08 / 05 / 2021

MADELINE ARIANA PÉREZ QUIÑÓNEZ 201807117

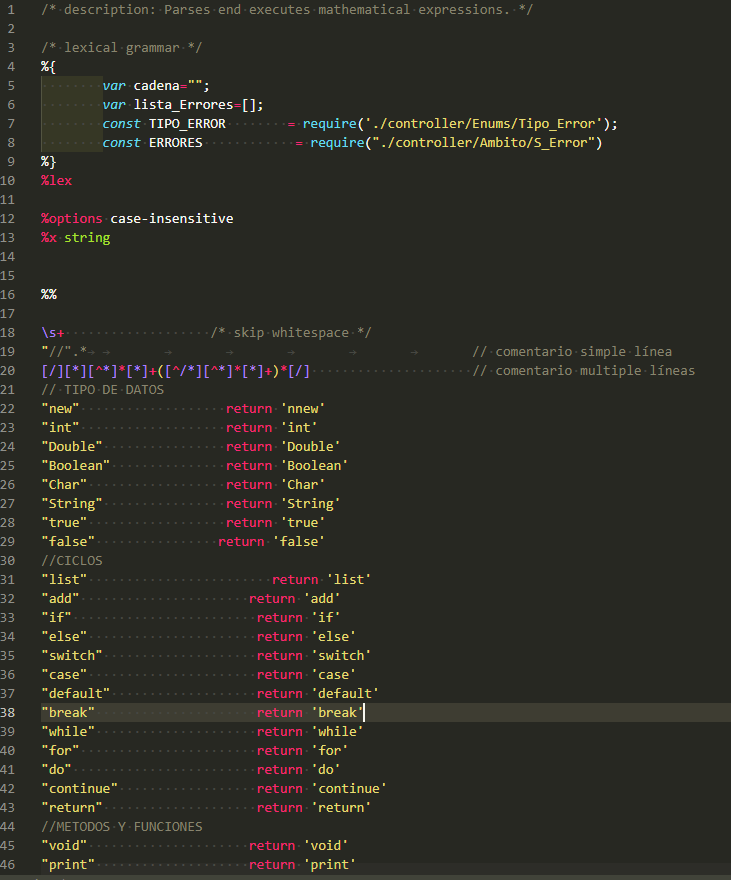
Manual Técnico

OLC 1 Sección “B”

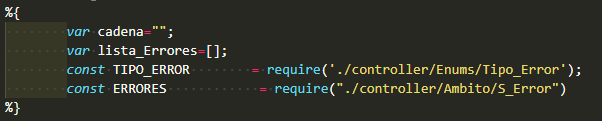
# analizdor.jison

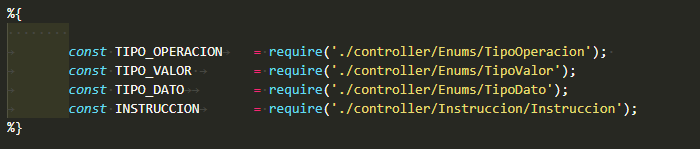
## Analizador lexico y sintactico

En esta parte se declaran los tokens a utilizar el lenguaje.

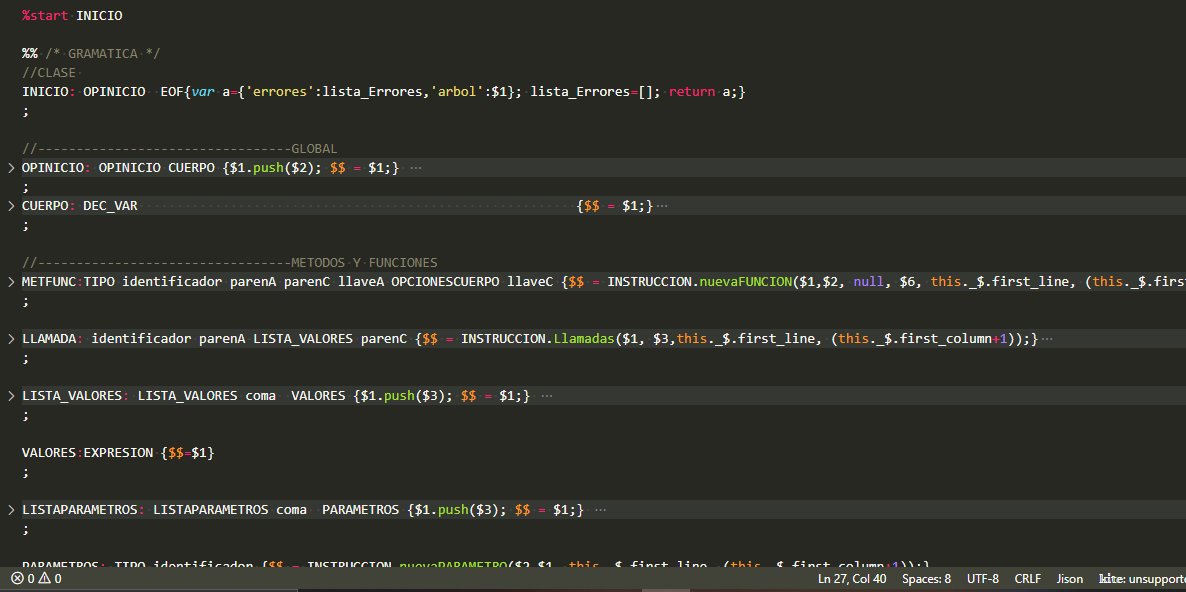


En esta parte se declaran variables y se colocan las importaciones a utilizar para crear el árbol y lista de errores.





Se elabora el analizador sintáctico. Al ya tener todo eso, ya se empieza a estructurar la gramática. (Explicación de la gramática en el archivo “Gramática\_201807117.pdf”)



# index.js

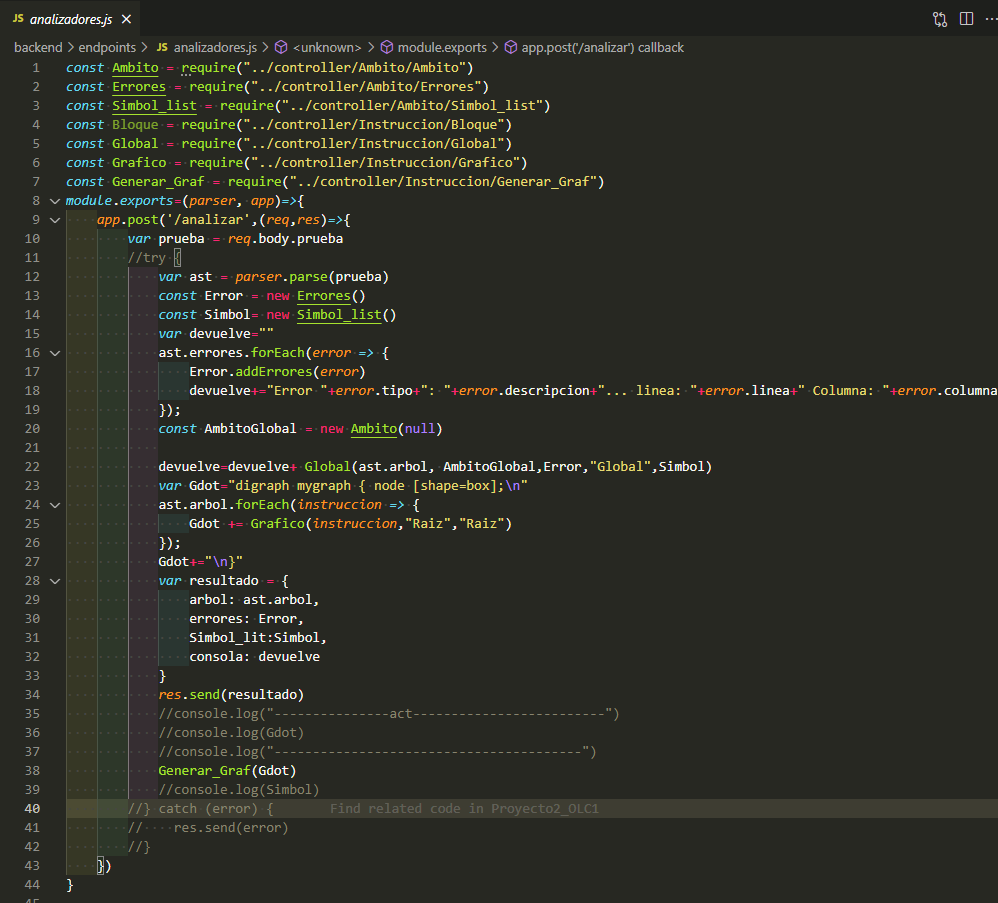
En esta parte se declaran las rutas del api.



# analizador.js

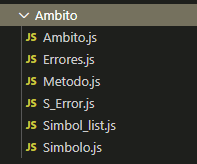
Aquí se manda a llamar es en donde:

* La entrada se envía al analizador jison
* Revisa si no hay errores léxicos y sintácticos
* Envía el árbol creado por el jison a el Ambito global para empezar el análisis semántico
* Manda a llamar una función para la creación del árbol AST
* Y por último envía la información al cliente



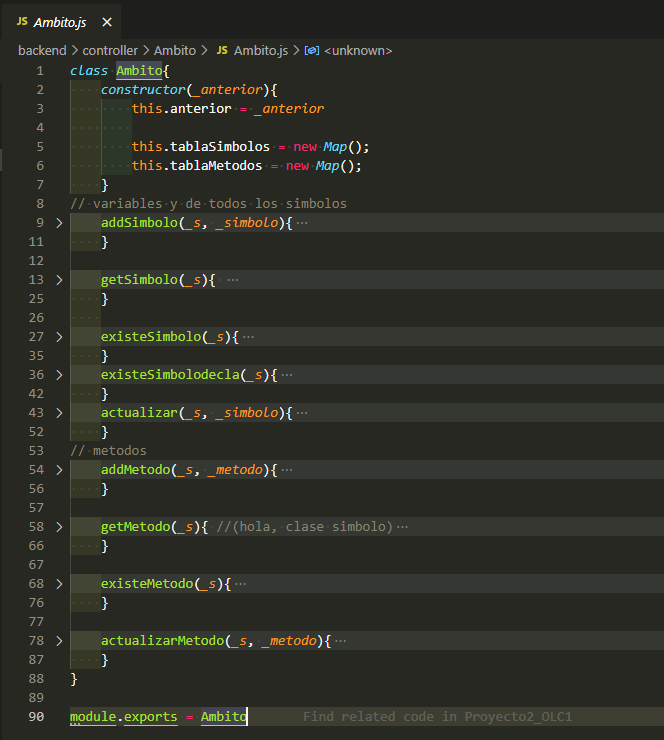
# ambito

En esta carpeta están los archivos que se utilizaran para guardar símbolos, errores y métodos.



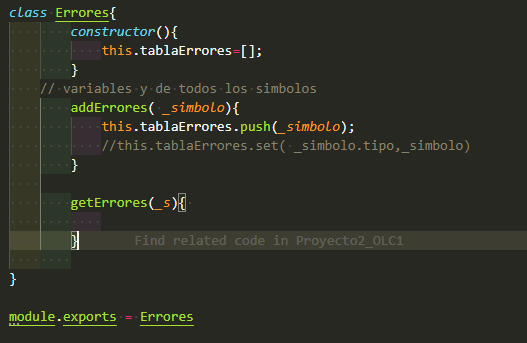
## ambito.js

Tiene los métodos para almacenar y acceder a los metodos y símbolos que se ingresen, según su entorno.



