-light" href="/analizadorLexico" role="button" style="margin-bottom: 20px;">Ir al Analizador</a>
25     </div>
26   </div>
27 </body>
28 </html>
```

Esta clase contiene código HTML y es la página principal de la aplicación, muestra la información del estudiante y la barra de navegación para ir al analizador Lexico.

## Extras.ejs



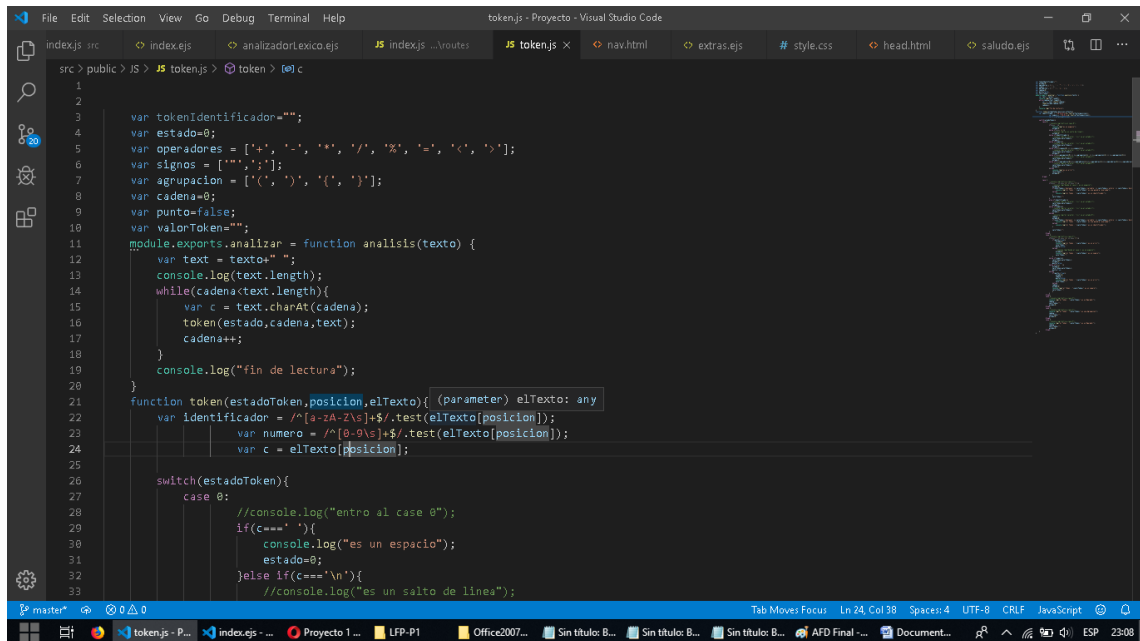
```
src> views > extras.ejs > html > body > ? > table.table > tbody > tr > td
33 <h1>Tabla de Transiciones</h1>
34 <table class="table">
35   <thead class="thead-light">
36     <tr>
37       <th scope="col">Estado</th>
38       <th scope="col">Letra</th>
39       <th scope="col">Numero</th>
40       <th scope="col">Operador</th>
41       <th scope="col">Agrupacion</th>
42       <th scope="col">Signos</th>
43     </tr>
44   </thead>
45   <tbody>
46     <tr>
47       <th scope="row">S0</th>
48       <td>S1</td>
49       <td>S2</td>
50       <td>S3</td>
51       <td>S4</td>
52       <td>S5</td>
53     </tr>
54     <tr>
55       <th scope="row">S1</th>
56       <td>S1</td>
57       <td>S1</td>
58       <td>ERROR</td>
59       <td>ERROR</td>
60       <td>ERROR</td>
61     </tr>
62     <tr>
63       <th scope="row">S2</th>
64       <td>ERROR</td>
65       <td>S2</td>
```



```
src> views > extras.ejs > html > body > ? > table.table > tbody > tr > td
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Document</title>
8   <link rel="stylesheet" href="CSS/style.css">
9   <link rel="stylesheet" href="CSS/bootstrap.css">
10 </head>
11
12 <body style="background-image: linear-gradient(120deg, #3498db, #9e44ad); color: #white;">
13   <header>
14     
15     <p style="text-align: center; margin: auto; margin-top: -50px;">ANALIZADOR LEXICO</p>
16   </header>
17   <% include bloque/nav.html%>
18   <br>
19   <h1>Diagramas AFD</h1>
20   <div class="card mb-3">
21     
22     <div class="card-body">
23       <h5 class="card-title">Automata</h5>
24     </div>
25   </div>
26   <div class="card mb-1">
27     
28     <div class="card-body">
29       <h5 class="card-title">Palabras Reservadas</h5>
30     </div>
31   </div>
32   <hr>
33   <h1>Tabla de Transiciones</h1>
```

Esta clase contiene código HTML y muestra los diagramas y la tabla de transición propia del proyecto.

