Manual Técnico

El desarrollo del proyecto **Mini-C** se apoyó en diversas tecnologías modernas que permitieron construir un entorno completo de edición, análisis y visualización de código para un lenguaje de programación propio. A continuación, se detallan las principales herramientas empleadas.

Tecnologías Usadas

Angular

El núcleo de la aplicación fue construido utilizando Angular, un framework para el desarrollo de aplicaciones web de una sola página (SPA). Se eligió Angular por su robustez, modularidad y facilidad de integración con componentes personalizados. La aplicación no depende de un backend externo, lo que simplifica la arquitectura y permite una ejecución completamente en el cliente.

Lenguaje de Programación

La implementación del proyecto incluyó los siguientes lenguajes:

TypeScript: Lenguaje principal para la lógica de la aplicación en Angular, proporcionando tipado estático y mejoras en el desarrollo.

JavaScript: Utilizado principalmente en la integración con bibliotecas como Peggy y Viz.js.

HTML: Definió la estructura visual de los componentes y la interfaz de usuario.

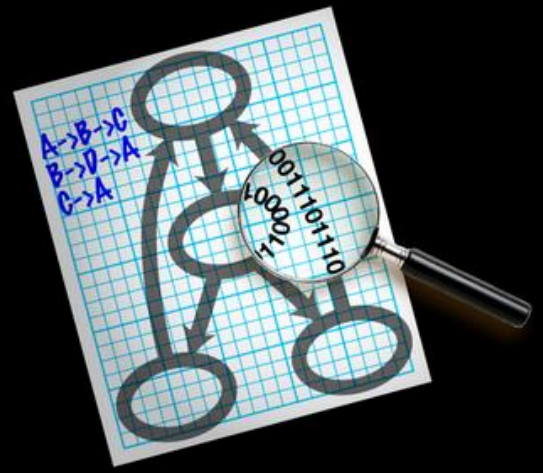
CSS: Se empleó para el diseño, estilos y tema visual del editor, incluyendo un tema similar a IntelliJ.

Peggy

Para la construcción del analizador sintáctico, se utilizó Peggy, un generador de parsers basado en expresiones gramaticales (PEGs). Esta herramienta permite definir la gramática del lenguaje Mini-C de manera declarativa, generando automáticamente el parser en JavaScript. Se integró con el sistema para permitir la detección y reporte de errores sintácticos y léxicos en tiempo real.

Graphviz (Viz.js)

La representación gráfica de estructuras como árboles de sintaxis abstracta (AST) se logró mediante la biblioteca Viz.js, una implementación en JavaScript de Graphviz. Esto permitió generar visualizaciones automáticas y dinámicas directamente en el navegador, sin necesidad de herramientas externas ni instalaciones adicionales.



Dependencias para (Viz.js)

Agregar
tsconfig.ts

```
"types": ["@viz-js/viz"],
"target": "ES2022",
"module": "ES2022",
```

Crear
typings.d.ts

```
declare module '@viz-js/viz' {
    export function instance(): Promise<any>;
    export function renderSVGElement(dot: string): SVGSVGElement;
}
```

Visual Studio Code

Durante el desarrollo, se utilizó Visual Studio Code (VSCode) como entorno de desarrollo principal. Gracias a su amplia gama de extensiones y herramientas integradas, facilitó la edición del código, el control de versiones y la integración con el flujo de trabajo basado en Angular y Peggy.