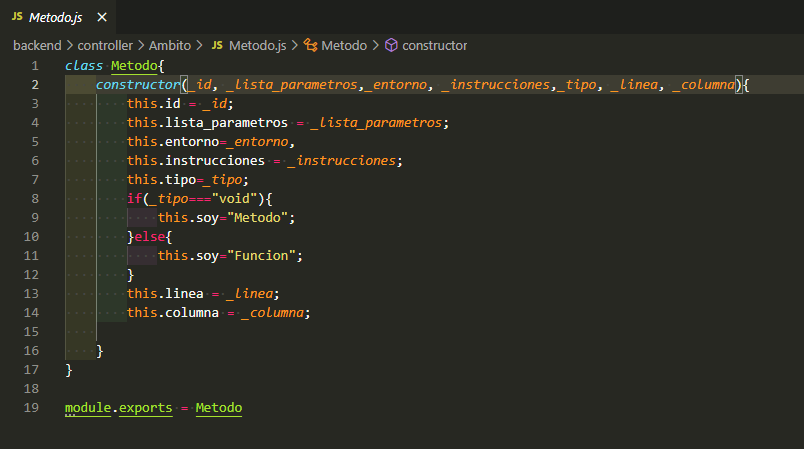
## errores.js

Tiene los métodos para almacenar los errores léxicos, semánticos y sintácticos.



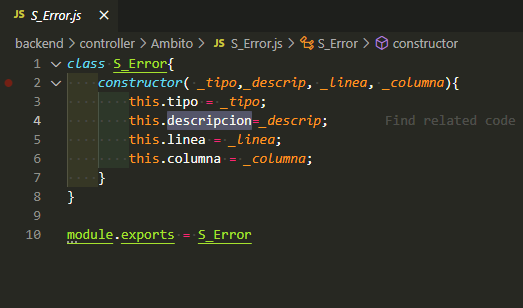
## metodo.js

Tiene la estructura con la que se almacenan los metodos.



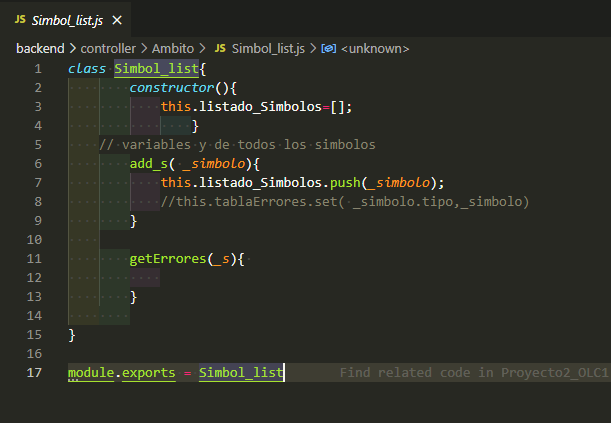
## S\_error.js

Tiene la estructura con que se guardaran los errores.



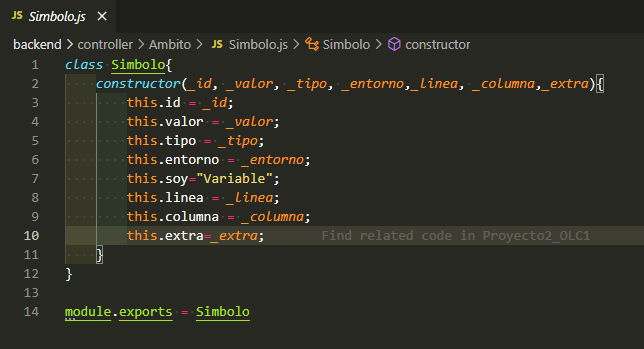
## Simbol\_list.js

Tiene una lista en donde se guardarán todos los símbolos (métodos y variables). Esta servirá para la tabla de símbolos que se muestra en el cliente.



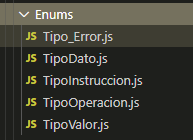
## simbolo.js

Tiene la estructura con que se guardaran las variables.



# enums

Aquí es en donde se declara los tipos de errores, tipos de datos, tipos de instrucciones, tipo de operaciones y tipo de valores.



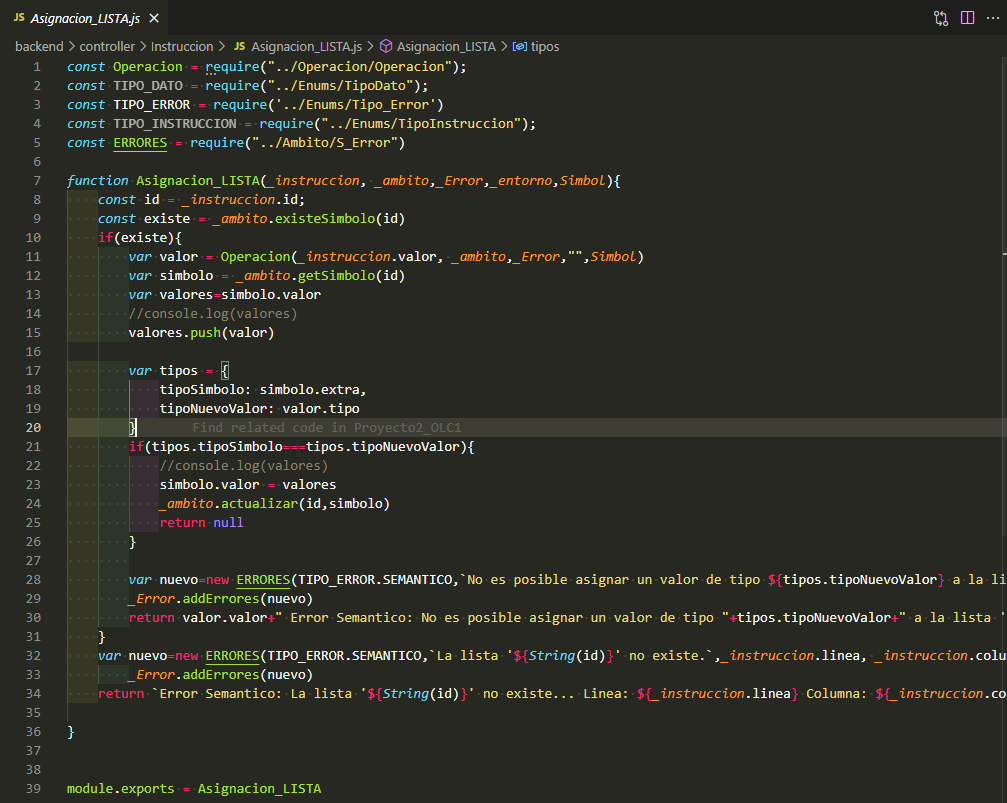
# instrucción

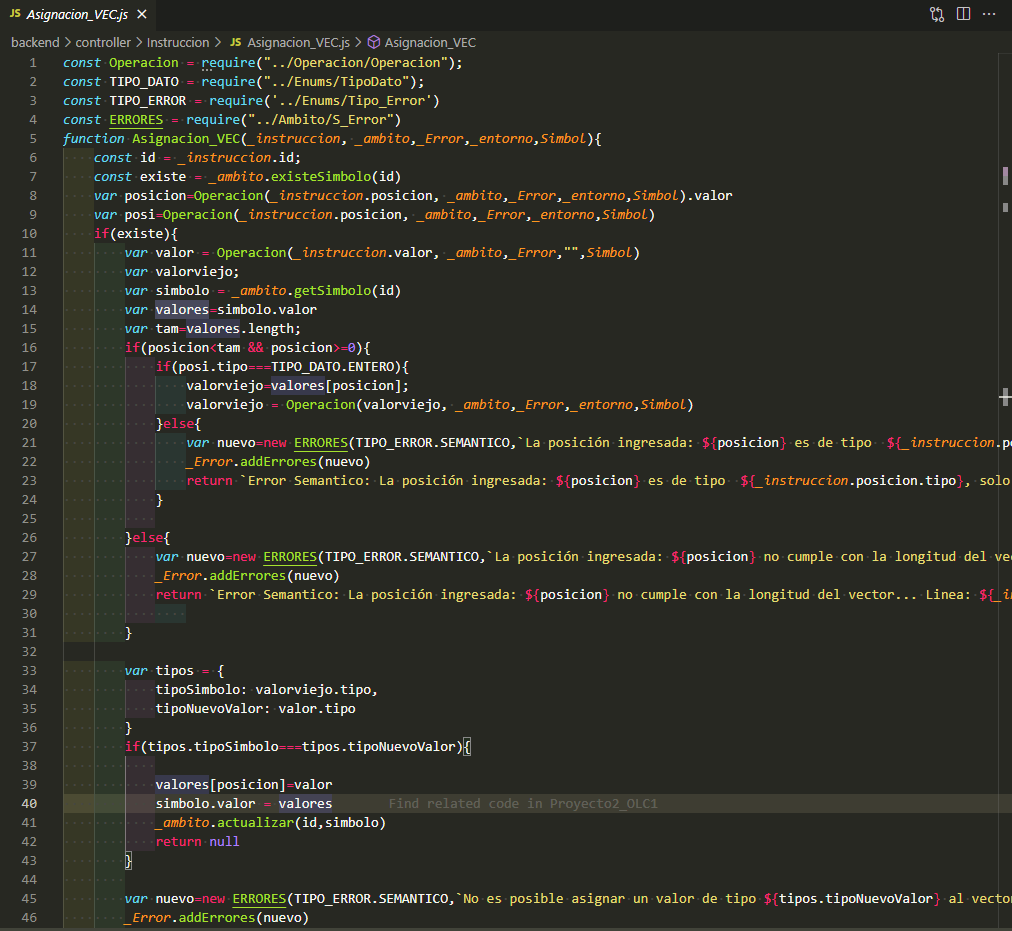
Esta carpeta almacena todos las instrucciones que se ejecutaran al recorrer el árbol que genera el analizador jison



## Asignacion\_LISTA.js y Asignacion\_VEC.js

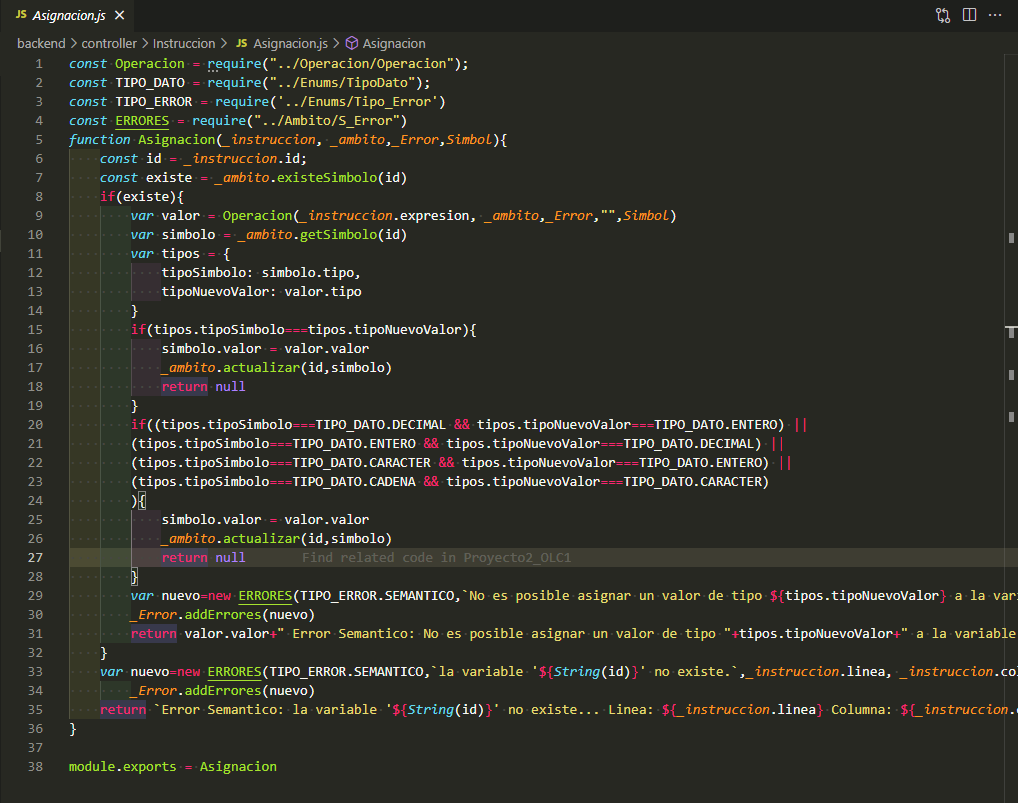
Este archivo contiene el proceso de la asignación de valores para las posiciones de los vectores y de las listas.