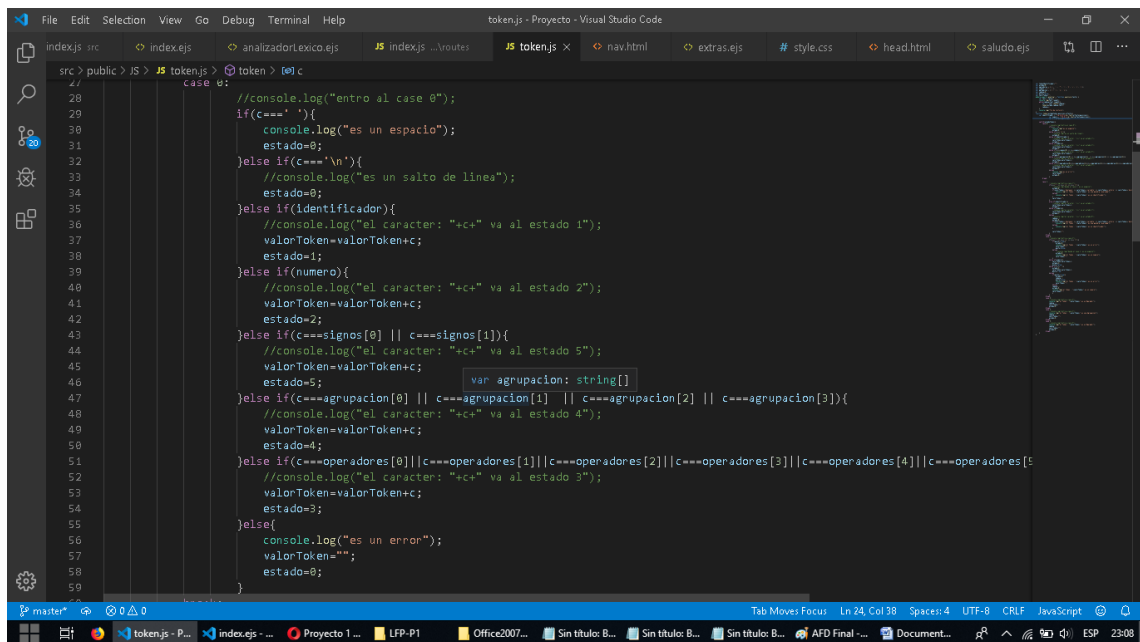
- Token.ejs



The screenshot shows the Visual Studio Code editor with a file named `token.js` open. The code is a JavaScript file for a tokenizer. It includes variables for `tokenIdentificador`, `estado`, `operadores`, `signos`, `agrupacion`, `cadena`, `punto`, and `valorToken`. A function `analizar` is defined, which takes `texto` as input and processes it character by character. The code is currently at line 33, showing the start of a `switch` statement for `estadoToken`.

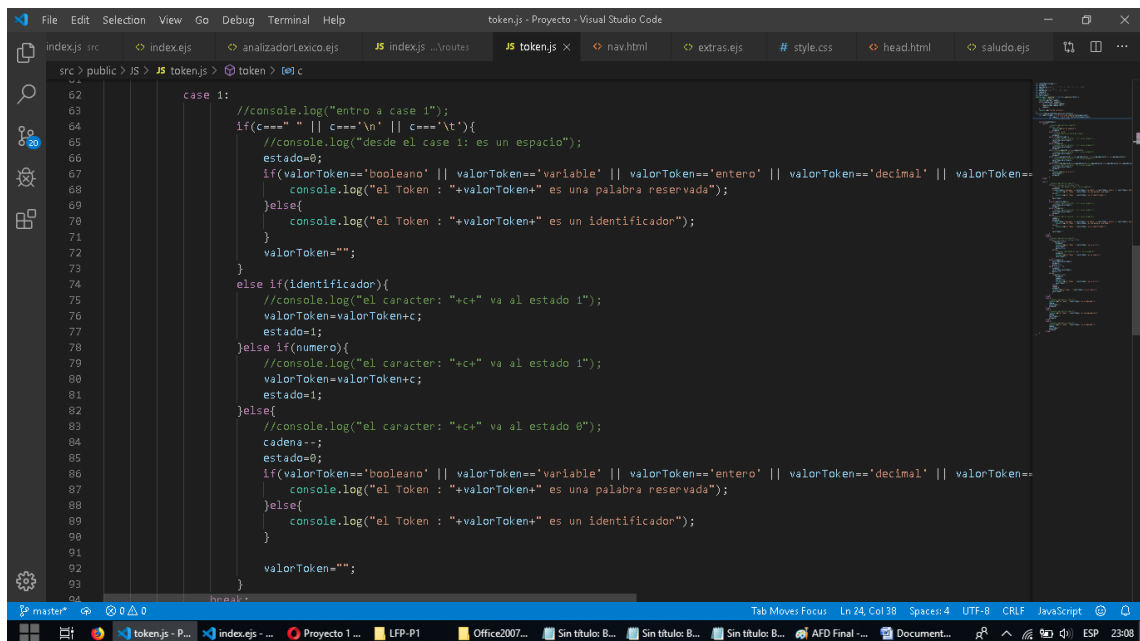
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
var tokenIdentificador="";
var estado=0;
var operadores = ['+', '-', '**', '/', '%', '=', '<', '>'];
var signos = ['"', '<', '>'];
var agrupacion = ['(', ')', '{', '}'];
var cadena="";
var punto=false;
var valorToken="";
module.exports.analizar = function analisis(texto) {
  var text = texto;
  console.log(text.length);
  while(cadena<text.length){
    var c = text.charAt(cadena);
    token(estado,cadena,text);
    cadena++;
  }
  console.log("fin de lectura");
}
function token(estadoToken, posicion, elTexto) { (parameter) elTexto: any
  var identificador = /^[a-zA-Z\s]+$/i.test(elTexto[posicion]);
  var numero = /^[0-9\s]+$/i.test(elTexto[posicion]);
  var c = elTexto[posicion];

  switch(estadoToken){
    case 0:
      //console.log("entro al case 0");
      if(c===' '){
        console.log("es un espacio");
        estado=0;
      }else if(c==='\\n'){
        //console.log("es un salto de líneas");
```



This screenshot shows the continuation of the `token.js` code from the previous image. It covers the `switch` statement for `estadoToken` from case 0 to case 5, and a final `else` block. The code handles spaces, newlines, identifiers, numbers, signs, and grouped expressions. It updates `valorToken` and `estado` based on the current character and state.

```
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
    case 0:
      //console.log("entro al case 0");
      if(c===' '){
        console.log("es un espacio");
        estado=0;
      }else if(c==='\\n'){
        //console.log("es un salto de líneas");
        estado=0;
      }else if(identificador){
        //console.log("el caracter: "+c+ " va al estado 1");
        valorToken=valorToken+c;
        estado=1;
      }else if(numero){
        //console.log("el caracter: "+c+ " va al estado 2");