Gramática para interprete mini-c

```
FLOAT = digits:[0-9]+ "." decimals:[0-9]+ { return parseFloat(digits.join("") + "." + decimals.join("")); }
;

INT = digits:[0-9]+ { return digits.join(""); }
;

STRING = "\"" chars:([^"]*) "\"" { return chars.join(""); }
;

CARACTER = "'" chars:([^']*) "'" { return chars.join(""); }
;

IDENTIFICADOR = [a-zA-Z][a-zA-Z0-9_]* { return text(); }
;

EOF = !. { return "EOF"; }
;

// Espacios
_  = [ \t\r\n]*
```

```
tipos = "int" { return new Tipo.default(Tipo.TipoDato.INT) }
      / "float" { return new Tipo.default(Tipo.TipoDato.FLOAT) }
      / "string" { return new Tipo.default(Tipo.TipoDato.STRING) }
      / "char" { return new Tipo.default(Tipo.TipoDato.CHAR) }
      / "bool" { return new Tipo.default(Tipo.TipoDato.BOOL) }
      / "void" { return new Tipo.default(Tipo.TipoDato.VOID) }
      / "struct" { return new Tipo.default(Tipo.TipoDato.STRUCT) }
;
```

```
exprStruct = id:IDENTIFICADOR _ "." _ atr:IDENTIFICADOR { return new acceStruct(id, atr, location().start.line, location().start.column) }
          / llamadaExpresion
;

llamadaExpresion = id:IDENTIFICADOR _ "(" _ args:argumentos? _ ")" { return new Llamada.default(id, args || [], location().start.line, location().start.column) }
                 / or
;

or = izq:and _ "||" _ der:or { return new Booleanas.default(Booleanas.Operadores.OR, location().start.line, location().start.column, izq, der); }
   / and
;

and = izq:igualdad _ "&&" _ der:and { return new Booleanas.default(Booleanas.Operadores.AND, location().start.line, location().start.column, izq, der); }
    / igualdad
;

igualdad = izq:relacional _ operador:("==" / "!=") _ der:igualdad { return new Relacionales.default( operador === "==" ? Relacionales.Operadores.EQUALS : Relacionales.Operadores.NOTEQUALS, location().start.line, location().start.column, izq, der); }
         / relacional
;

relacional = izq:suma _ operador:("<=" / ">=" / "<" / ">") _ der:relacional {
              let tipo = {
                "<" : Relacionales.Operadores.MENORQUE,
                ">" : Relacionales.Operadores.MAYORQUE,
                "<=" : Relacionales.Operadores.MENORIGUAL,
                ">=" : Relacionales.Operadores.MAYORIGUAL
              }[operador];
              return new Relacionales.default(tipo, location().start.line, location().start.column, izq, der);
            }
            / suma
;

suma = izq:producto _ operador:("+" / "-") _ der:suma { return new Aritmeticas.default( operador === "+" ? Aritmeticas.Operadores.SUMA : Aritmeticas.Operadores.RESTA, location().start.line, location().start.column, izq, der); }
     / producto
;

producto = izq:negacion _ operador:("*" / "/") _ der:producto { return new Aritmeticas.default( operador === "*" ? Aritmeticas.Operadores.MULTIPLICACION : Aritmeticas.Operadores.DIVISION, location().start.line, location().start.column, izq, der); }
         / negacion
;

negacion = "-" _ uni:negacion { return new Aritmeticas.default(Aritmeticas.Operadores.NEGACION, location().start.line, location().start.column, uni); }
         / "!" _ uni:negacion { return new Booleanas.default(Booleanas.Operadores.NOT, location().start.line, location().start.column, uni); }
         / potencia
;

potencia = izq:termino _ "^" _ der:potencia { return new Aritmeticas.default(Aritmeticas.Operadores.POTENCIA, location().start.line, location().start.column, izq, der); }
         / termino
;

termino = cadena:STRING { return new Nativo.default(new Tipo.default(Tipo.TipoDato.STRING), cadena, location().start.line, location().start.column); }
        / char:CARACTER { return new Nativo.default(new Tipo.default(Tipo.TipoDato.CHAR), char, location().start.line, location().start.column); }
        / decimal:FLOAT { return new Nativo.default(new Tipo.default(Tipo.TipoDato.FLOAT), decimal, location().start.line, location().start.column); }
        / entero:INT { return new Nativo.default(new Tipo.default(Tipo.TipoDato.INT), entero, location().start.line, location().start.column); }
        / "true" { return new Nativo.default(new Tipo.default(Tipo.TipoDato.BOOL), true, location().start.line, location().start.column); }
        / "false" { return new Nativo.default(new Tipo.default(Tipo.TipoDato.BOOL), false, location().start.line, location().start.column); }
        / variable:IDENTIFICADOR { return new Variable.default(variable, location().start.line, location().start.column); }