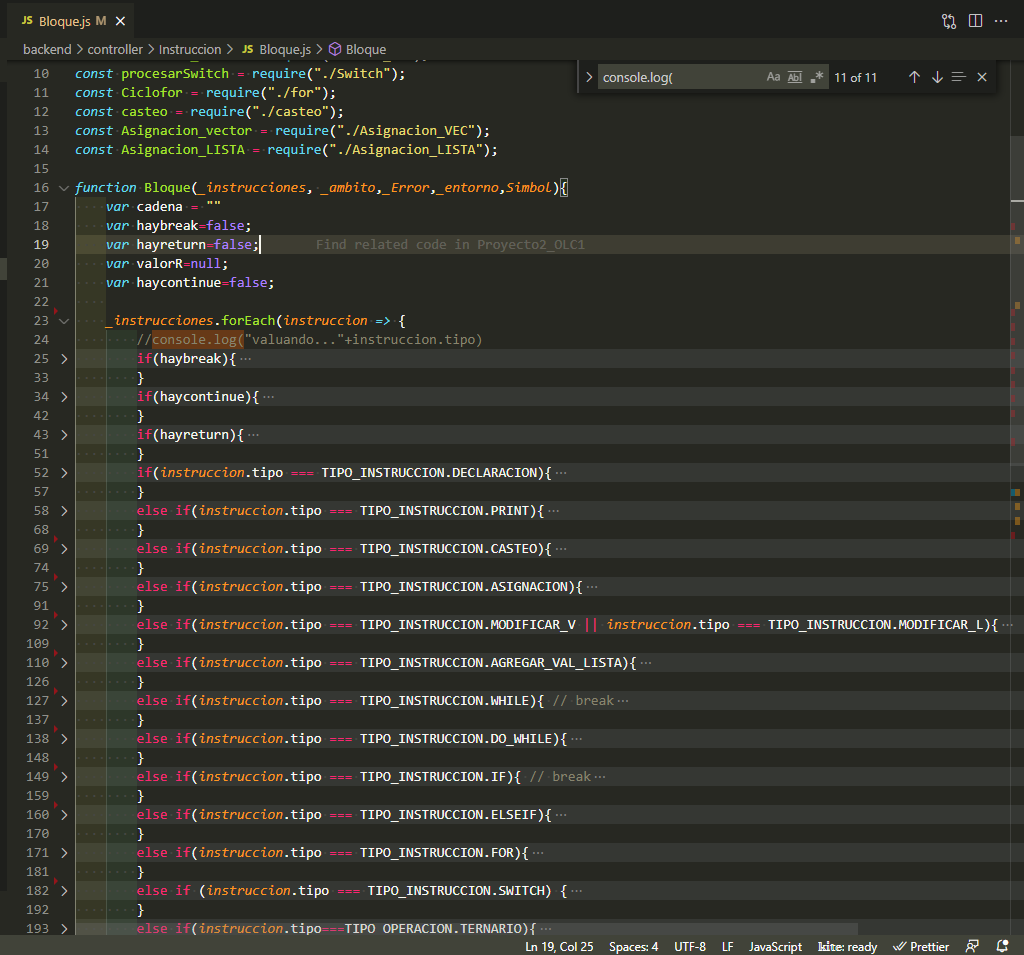
## asignacion.js

Este archivo contiene el proceso de la asignación de valores a las variables declaradas en el ámbito.

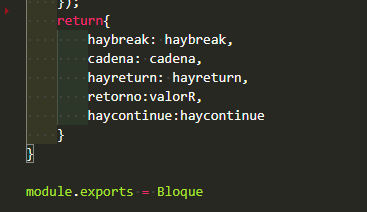


## bloque.js

Este archivo contiene un ciclo que recorre todas las instrucciones que trae el árbol.

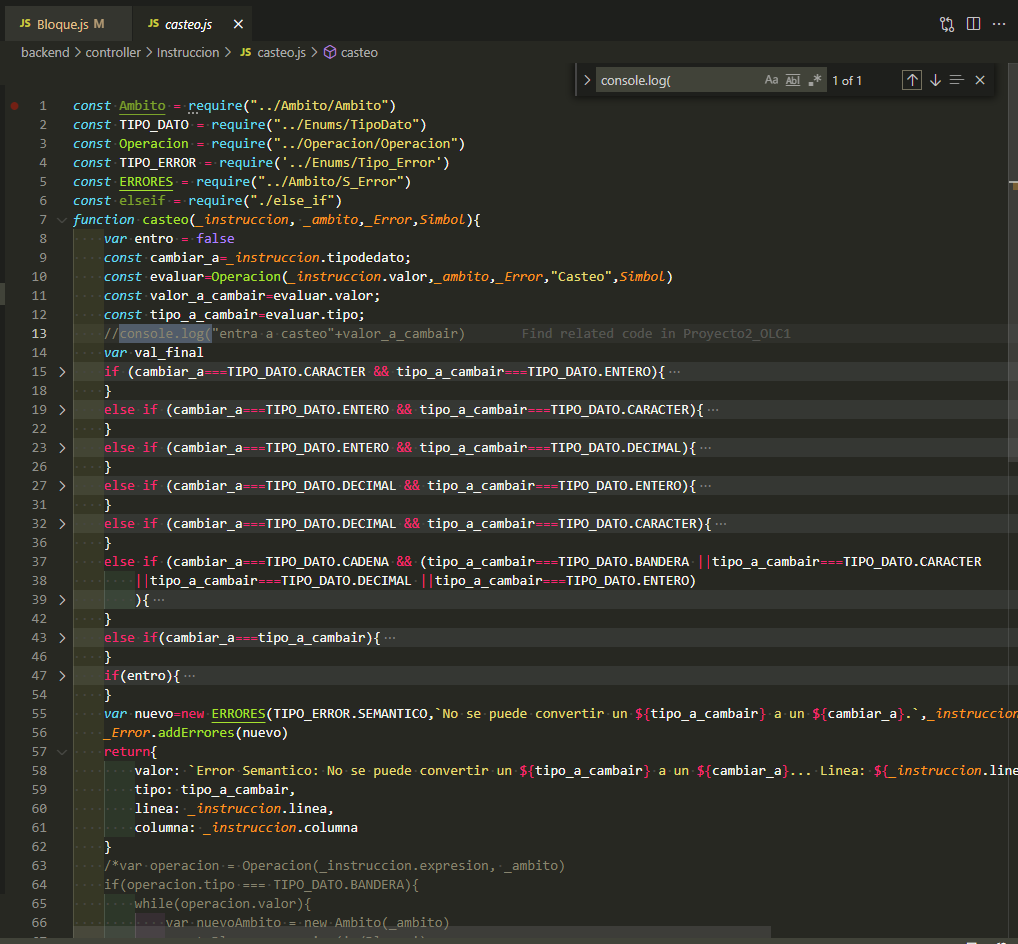


Y así al terminar el recorrido retornara una cadena y los valores del continue, return y break.



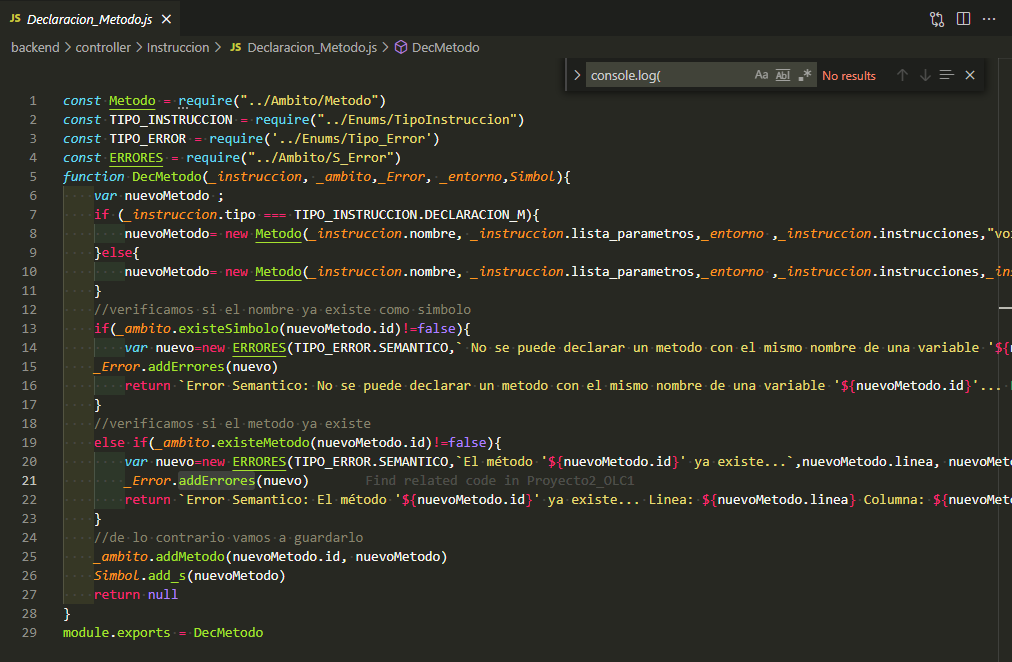
## casteo.js

Este archivo contiene el proceso y restricciones del casteo.



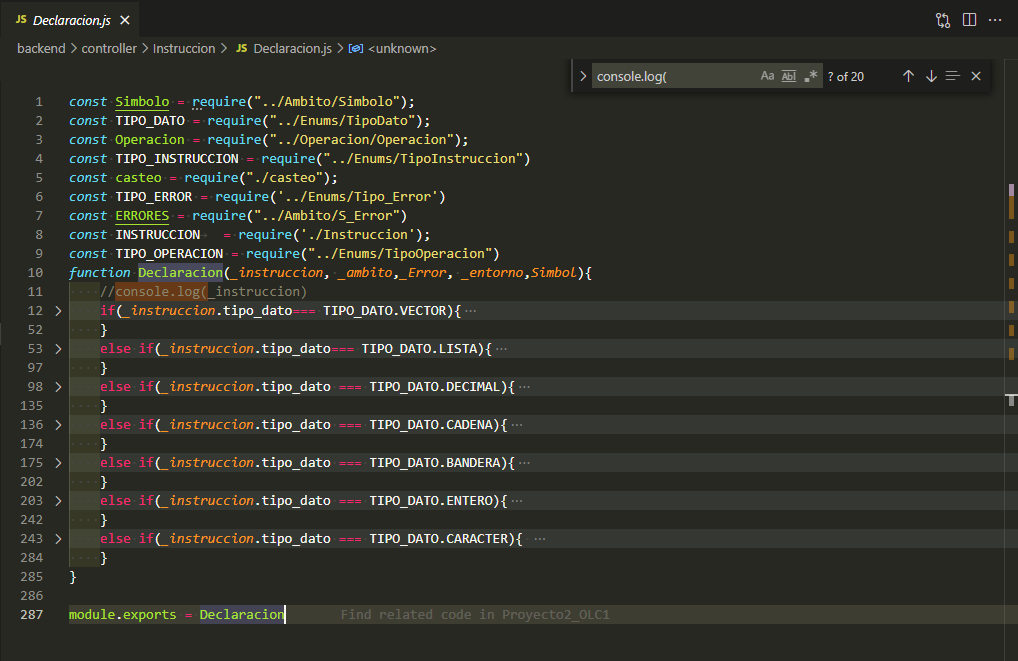
## decmetodo.js

Este archivo contiene el proceso de la declaración de métodos y funciones.



