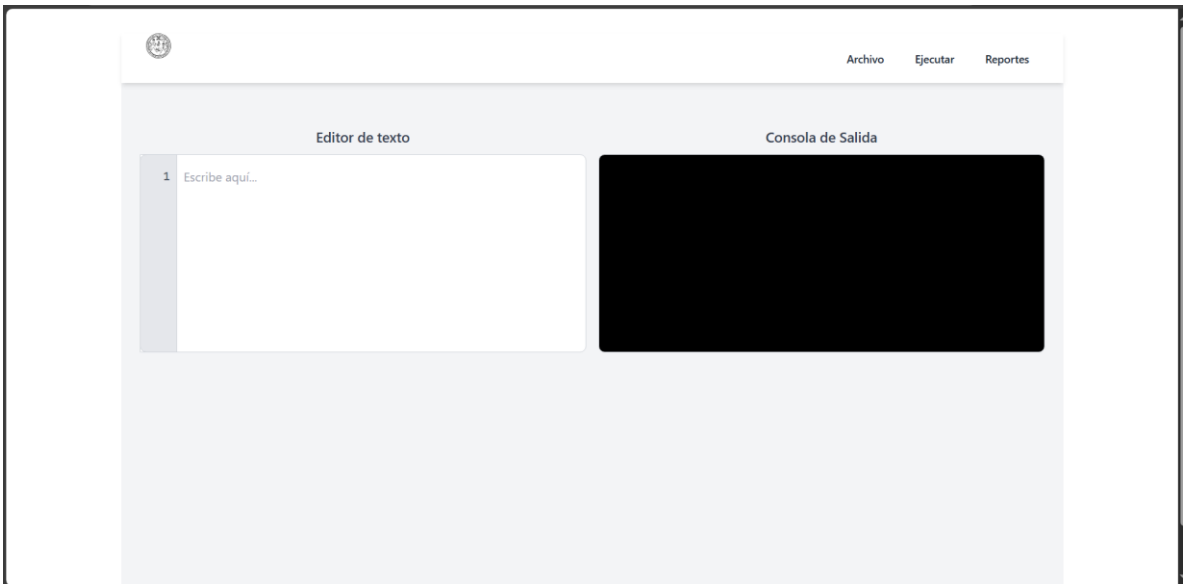
Manual Técnico

El programa fue desarrollado en lenguaje Javascript, el IDE utilizado fue Visual studio Code. Consiste en un Analizador tanto léxico somo sintactico que es capaz de realizar operaciones tanto aritméticas,relacionales y logicas, así como instrucciones como if, for, while, Do-while. Su funcionamiento consiste en una interfaz grafica, donde el Area de la ENTRADA se ingresa el código a analizar y la SALIDA donde se mostraran las impresiones en consola. Para poder realizar el análisis primero se debe de crear un archivo vacio o abrir un archivo con extensión .sc y al presionar la opción analizar se realizara el análisis del código de la entrada y los resultados se mostraran en SALIDA. Se tienen botones para poder generar los diversos reportes del código analizado. La interfaz grafica es la siguiente:



El analizador cuenta con recuperación de errores en modo de pánico el cual al detectar un error lo elimina hasta que encuentra uno que si pertenece a la gramática. Ambos analizadores están implementados para ignorar saltos de línea, espacios en blanco, comentarios de una línea y comentarios multilínea por lo que no los tomara en cuenta en el análisis del archivo de entrada El analizador cuenta con reporte de árbol AST y reporte de errores, en el cual se generará una tabla en la parte de debajo de los botones con los simbolos detectados y otra tabla con los errores tanto léxicos como sintácticos y semánticos que en analizador encuentre. Todas las funciones agregadas a las producciones se encuentran ordenadas por carpetas, una para instrucciones, expresiones y otra para funciones. Se aplico el patron interprete para ir analizando cada parte que vaya detectando el análisis sintactico. Cada producción

generara los nodos que posteriormente se graficaran en el reporte de árbol AST. Para la graficacion del Árbol AST se utilizaron las siguientes funciones:

```
const fs = require('fs');
const { exec } = require('child_process');

function graficarArbol(arbolitos) {
    global.cuerpo = '';
    global.contador = 0;

    global.contador = 1;
    global.cuerpo = '';
    graphAST('n0', arbolitos);
    let principal = `digraph arbolAST{
        n0[label="${arbolitos.valor.replace('"', '\\"')}"];
        ${cuerpo}
    }`;
    fs.writeFile('arbolAST.dot', principal, () => {});
    exec('dot -Tjpg arbolAST.dot -o arbolAST.jpg', (error, stdout, stderr) => {
        if (error) {
            return;
        }
        if (stderr) {
            return;
        }


        exec('start arbolAST.jpg', (openError, openStdout, openStderr) => {
            if (openError) {
                console.error('Error al abrir la imagen:', openError);
            }
        });
    });
    return principal;
}

function graphAST(texto, padre) {
    for (let hijo of padre.listaHijos) {
        let nombreHijo = `n${global.contador}`;
        global.cuerpo += `${nombreHijo}[label="${hijo.valor.replace('"', '\\"')}"];
        ${texto} -> ${nombreHijo};\n`;
        global.contador++;
        graphAST(nombreHijo, hijo);
    }
}
```