        / "(" expr:expresion ")" { return expr; }
;
```

```
print = "print" "("expr:expresion")" { return new Print.default(expr, location().start.line, location().start.column); }
;

declaracion = tip:tipos _ id:IDENTIFICADOR _ "=" _ expr:expresion { return new Declaracion.default(tip, location().start.line, location().start.column, id, expr) }
;

asignacion = id:IDENTIFICADOR _ "=" _ expr:expresion _ ";" { return new Asignacion.default(id, location().start.line, location().start.column, expr) }
           / id:IDENTIFICADOR _ "++" { return new Incremento.default(id, location().start.line, location().start.column) }
           / id:IDENTIFICADOR _ "--" { return new Decremento.default(id, location().start.line, location().start.column) }
;

return = "return" _ expr:expresion { return new Return.default(expr, location().start.line, location().start.column) }
;

break = "break" _ { return new Break.default(location().start.line, location().start.column) }
;

sif = "if" _ "(" _ expr:expresion _ ")" _ "{" _ inst:instrucciones _ "}" _ "else" _ "{" _ insts:instrucciones _ "}" { return new If.default(expr, inst, null, insts, location().start.line, location().start.column) }
    / "if" _ "(" _ expr:expresion _ ")" _ "{" _ inst:instrucciones _ "}" _ "else" _ si:sif { return new If.default(expr, inst, si, null, location().start.line, location().start.column) }
    / "if" _ "(" _ expr:expresion _ ")" _ "{" _ inst:instrucciones _ "}" { return new If.default(expr, inst, null, null, location().start.line, location().start.column) }
;

cfor = "for" _ "(" _ dec:declaracion _ ";" _ exp:expresion _ ";" _ asg:asignacion _ ")" _ "{" _ inst:instrucciones _ "}" { return new For.default(dec, exp, asg, inst, location().start.line, location().start.column) }
;

structDef = "struct" _ id:IDENTIFICADOR _ "{" _ atr:atributos+ _ "}" { return new Structs(id, atr, location().start.line, location().start.column) }
;

atributos = tip:tipos _ id:IDENTIFICADOR _ ";" _ { return { tipo: tip, id: id }; }
;

structInst = "struct" _ tip:IDENTIFICADOR _ id:IDENTIFICADOR _ init:structInit? _ ";" { return new instaStruct(tip, id, init, location().start.line, location().start.column) }
;

structInit = _ "=" _ "{" _ valor:lista _ "}" { return valor; }
;

lista = exp:expresion resto:(_ "," _ expresion)* { return [exp].concat((resto || []).map(e => e[3])); }
;

accesoStruct = id:IDENTIFICADOR _ "." _ atributo:IDENTIFICADOR _ ";" { return new acceStruct(id, atributo, location().start.line, location().start.column) }
;

asigStruct = id:IDENTIFICADOR _ "." _ atributo:IDENTIFICADOR _ "=" _ expr:expresion _ ";" { return new AsigStruct(id, atributo, expr, location().start.line, location().start.column) }
;

ambito = "{" _ inst:instrucciones _ "}" { return new Ambito.default(inst, location().start.line, location().start.column) }
;

funcion = tip:tipos _ id:IDENTIFICADOR _ "(" _ params:parametros? _ ")" _ "{" _ inst:instrucciones _ "}" { return new Function.default(tip, id, params || [], inst, location().start.line, location().start.column) }
;

llamada = id:IDENTIFICADOR _ "(" _ args:argumentos? _ ")" _ ";" { return new Llamada.default(id, args || [], location().start.line, location().start.column) }
;

parametros = first:parametro rest:(_ "," _ parametro)* { return [first, ...rest.map(r => r[3])] }
;

parametro = tip:tipos _ id:IDENTIFICADOR { return { tipo: tip, id: id } }
;

argumentos = first:expresion rest:(_ "," _ expresion)* { return [first, ...rest.map(r => r[3])] }
;
```

```
inicio = importar? _ "void" _ "main" _ "(" _ ")" _ "{" _ instrs:instrucciones _ "}" { return instrs; }
;

importar = "#" "import" _ mod:IDENTIFICADOR "." arch:IDENTIFICADOR ";" { console.log("Importando: ", mod, arch) }
;

instrucciones = inst:instruccion insts:(_ instruccion)* { return [inst, ...insts.map(t => t[1])]; }
;

instruccion = imprimir:print ";" { return imprimir; }
            / declarar:declaracion ";" { return declarar; }
            / asignar:asignacion { return asignar; }
            / retu:return ";" { return retu; }
            / bre:break ";" { return bre; }
            / si:sif { return si; }
            / cf:cfor { return cf; }
            / sd:structDef { return sd; }
            / si:structInst { return si; }
            / as:accesoStruct { return as; }
            / ss:asigStruct {return ss; }
            / amb:ambito { return amb; }
            / fun:funcion { return fun; }
            / ll:llamada { return ll; }
;
```