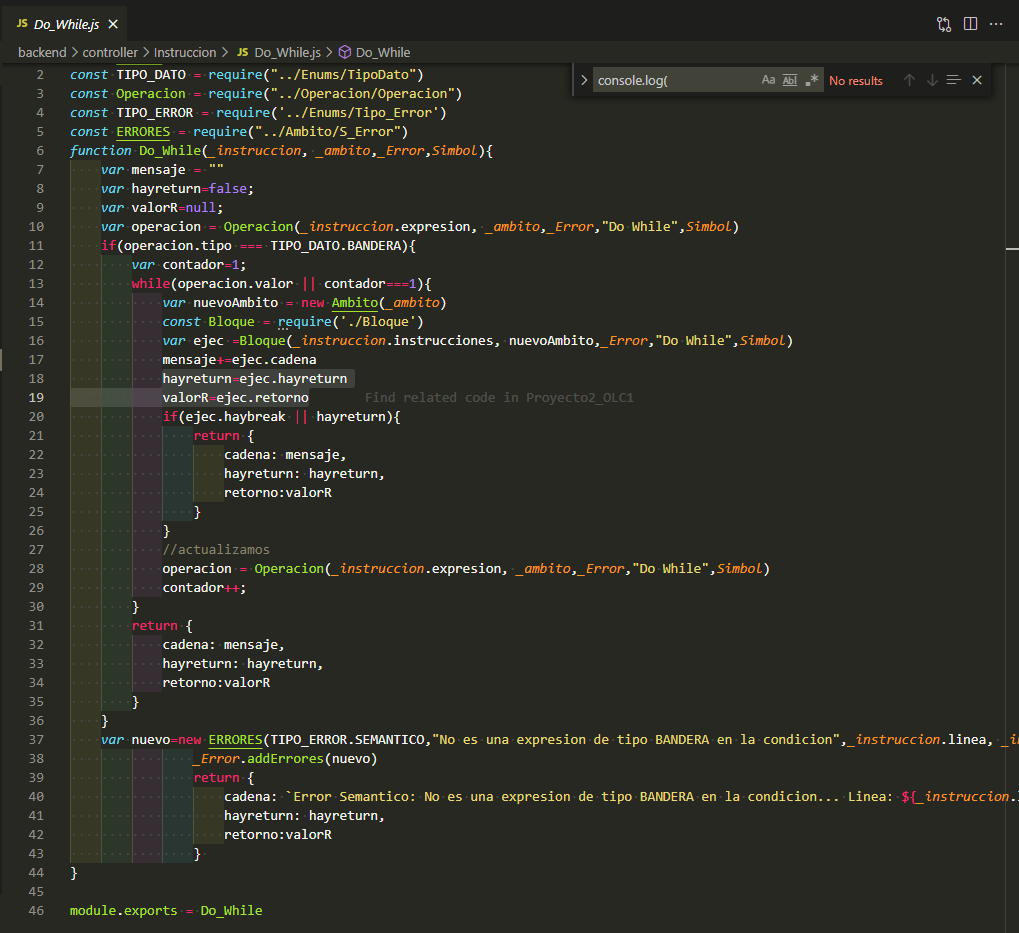
## Declaracion.js

Este archivo contiene el proceso de la declaración de variables, listas y vectores, con y sin valor.



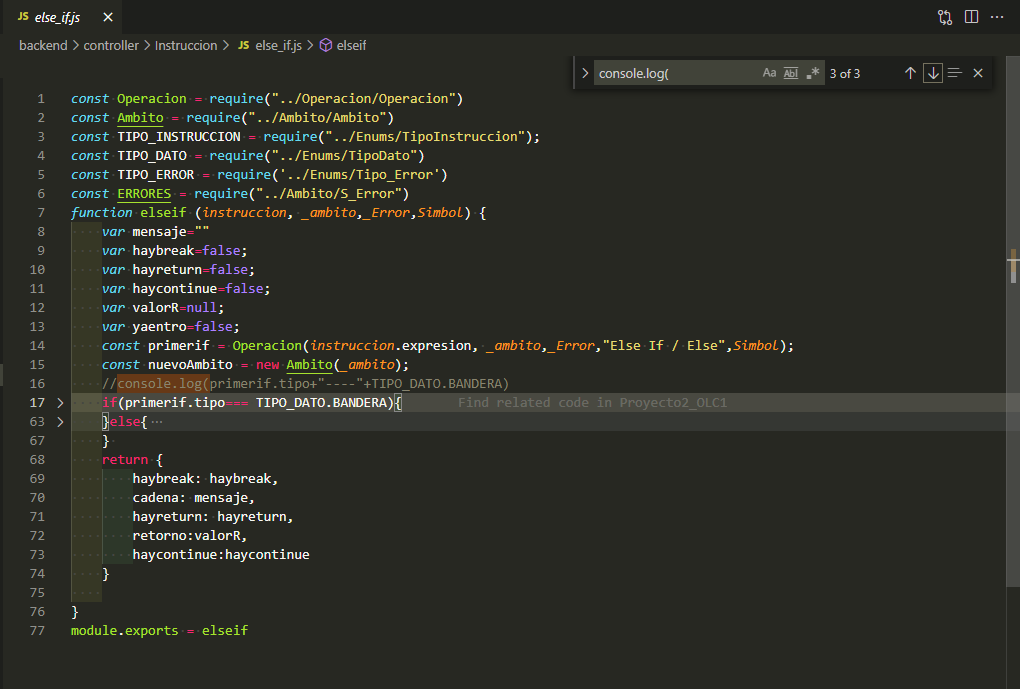
## do\_while.js

Este archivo contiene el proceso del ciclo do while, teniendo como restricción que mínimo una vez se debe ejecutar el cuerpo.



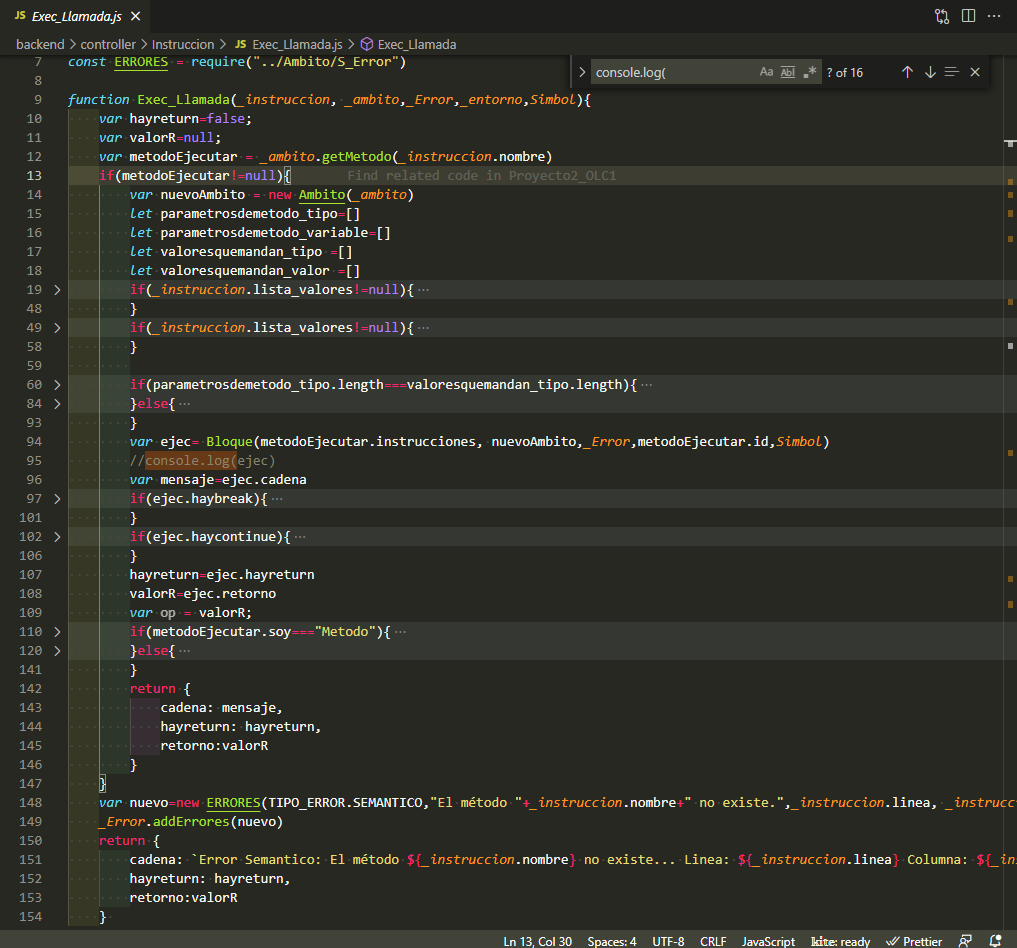
## Else\_if.js

Este archivo contiene el proceso de la sentencia de control else if. Que contiene un if y una lista de else if y else… que tiene como condición que al entrar una sentencia ya no podrá entrar a otra.



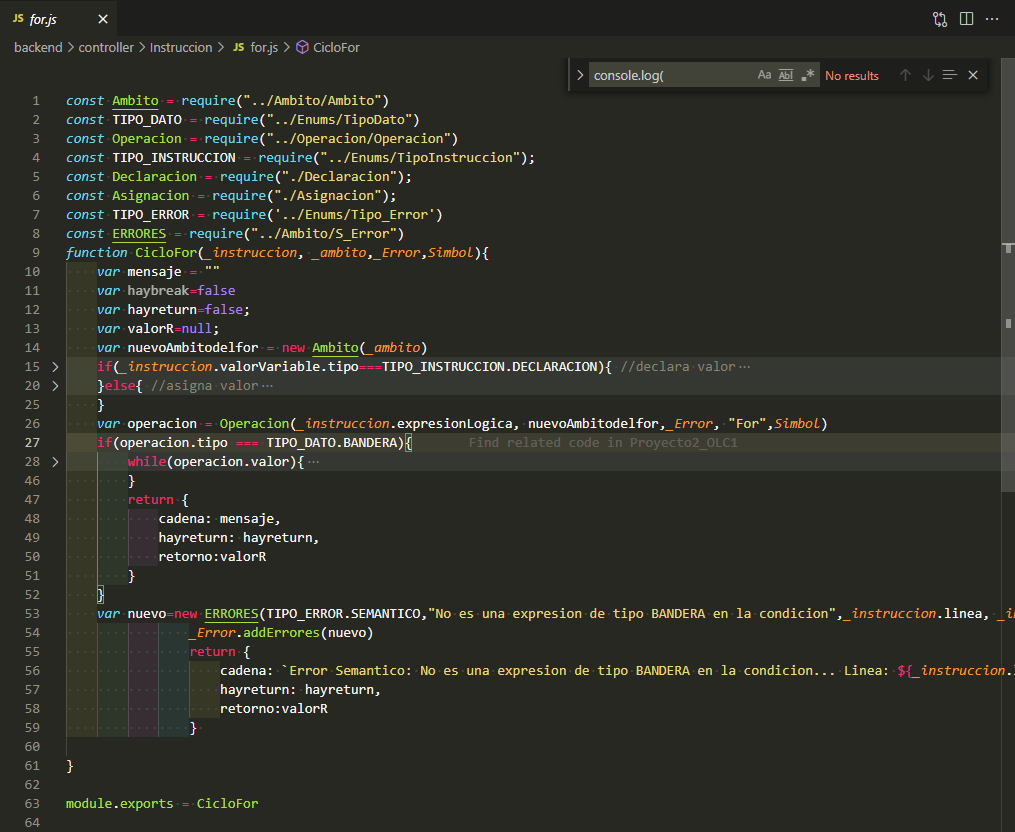
## exec\_llamada.js

Este archivo contiene el proceso de la llamada de un método o función, en donde se declaran los parámetros y sus valores por medio de listas y verificaciones.



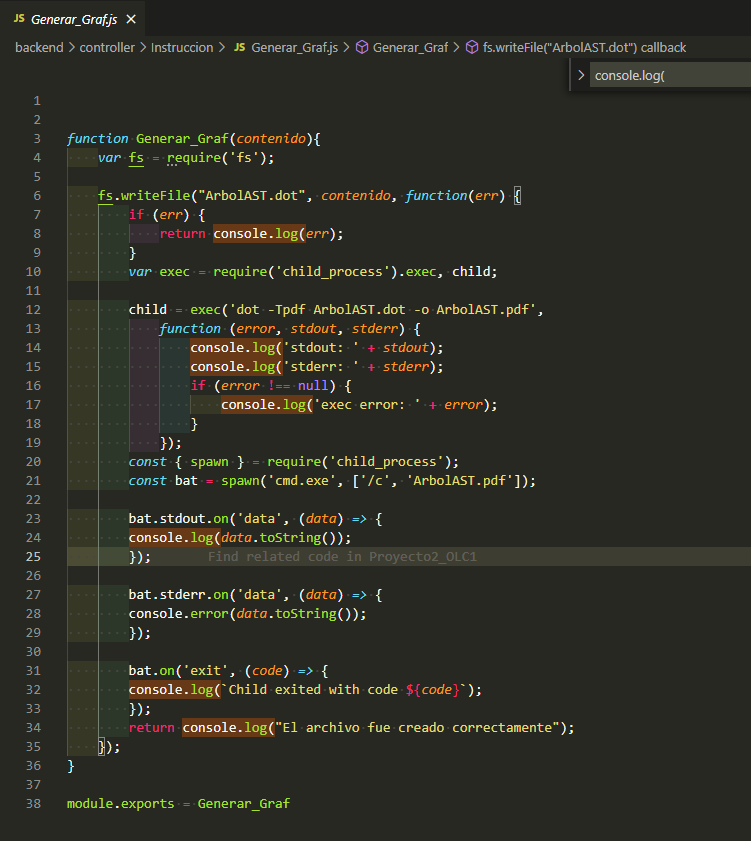
## For.js

Este archivo contiene el proceso del ciclo for, se hace desde un while y se va actualizando la condición, y hasta que se deja de cumplir la condición sale y retorna los valores necesarios.



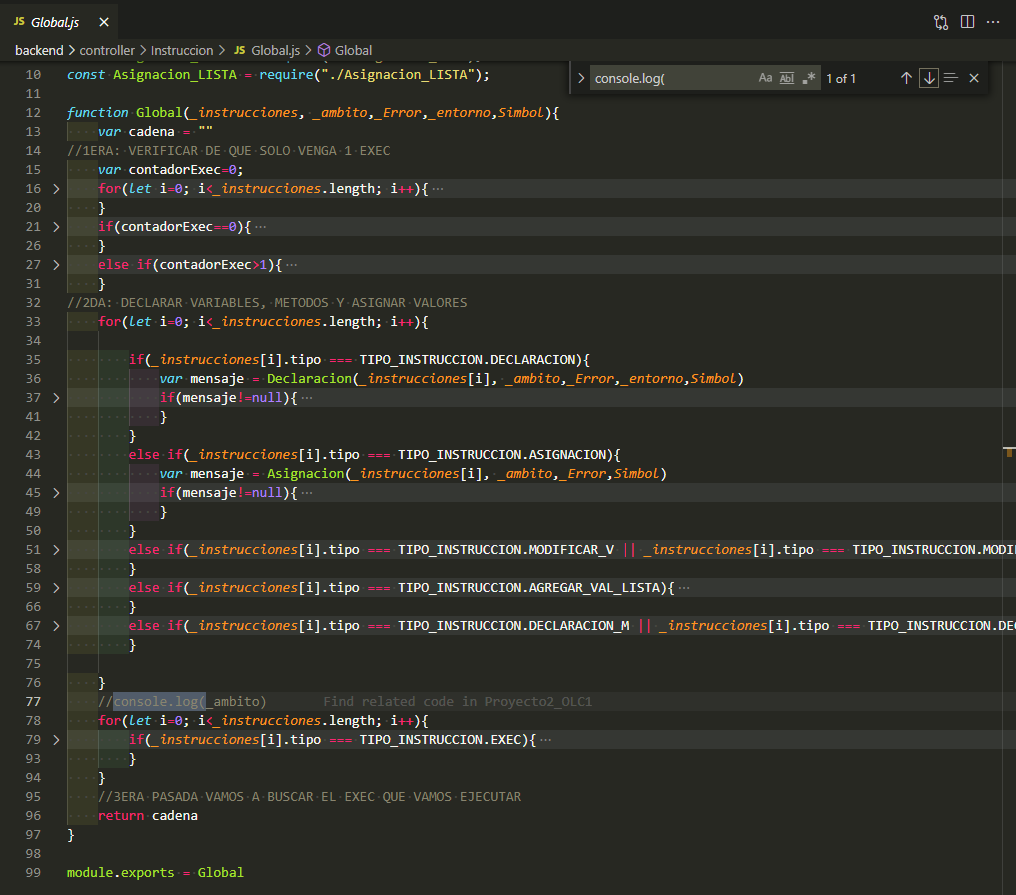
## Generar\_graf.js

Este archivo contiene el proceso de generar el dot, convertirlo a pdf y al final abrir el pdf en el navegador.



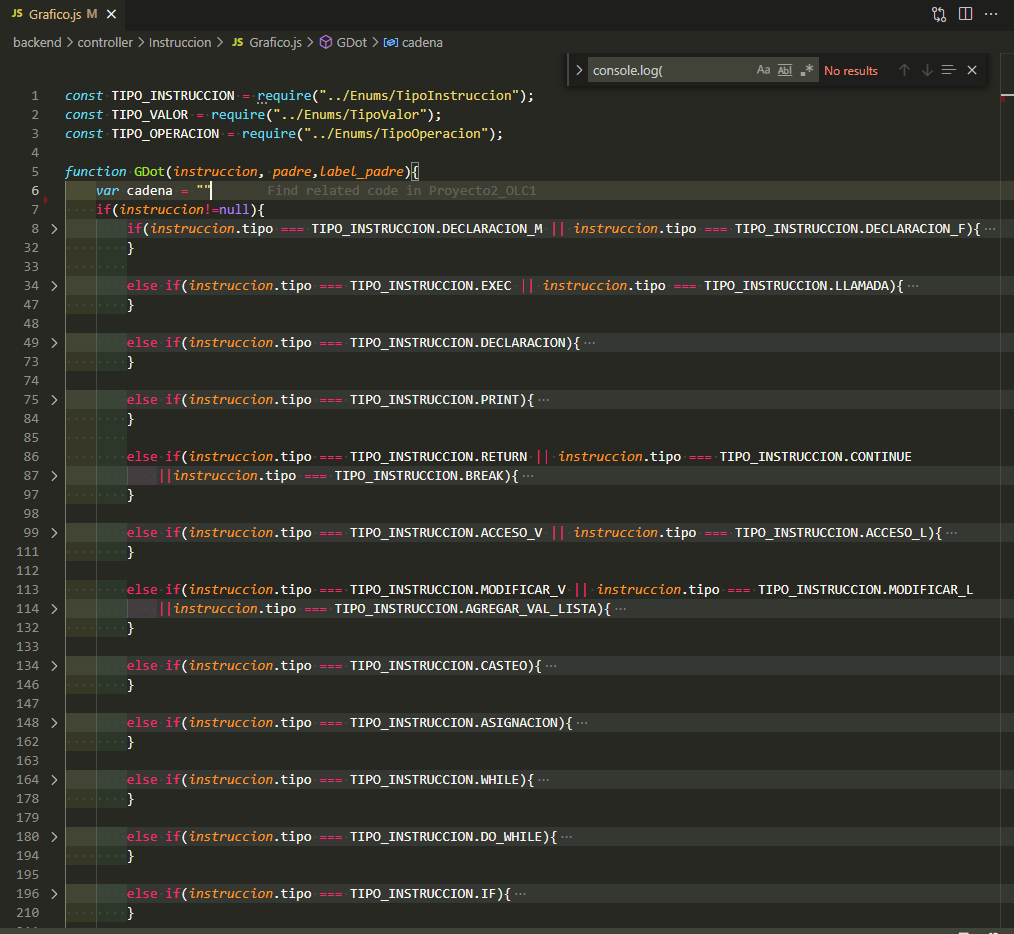
## Global.js

Este archivo contiene el proceso del verificar la cantidad de funciones exec, declarar variables globales y metodos y por ultimo mandar a llamar la clase de llamadas para ejecutar el método del exec.



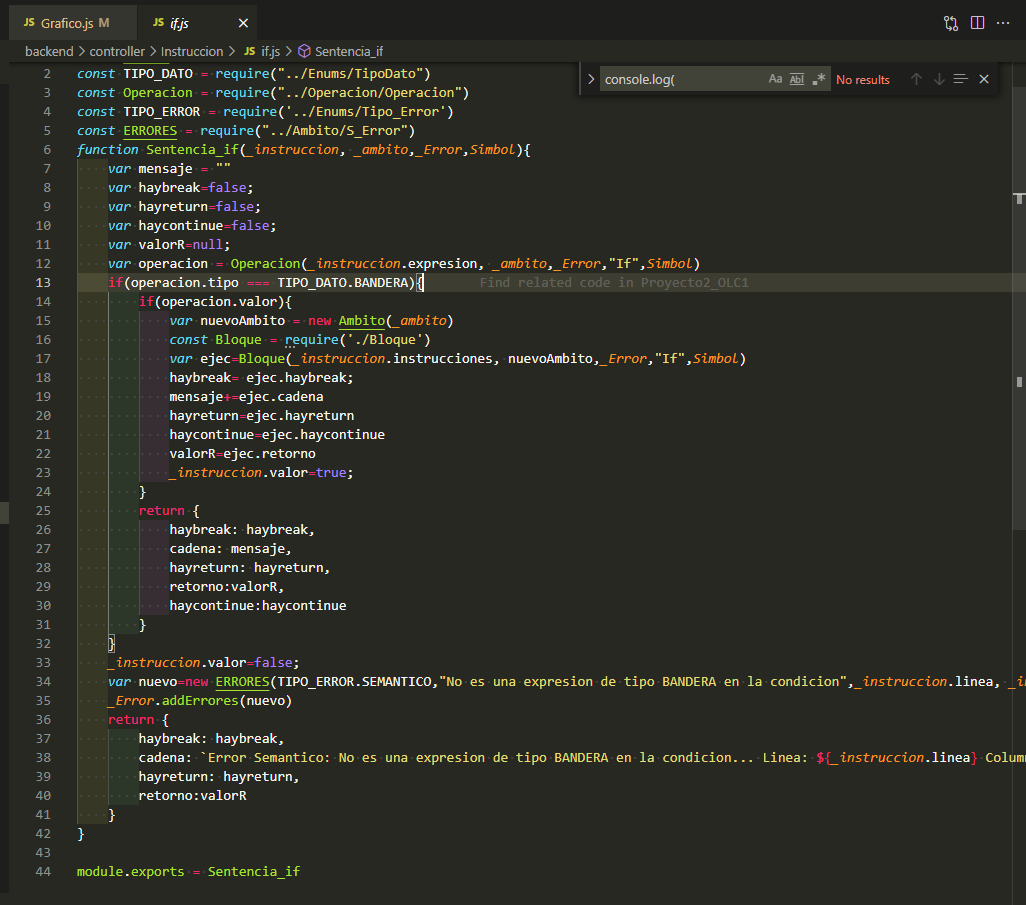
## grafico.js

Este archivo contiene el proceso de la creación del dot para el árbol AST. Lo hace en base al árbol que genera el analizador jison.



