Universidad de San Carlos de Guatemala
Escuela de Ciencias y Sistemas
Facultad de Ingeniería Lenguajes formales y de programación
Vacaciones Primer Semestre 2025
Catedrático: Inga. Asunción Mariana Sic Sor
Tutor académico: Elian Saúl Estrada Urbina

**PROYECTO 2**

"Transpilador"

Axel David González Molina
202402074

Fecha de entrega:
29/6/2025


# Introducción

Este proyecto se centra en el desarrollo de un transpilador completo en TypeScript, diseñado para traducir código fuente escrito en C# a su equivalente en TypeScript, abarcando las estructuras de programación más utilizadas en el desarrollo de aplicaciones.

La necesidad de un sistema que permita la traducción automática entre estos lenguajes surge de la creciente demanda de interoperabilidad en el ecosistema de desarrollo web y de aplicaciones, donde TypeScript ha ganado popularidad significativa por su capacidad de proporcionar tipado estático sobre JavaScript. El transpilador no solo se encargará de realizar la traducción sintáctica del código, sino que también implementará un intérprete capaz de ejecutar las instrucciones de salida por consola, proporcionando una validación inmediata de la funcionalidad del código traducido.

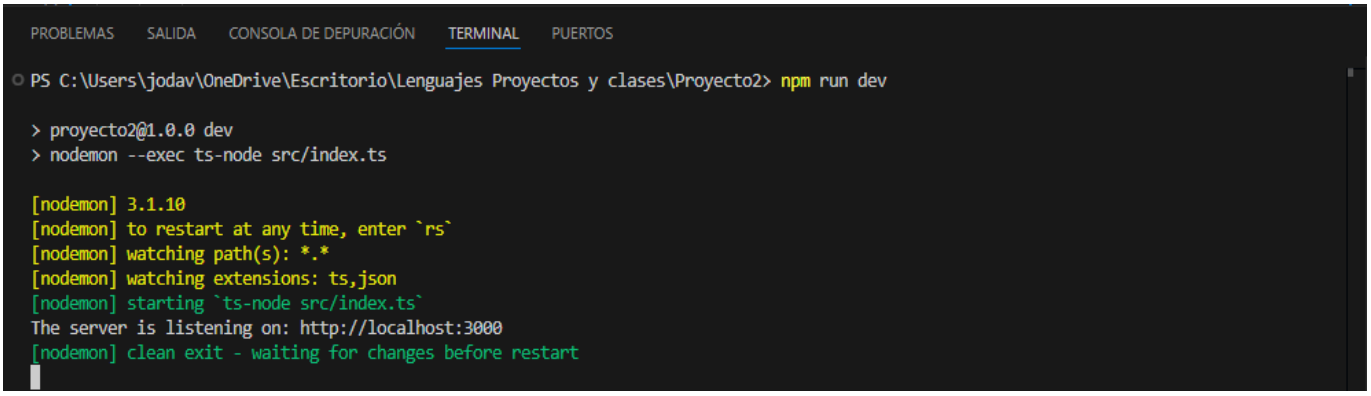
A través de la implementación de un analizador léxico basado en autómatas finitos deterministas (AFD) y un analizador sintáctico fundamentado en gramáticas libres de contexto mediante autómatas de pila, el sistema será capaz de reconocer y procesar patrones complejos del lenguaje C#.

El objetivo principal de este proyecto es implementar un análisis léxico y sintáctico robusto que no solo cumpla con los requisitos de traducción, sino que también proporcione una comprensión profunda de los conceptos fundamentales en la teoría de compiladores y lenguajes formales. La interfaz gráfica desarrollada será intuitiva y funcional, permitiendo a los usuarios cargar archivos C# (.cs), visualizar la traducción resultante en TypeScript (.ts), monitorear la ejecución a través de una consola integrada y acceder a reportes detallados de tokens, errores y símbolos. Este documento establece los fundamentos teóricos y metodológicos necesarios para la implementación exitosa de esta herramienta de transpilación.