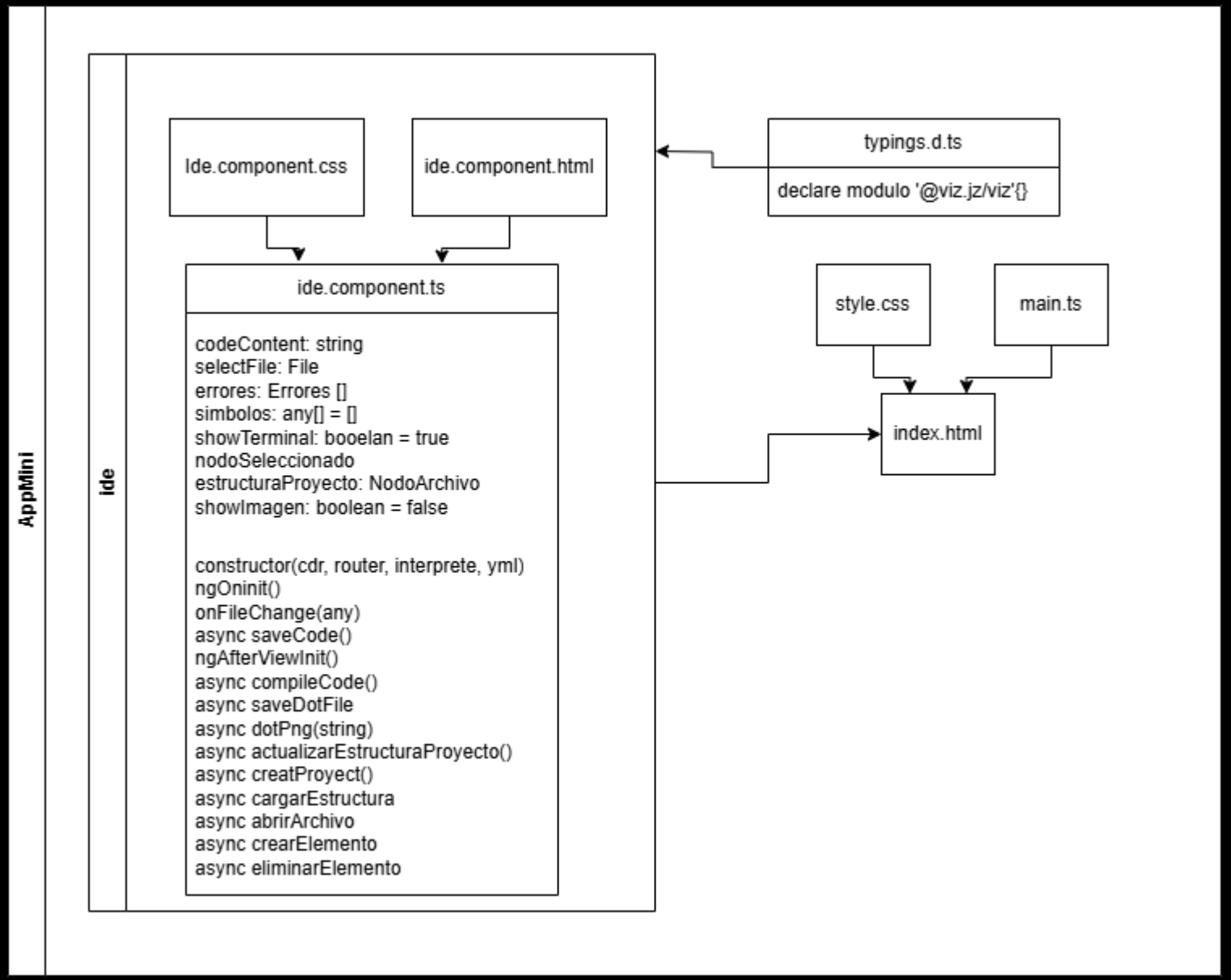
Gramática para yml

```
  /* LEXICO */
DATO = "\"" chars:([^"]*) "\"" { return chars.join(""); }

IDENTIFICADOR = [a-zA-Z][a-zA-Z0-9_.]* { return text(); }

EOF = !. { return "EOF"; }

_ = ( [ \t\r\n] / COMENTARIO )*

INDENT = [ \t]*

COMENTARIO = "#" [^\r\n]* ("\r"? "\n" / !.)
```

```
llaves = id:IDENTIFICADOR _ ":" _ valor:DATO { return { tipo: "llave", id, valor } }
;

listas = id:IDENTIFICADOR _ ":" _ p:parametros? { return { tipo: "lista", id, parametros: p ?? [] } }
;

parametros = first:parametro rest:(_ parametro)* { return [first, ...rest.map(r => r[1])] }
;

parametro = INDENT "-" _ id:IDENTIFICADOR _ ":" _ valor:DATO { return { id, valor } }
;
```

```
  /* SINTACTICO */
inicio = _ instrucciones:instrucciones EOF { return instrucciones }

instrucciones = _ lista:(instruccion _)* { return lista.map(e => e[0]) }
;

instruccion = llaves
            / listas
;
```

Diagramas de Clases editor



Diagramas de Clases Interprete