        valorToken=valorToken+c;
        estado=2;
      }else if(c===signos[0] || c===signos[1]){
        //console.log("el caracter: "+c+ " va al estado 5");
        valorToken=valorToken+c;
        estado=5;
      }else if(c===agrupacion[0] || c===agrupacion[1] || c===agrupacion[2] || c===agrupacion[3]){
        //console.log("el caracter: "+c+ " va al estado 4");
        valorToken=valorToken+c;
        estado=4;
      }else if(c===operadores[0] || c===operadores[1] || c===operadores[2] || c===operadores[3] || c===operadores[4] || c===operadores[5]){
        //console.log("el caracter: "+c+ " va al estado 3");
        valorToken=valorToken+c;
        estado=3;
      }else{
        console.log("es un error");
        valorToken="";
        estado=0;
      }
    }
```



```
62 case 1:
63     //console.log("entro a case 1");
64     if(c==" " || c=="\n" || c=="\t"){
65         //console.log("desde el case 1: es un espacio");
66         estado=0;
67         if(valorToken=='booleano' || valorToken=='variable' || valorToken=='entero' || valorToken=='decimal' || valorToken==
68             console.log("el Token : "+valorToken+" es una palabra reservada");
69         }else{
70             console.log("el Token : "+valorToken+" es un identificador");
71         }
72         valorToken="";
73     }
74     else if(identificador){
75         //console.log("el caracter: "+c+" va al estado 1");
76         valorToken=valorToken+c;
77         estado=1;
78     }else if(numero){
79         //console.log("el caracter: "+c+" va al estado 1");
80         valorToken=valorToken+c;
81         estado=1;
82     }else{
83         //console.log("el caracter: "+c+" va al estado 0");
84         cadena--;
85         estado=0;
86         if(valorToken=='booleano' || valorToken=='variable' || valorToken=='entero' || valorToken=='decimal' || valorToken==
87             console.log("el Token : "+valorToken+" es una palabra reservada");
88         }else{
89             console.log("el Token : "+valorToken+" es un identificador");
90         }
91     }
92     valorToken="";
93 }
```

Esta clase llamada token contiene toda la logica de programacion del analizador lexico, su funcionamiento consisten en leer caracter por caracter, verifica que tipo de caracter es segun el alfabeto y lo envia a un case especifico para su analisis.

Luego devuelve el token evaluado indicanto el tipo de token que es.