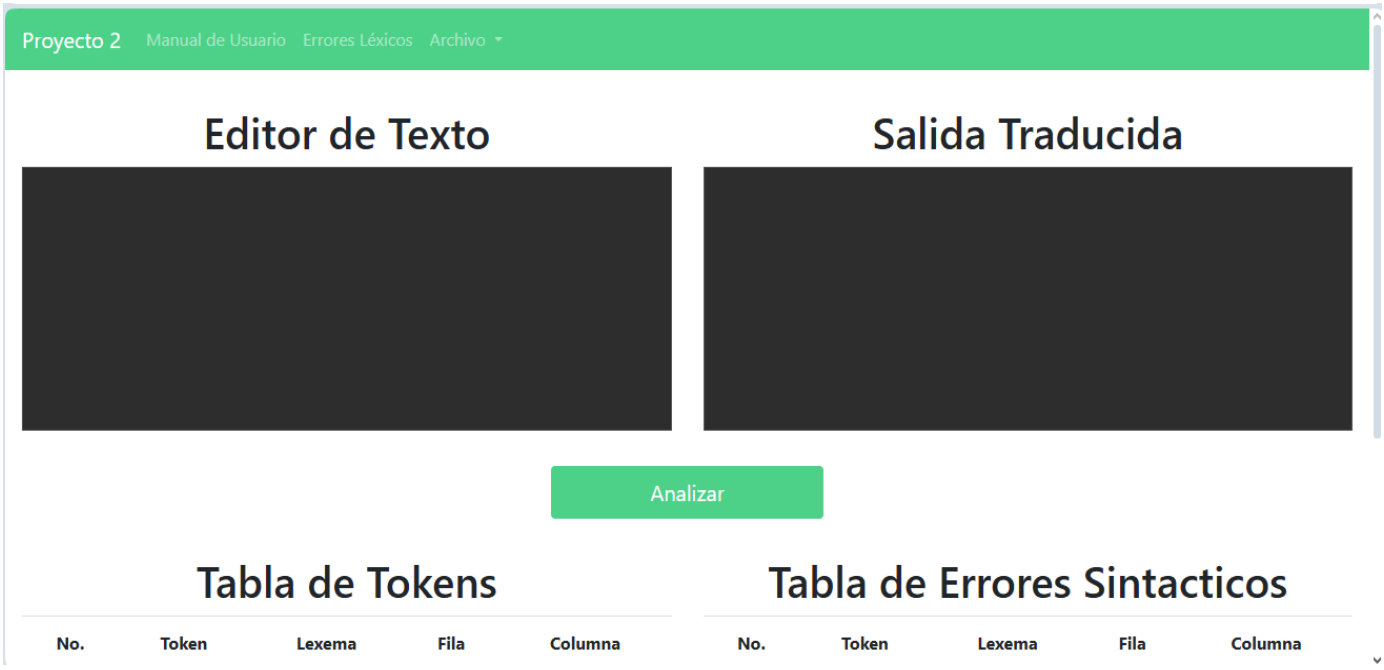
## Manual de usuario

A continuación, se detallará brevemente cómo funciona la aplicación y cómo hacer uso de todas sus funciones:

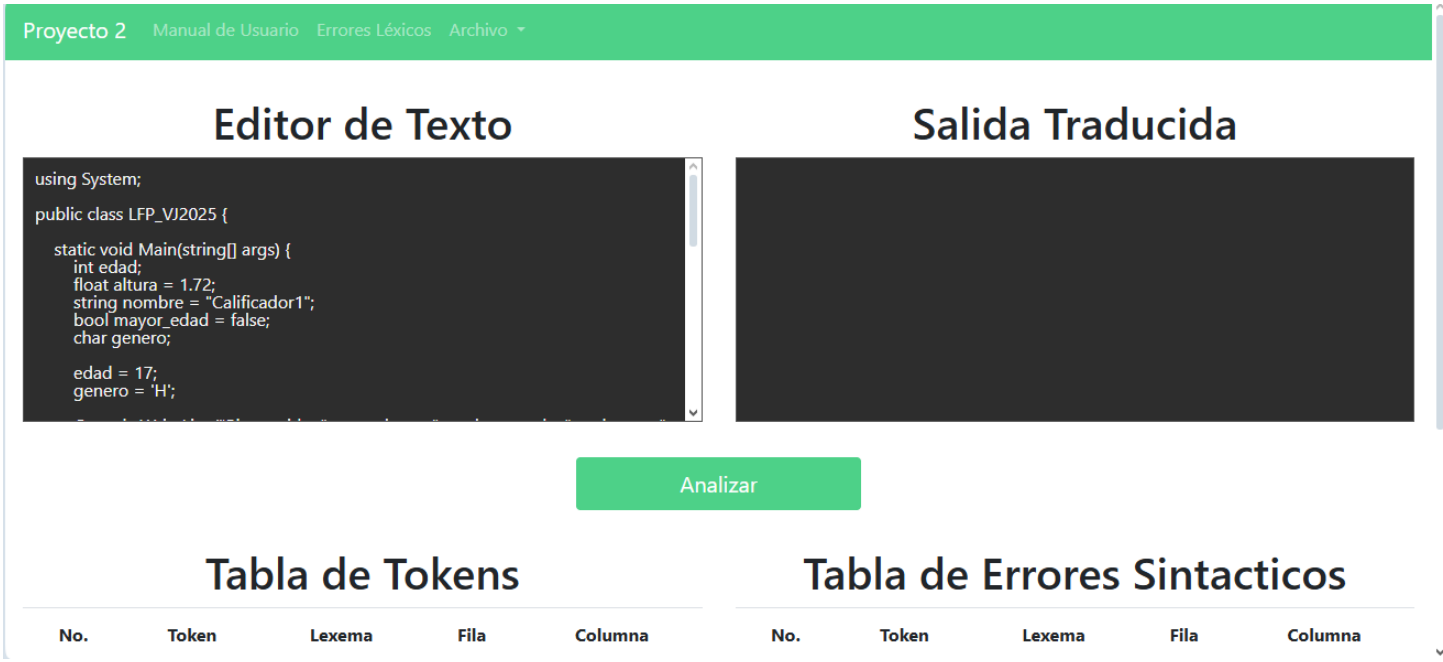
- La aplicación está diseñada para realizar un análisis léxico de archivos de texto que contienen información sobre los cursos y prerrequisitos en un pensum académico. Utiliza un analizador léxico que identifica y clasifica los tokens presentes en el texto, permitiendo a los usuarios visualizar la estructura del pensum y los requisitos de cada curso.
- Al iniciar la aplicación de Visual Studio Code, se deberá ingresar el siguiente comando en la terminal para poder iniciar la aplicación: npm run dev.




- En la interfaz principal, encontrar una barra de navegación desde la cual se podrán cargar y guardar archivos, limpiar el editor, y cambiar de página, además de visualizar el transpilador y tablas con los tokens.



- Se necesita ingresar información directamente en el área de texto o cargarlo para comenzar el análisis




- Presionar el botón "Analizar" para que se llene la tabla con los respectivos tokens y se coloreara de colores el texto para indicar el texto escaneado, **Si el texto contiene "errores" (palabras que no pertenezcan a nuestro lenguaje) El texto en código no se transpilara.**

- Después de realizar el análisis, la aplicación transpilara el código ingresado según el la gramática BFN y se podrá visualizar en un área de texto en el lenguaje de TypeScript, y se podrá visualizar cada token del texto en la tabla.


- Cuando hayas terminado de usar la aplicación, puedes cerrarla simplemente cerrando la pestaña del navegador o deteniendo el servidor en la terminal con Ctrl + C.


# Conclusiones

El desarrollo de un transpilador de C# a TypeScript aplica principios fundamentales de teoría de compiladores y lenguajes formales. La implementación de analizadores léxico y sintáctico mediante autómatas finitos y gramáticas libres de contexto permite una traducción efectiva y la interpretación de instrucciones, validando la funcionalidad del sistema.

Además, la interfaz gráfica de usuario, que incluye capacidades de edición, análisis y generación de reportes, es una herramienta educativa valiosa que ayuda a comprender conceptos complejos como tablas de símbolos y análisis semántico.

El proyecto equilibra robustez técnica y usabilidad, proporcionando reportes detallados que facilitan el debugging y la comprensión del transpilador. Esta aproximación integral sienta las bases para futuras extensiones y el estudio avanzado de la teoría de compilador


# Recomendaciones

Desarrollar el sistema siguiendo principios de arquitectura modular, separando claramente el analizador léxico, sintáctico, generador de código y intérprete para facilitar el mantenimiento y testing.

Implementar un sistema de manejo de errores que proporcione información precisa sobre la ubicación y naturaleza de los errores sintácticos y léxicos, incluyendo mensajes descriptivos para facilitar la corrección.

Desarrollar un conjunto comprehensivo de casos de prueba que cubran todos los escenarios posibles, incluyendo casos límite y situaciones de error.

Realizar un estudio exhaustivo de la teoría de compiladores antes de iniciar la implementación, focalizándose en autómatas finitos, gramáticas libres de contexto y análisis sintáctico.