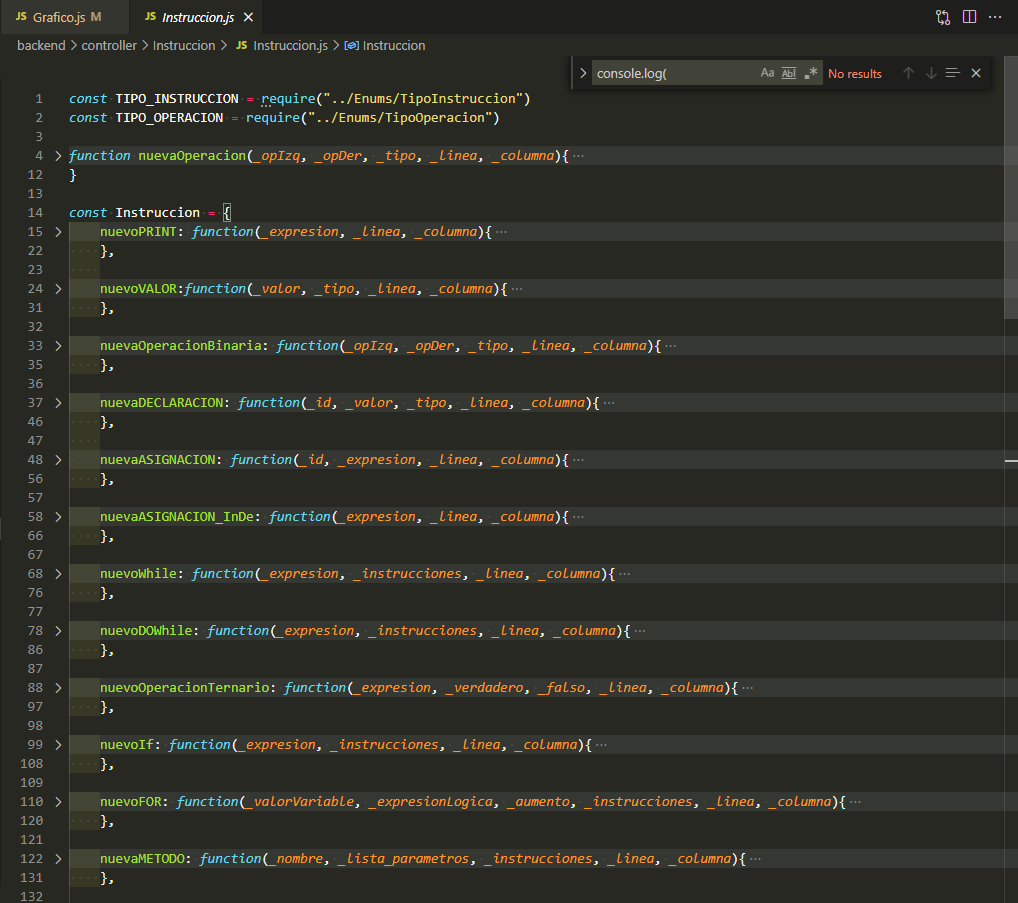
## if.js

Este archivo contiene el proceso de la sentencia de control if que valida si la condición es verdadera, entrara a ejecutar las instrucciones contenidas, de lo contrario no retornara nada.



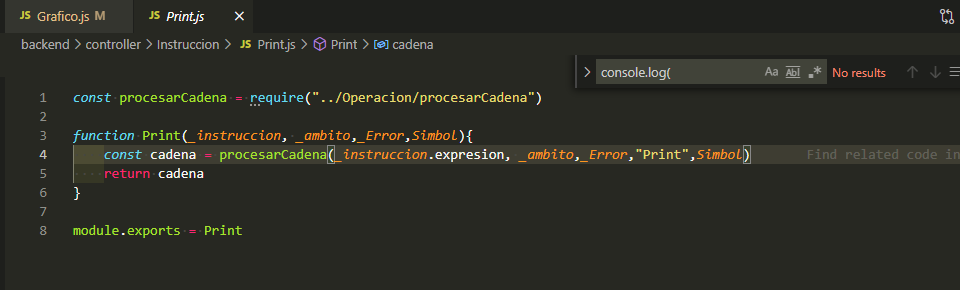
## Instrucción.js

Este archivo contiene todas las estructuras que traerá cada producción del analizador jison.



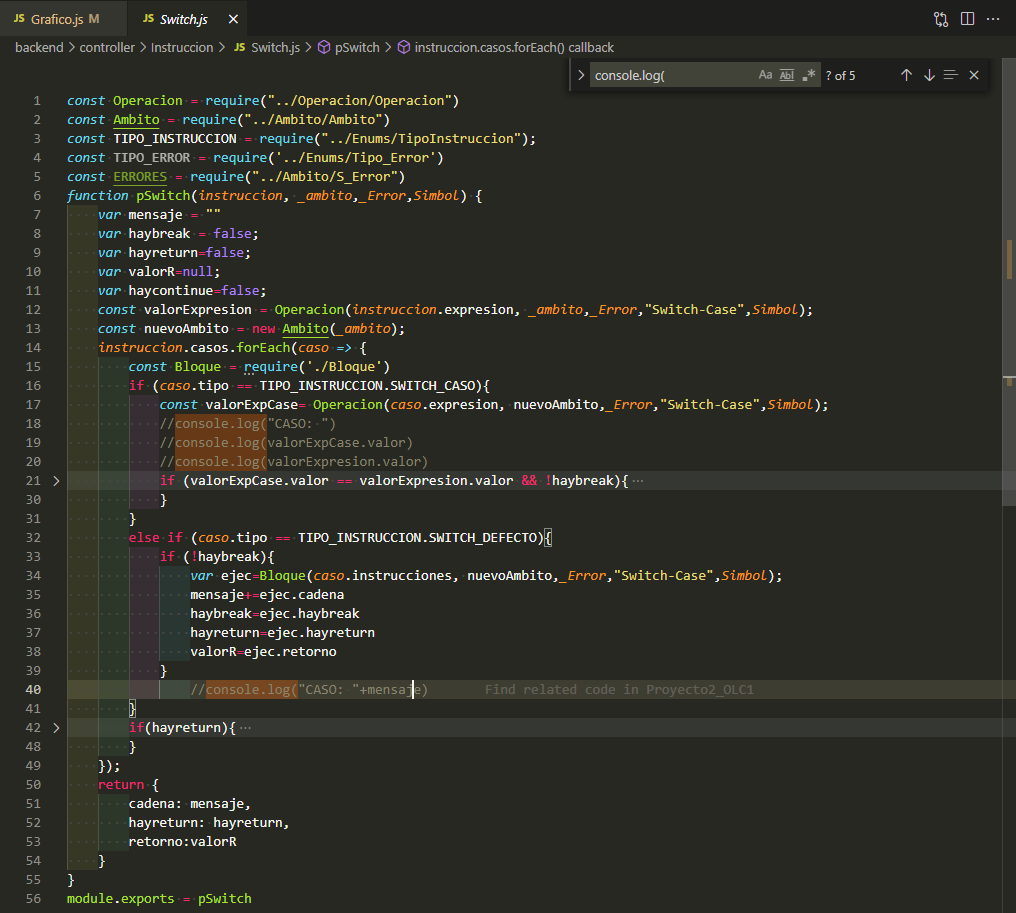
## Print.js

Este archivo contiene el proceso del print en donde busca el valor y el tipo para poder imprimirlo en consola.



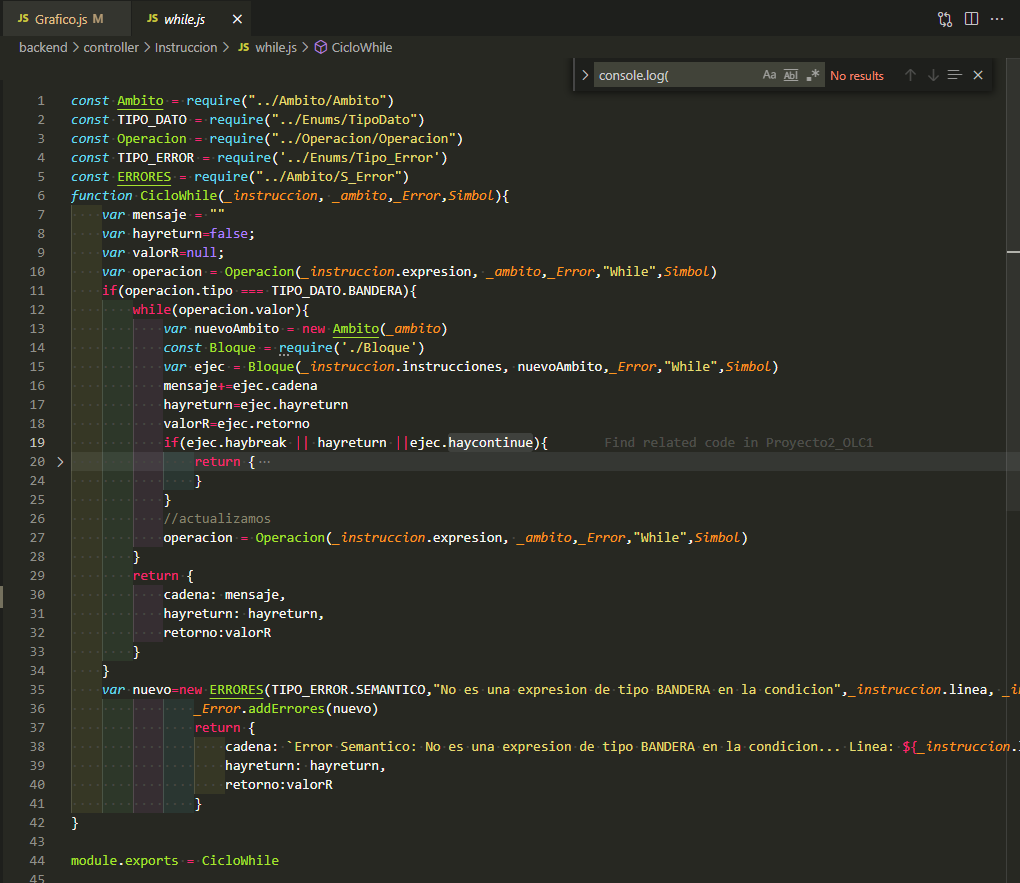
## Switch.js

Este archivo contiene el proceso de la sentencia del switch se evaluaran las condiciones y si llega a entrar alguno con un break adentro ya no ejecutara el default.



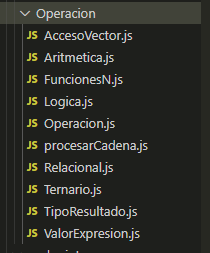
## While.js

Este archivo contiene el proceso del ciclo while, se hace desde un while y se va actualizando la condición, y hasta que se deja de cumplir la condición sale y retorna los valores necesarios.



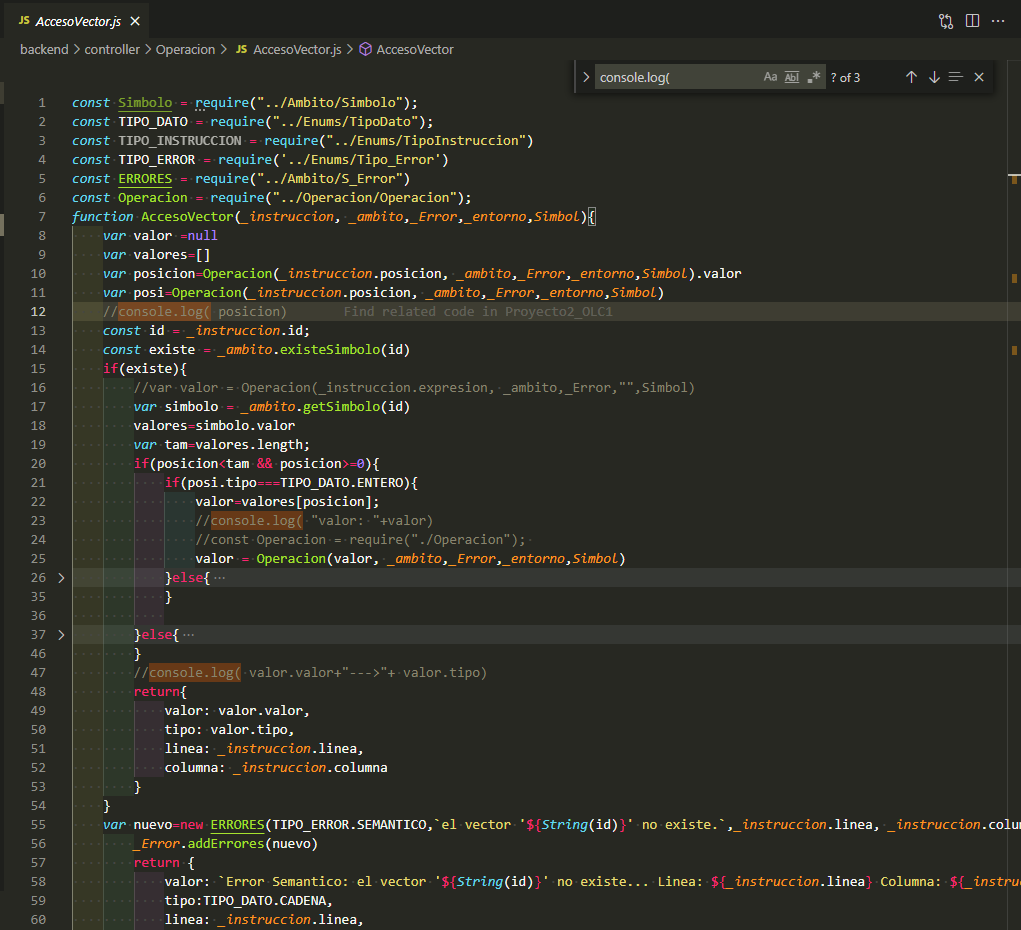
# operación

Esta carpeta contiene todas las operaciones y el acceso a todos los valores y tipos.



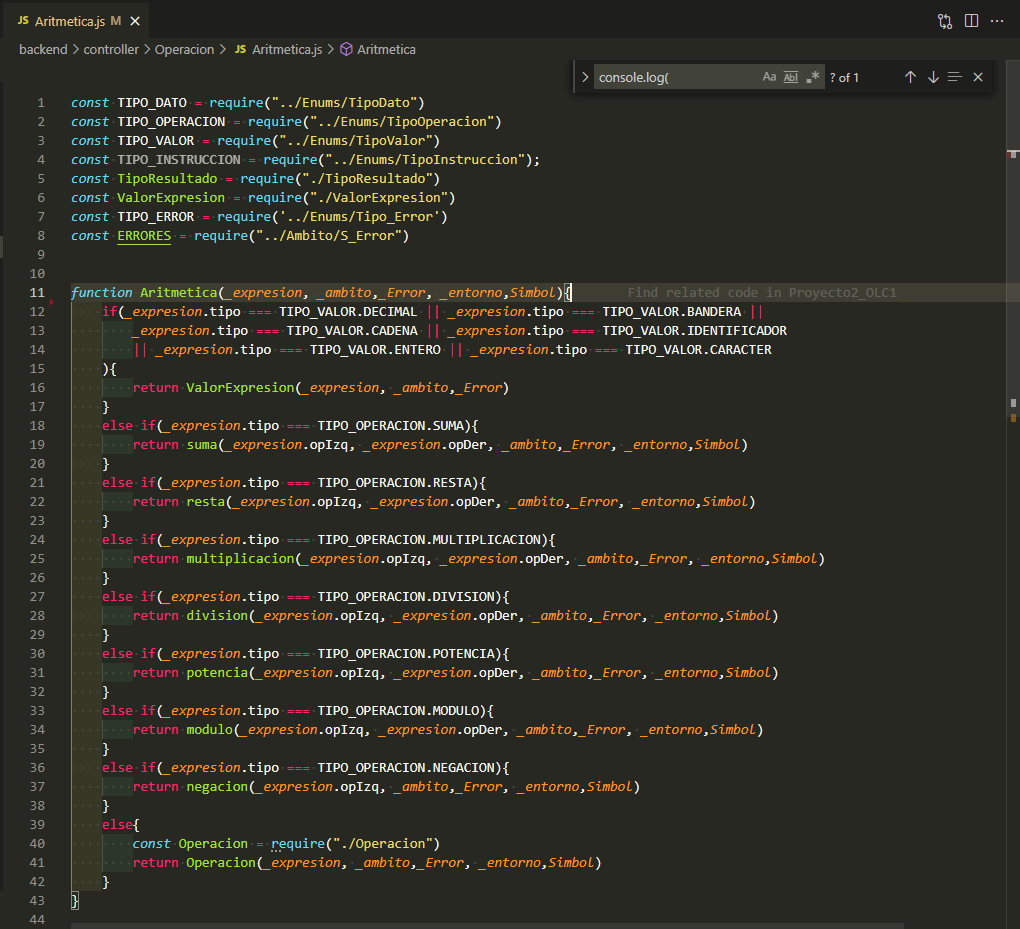
## AccesoVector.js

Este archivo contiene el proceso del acceso de listas y vectores en una posición.



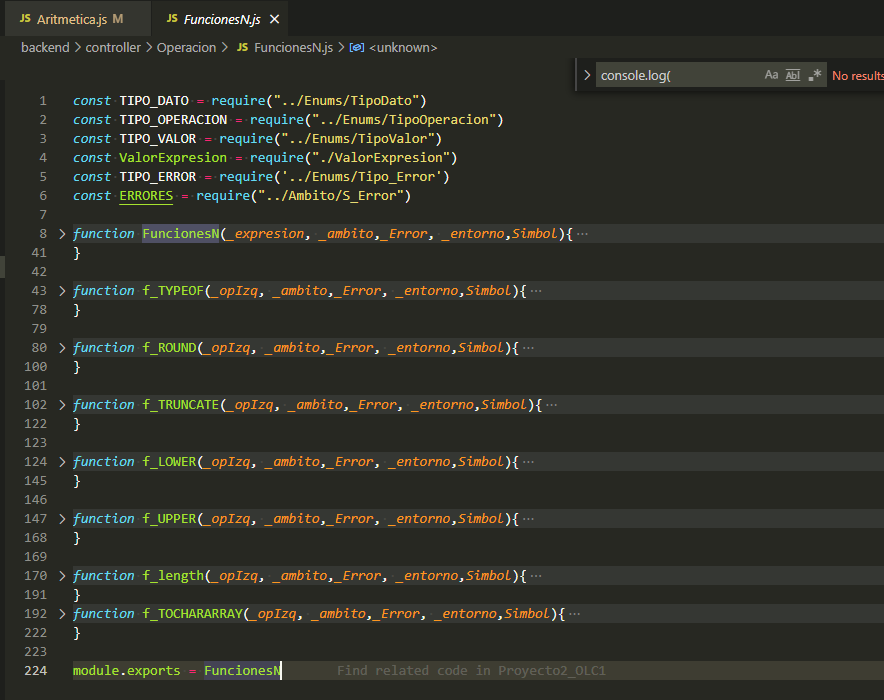
## aritmetica.js

Este archivo contiene el proceso de las operaciones aritméticas, como lo son suma, resta, multiplicación, etc.



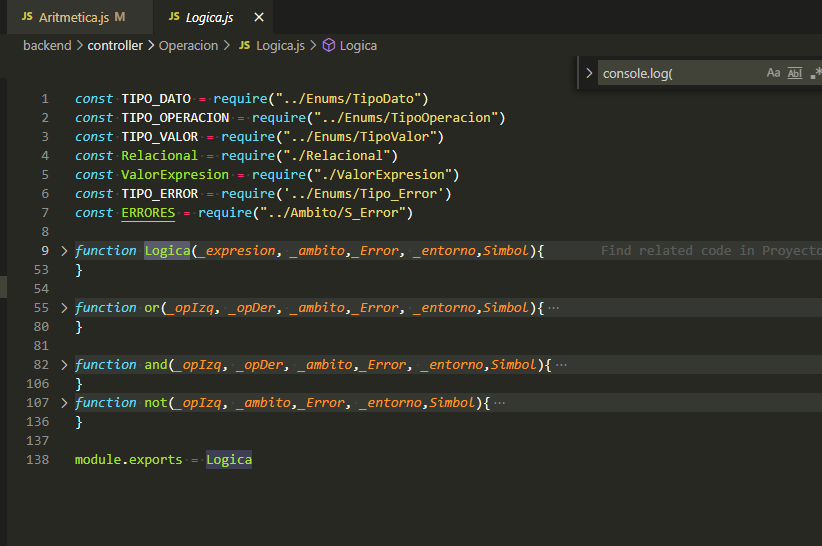
## funcionesn.js

Este archivo contiene el proceso de las funciones nativas y especiales.



