1. Arquitectura del compilador:
   1. Lexer y Parser:

Para la implementación del lexer y el parser se utilizo como herramienta ANTLR4 donde en COOLgrammar.g4 se definió el lexer dando una definición de cada token a partir del uso de expresiones de regulares. También en dicho fichero se definió la estructura de la gramática, en cuanto a este aspecto cabe destacar que dicho ¿framework? maneja por nosotros la recursión izquierda inmediata eliminando la ambiguedada que introduce esta ¿? y establece el orden de las operaciones de acorde al orden en que se definen las producciones de cada no terminal lo cual simplifico en gran medida esta parte del proyecto. EL árbol de derivaciones generado por el parser esta definido en COOLgrammarParser.cs el cual se usó como el ast de Cool.

* 1. Análisis semánticos:

En este se visita el ast en 3 pasadas:

* La primera para capturar la definición de los tipos
* La segunda capturar la definición de los atributos y métodos. En esta pasada las clases se analizan en orden topológico para cuando se analiza un método de una clase A heredado de otra clase B tener los métodos de dicha clase B previamente analizados.
* La tercera es para analizar las expresiones definidas en el cuerpo de los métodos y las inicializaciones de los atributos.

En estas 3 pasadas se usaron estructuras que implementan IContext las cuales se usan para:

* Atrapar las definiciones de los Tipos, atributos, metodos, parámetros, variables introducidas por las expresones let .
* Preguntar si en un determinado contexto alguna de estas definiciones existe.
* Acceder a estas definiciones y preguntar por sus propiedades.
  1. Generación de código:
     1. Generando de código de Cool a Cil: