Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes Formales y Autómatas, Sección 02
Docente: Mgtr. Vivian Damaris Socorro Campos Gonzales

Manual Técnico

"Biblioteca Digital"

Versión 1.0

Autores:
Lizbeth Andrea Herrera Ortega – 1246024
Marcela Nicole Letran Lee – 1102124

Guatemala, 05 de septiembre de 2025

Requisitos para la Compilación y Ejecución

Requisitos de Hardware

Este sistema está diseñado para ejecutarse en entornos de escritorio o portátiles con capacidades básicas de procesamiento, por lo que no requiere procesamiento gráfico ni cómputo intensivo. No requiere GPU ni acelerador de Hardware

Componente	Requisitos mínimos	Requisitos recomendados para mejorar rendimiento
CPU	Procesador de 2 núcleos (x86_64)	Intel i5 o superior
RAM	2 GB	4 GB o más
Almacenamiento	100 MB libres	SSD con al menos 500 MB disponibles
Periféricos	Teclado y pantalla estándar	Mouse para navegación

Requisitos de Software

El sistema escrito en Python y utiliza bibliotecas estándar del mismo, por lo que no requiere frameworks externos ni dependencia de terceros. Compatible con:

- Windows 10 o superior
- macOS 11 (Big Sur) o superior
- Distribuciones Linux (Ubuntu 20.04+, Debian)

Es necesario tener un intérprete de Python, con una versión mínima de 3.10.

No se requiere alguna instalación adicional. Todos los módulos son parte de la biblioteca estándar de Python:

- datetime – Para validación y comparación de fechas
- tkinter – Aunque se importa, no se utiliza en el código actual
- collections.Counter – Para estadísticas de préstamos

Editor recomendado:

Visual Studio Code, PyCharm, o cualquier editor con soporte para Python 3.10+

Implementación del analizador como la fase inicial de un compilador

El sistema recrea la primera etapa de un compilador, es decir, la conversión del texto fuente a una secuencia de tokens validada.

1. Lectura y procesamiento de entrada
   El programa lee línea a línea un archivo (.lfa) donde cada registro contiene seis campos separados por comas.
- La función leer_archivo abre el archivo en modo UTF-8, ignora la cabecera y mantiene un contador de líneas para reportes de error.
- Cada línea se pasa a analizador_lexico, que acumula caracteres hasta encontrar una coma y así delimitar lexemas (palabras clave, identificadores, fechas, etc.), insertando también tokens de tipo "coma".
2. Extracción de Lexemas y Asignación de Tokens
   Dentro de analizador_lexico, al detectar separadores:
- Se recorta el lexema (.strip()).
- Se llama a clasificar_lexema(lexema, posicion), que, según el índice de campo:
    - Valida formato numérico para id_usuario.
    - Comprueba caracteres alfabéticos para nombre_usuario.
    - Reconoce prefijos “LIB” y dígitos para id_libro.
    - Acepta cualquier cadena alfanumérica para titulo_libro.
    - Intenta parsear fechas con %Y-%m-%d para fecha_prestamo y fecha_devolucion.
- Los lexemas que no casan con ninguna regla reciben el token "desconocido".
3. Detección de Errores Léxicos y Semánticos por Campo
   La función validar_campo profundiza la verificación de cada lexema:
- Rechaza caracteres fuera del alfabeto español en nombres de usuario.
- Impide comas en títulos de libros.
- Reporta formatos distintos a “YYYY-MM-DD” en fechas y errores semánticos
- Genera mensajes claros indicando línea, columna, carácter y contexto del fallo.
4. Ensamble y Validación de la Línea Completa
   validar_linea agrupa los tokens no “coma” y:
- Confirma que existan exactamente seis campos.
- Comprueba que no aparezca ningún token "desconocido".
- Verifica que el orden de tokens coincida con el arreglo orden_valido.
- Para cada campo, llama de nuevo a validar_campo.
- Evalúa la consistencia temporal: si la fecha de préstamo es posterior a la de devolución, arroja un error semántico.
5. Construcción de Tablas de Símbolos y Enlace Semántico

Antes de procesar préstamos, cargar_archivo arma dos tablas de símbolos:

- Usuarios: clave id_usuario → valor nombre_usuario.
- Libros: clave id_libro → valor titulo_libro. Durante leer_archivo, cada préstamo validado se enlaza contra estas tablas, descartando registros cuyos identificadores no existan o cuyos valores no coincidan.

6. Salida Estructurada para Análisis Posterior
   Las líneas libres de errores generan un diccionario con campos tipados y validados, que se acumula en la lista prestamos. Este resultado:

- Equivale a la “secuencia de tokens” de salida de un analizador léxico tradicional.
- Se alimenta a fases posteriores

El código implementa las funciones centrales de un analizador léxico: separación de lexemas, asignación de tokens, eliminación de elementos irrelevantes, detección de errores y almacenamiento en tablas de símbolos, tal como exige la arquitectura de un compilador.