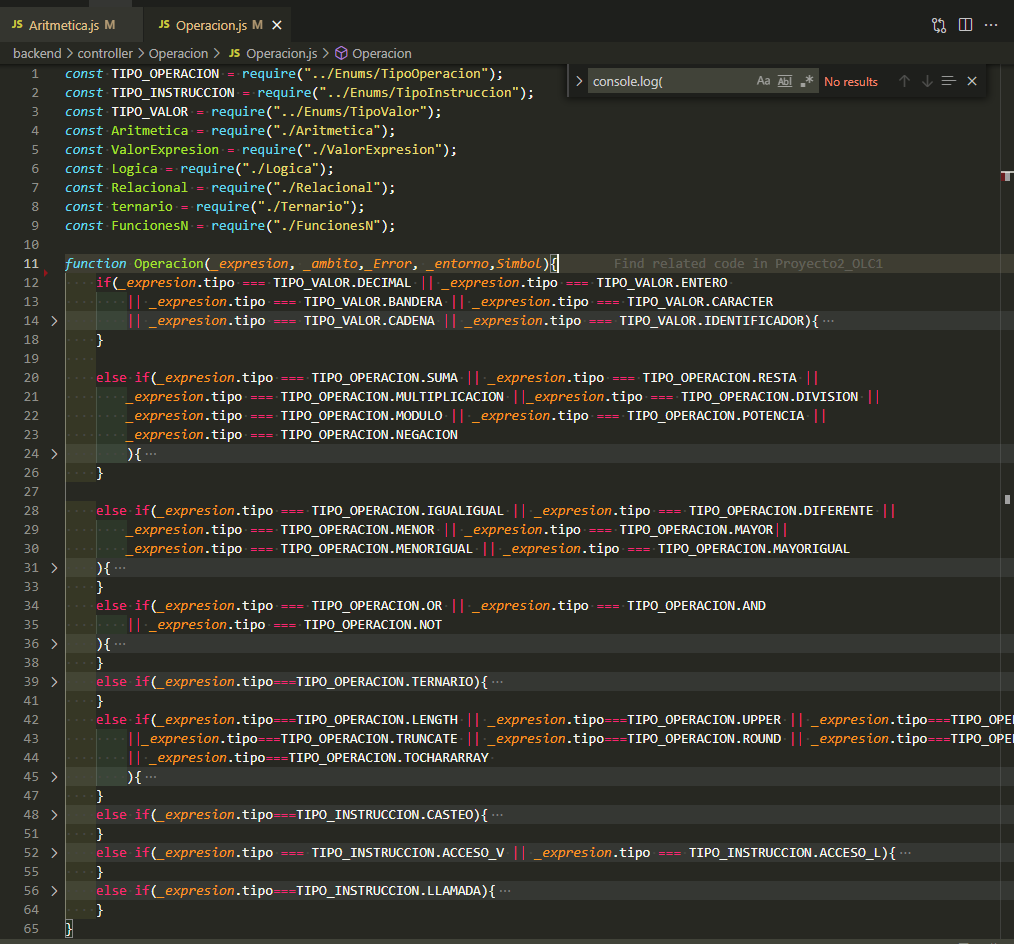
## Logica.js

Este archivo contiene el proceso de las operaciones lógicas.



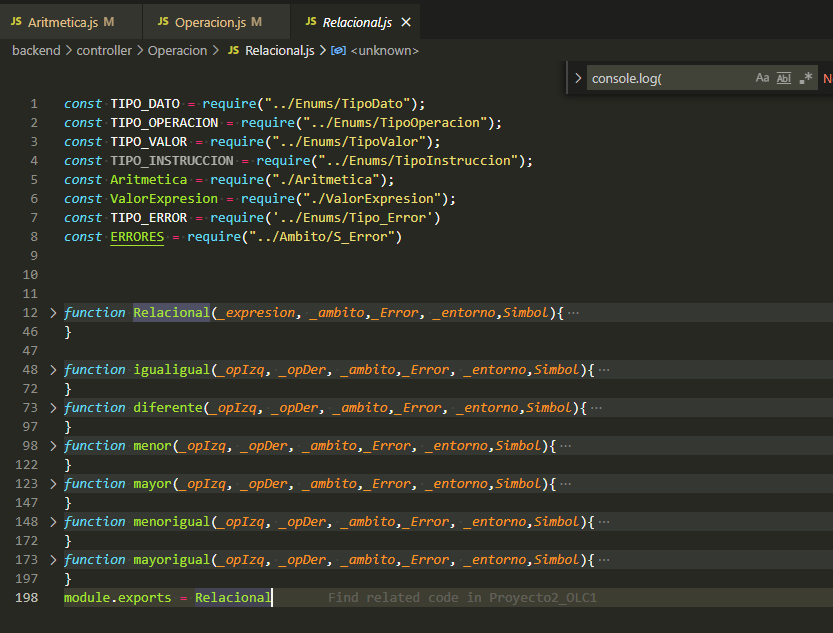
## Operación.js

Este archivo contiene es el que manda a llamar a cada archivo según el tipo de instrucción que sea.



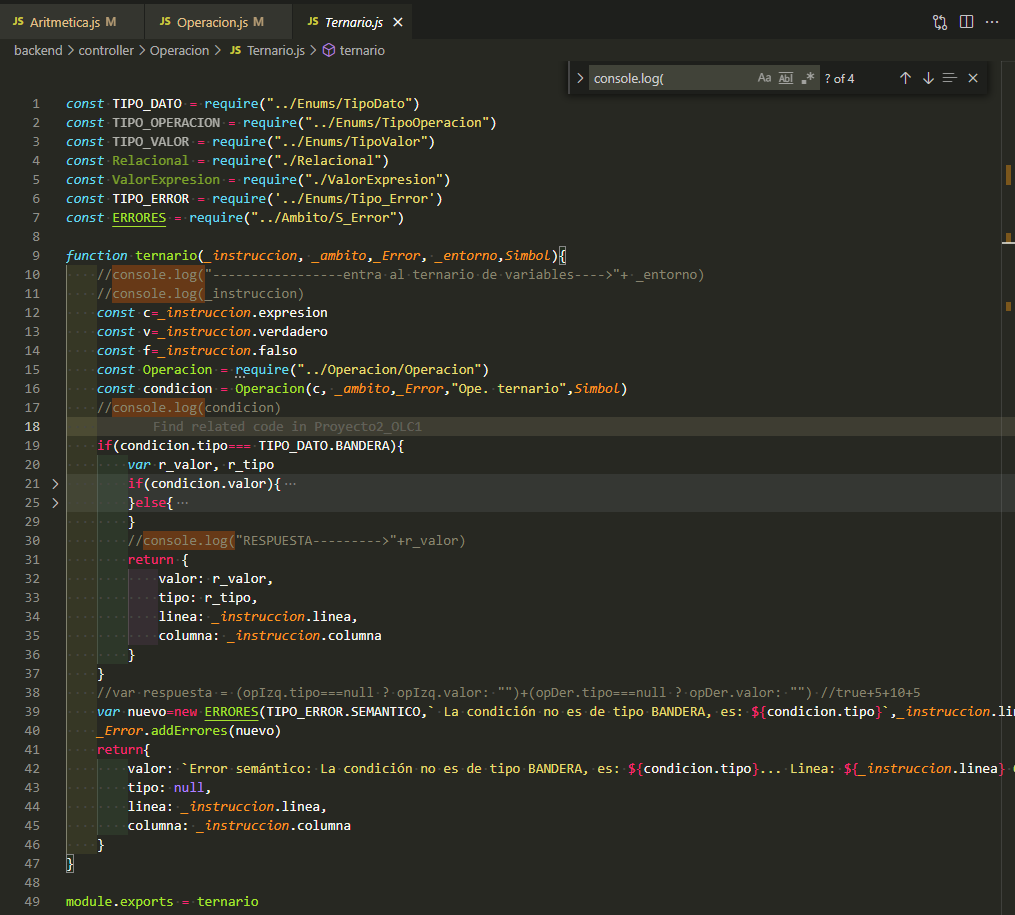
## Relacional.js

Este archivo contiene el proceso de las operaciones relaciones.



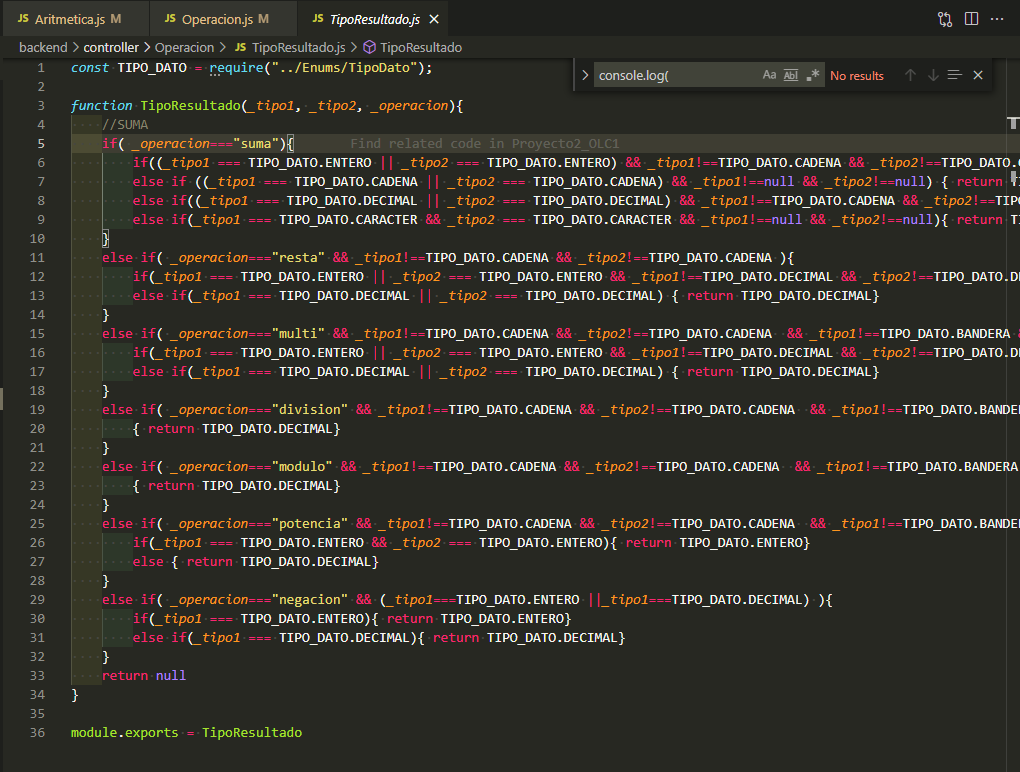
## Ternario.js

Este archivo contiene el proceso del operador ternario, que condición en donde si es verdadera ejecutará la primera expresión, de lo contrario ejecutará la segunda expresión.



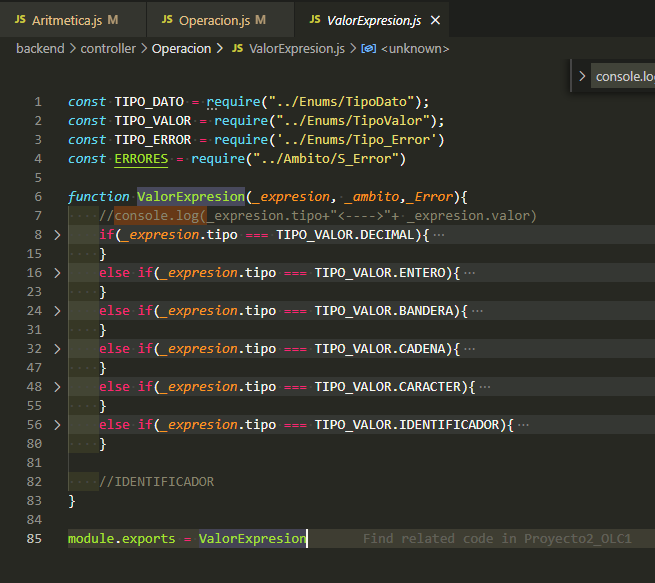
## Tiporesultado.js

Este archivo contiene las condiciones para saber el tipo de resultado para las operaciones aritméticas.



## Valorexpresion.js

Este archivo devuelve el tipo, valor, id, línea y columna de una variable.



# frontend

En la carpeta frontedN >> src >> components contiene el html, css y ts a utilizar como cliente.

