

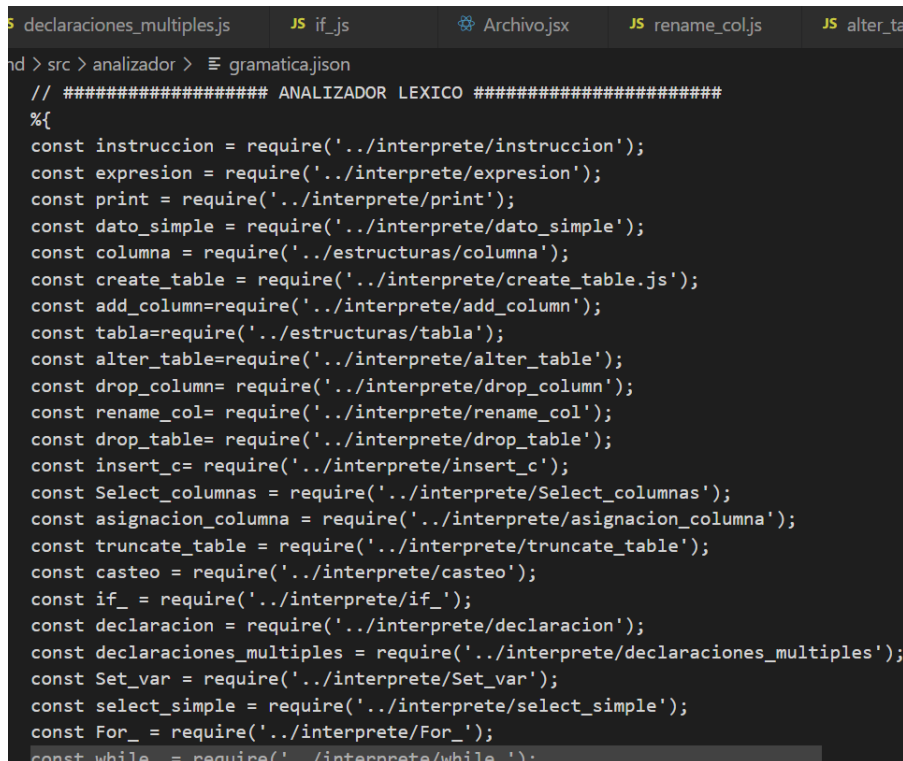
# Manual técnico

Para esta aplicación se usaron diferentes herramientas como el entorno de desarrollo de NODE JS, como el entorno de REACT, esto se hizo con el fin de aprender a usar JISON, ya que para este proyecto fue requisito el uso de esta herramienta.

A continuación, se mostrarán algunas de las formas de la que se realizó esta aplicación.

Herramientas usadas:

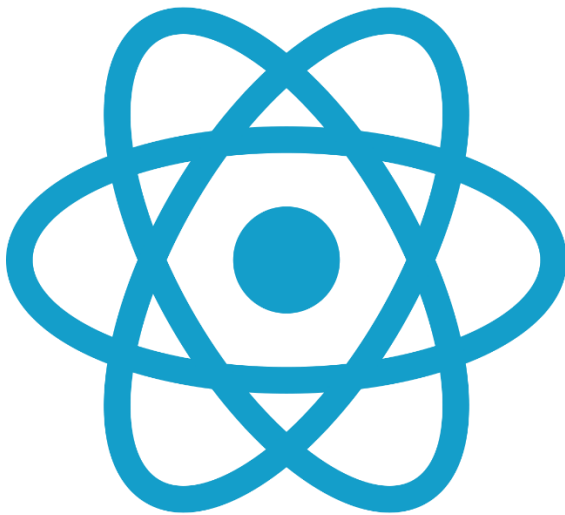
**JISON:**

A screenshot of a code editor with a dark theme. The editor has several tabs at the top: 'declaraciones\_multiples.js', 'JS if\_js', 'Archivo.jsx', 'JS rename\_col.js', and 'JS alter\_ta'. The active tab is 'gramatica.jison'. The code in the editor is a JISON grammar file. It starts with a comment '// ##### ANALIZADOR LEXICO #####'. Below the comment, there is a block comment '%{' followed by a series of 'const' declarations for various functions. These functions are required from other files in the project, such as './interprete/instruccion', './interprete/expresion', './interprete/print', './interprete/dato\_simple', './estructuras/columna', './interprete/create\_table.js', './interprete/add\_column', './estructuras/tabla', './interprete/alter\_table', './interprete/drop\_column', './interprete/rename\_col', './interprete/drop\_table', './interprete/insert\_c', './interprete/Select\_columnas', './interprete/asignacion\_columna', './interprete/truncate\_table', './interprete/casteo', './interprete/if\_', './interprete/declaracion', './interprete/declaraciones\_multiples', './interprete/Set\_var', './interprete/select\_simple', and './interprete/For\_'. The code ends with 'const while = require('./interprete/while');'.

## NODE JS



## REACT



Para este proyecto se utilizó patrón de diseño que consiste en una solución general y reutilizable para un problema común que se encuentra al desarrollar software. Estos patrones representan las mejores prácticas probadas y se han desarrollado a lo largo del tiempo a medida que los programadores han enfrentado y resuelto problemas similares.

En esta ocasión se utilizó la clase instrucción como clase abstracta

Ya que las demás instrucciones heredarían sus métodos, para agilizar el desarrollo de la aplicación como se muestra a continuación:

```
1 / src / interprete / > > > instruccion.js / > > > instruccion / > > > getAst
class instruccion{
  constructor (){}
  ejecutar(entorno){}
  getAst(){
    let nodo = {
      padre: -1,
      cadena: ""
    }
    return nodo;
  }
}
```

```
module.exports=instruccion;
```

Debug Console (Ctrl+Shift+Y)

Así se creo la clase padre, y esta heredo sus métodos:

```
const entorno = require('../tabla_simbolos/entorno');
const instruccion = require('../interprete/instruccion');
const clase_ast=require('../ast/clase_ast');
class For_ extends instruccion{
  constructor(ID,rango,instrucciones){
    super();
    this.ID=ID;
    this.rango=rango;
    this.instrucciones=instrucciones;
    this.entorno=new entorno("for",null);

  }
}
```

#### REFERENCIAS DE DESARROLLO:

[AlexIngGuerra/OLC1-2S2023: Ejemplos del Curso OLC1-2S2023 \(github.com\)](https://github.com/AlexIngGuerra/OLC1-2S2023)

[https://www.youtube.com/@diians\\_2302/](https://www.youtube.com/@diians_2302/)