Universidad Rafael Landívar
Facultad de ingeniería
Ingeniería en informática y sistemas
Lenguajes formales y autómatas, sección 02
Docente: Ing. Vivian Damaris Campos González

MANUAL TÉCNICO
Proyecto 01

Estudiantes:
González Pérez, Pablo Javier - 1211624
Miranda Abrego, Andrea Sofía - 1065824

Guatemala, 05 de septiembre del 2025

I. INTRODUCCIÓN

El proyecto tiene el objetivo de representar una simulación de préstamos de libros en una biblioteca digital manejando datos básicos del usuario y los libros que se prestan. Su funcionamiento se basa en procesar el archivo de texto con los datos, realizar un análisis léxico, almacenar los datos y generar un reporte en HTML partiendo en eso.

Por otro lado, el programa fue desarrollado en el lenguaje de Python debido a dos razones principales. Primero, Python ofrece mayor simplicidad a la hora de estructurar el código por lo que su desarrollo y lectura se vuelve más práctico. Además, se aplican conocimientos aprendidos en otro curso que se está llevando al mismo tiempo que este, por lo que apoyarse de eso complementa lo aprendido en ambos cursos. De la misma manera, se usan funciones para definir los métodos del programa y se omiten las bibliotecas de terceros, usando solo “datetime” como biblioteca estándar.

Para explicar el desarrollo general del proyecto se realizó un manual técnico que contiene tres secciones. La primera muestra el análisis y diseño del proyecto. Como parte del análisis se tienen las entradas, procesos, salidas y restricciones mientras que con el diseño se muestran las estructuras usadas, el manejo de errores y módulos. En la siguiente parte, se explica el análisis léxico aplicado, mostrando ejemplos de fragmentos de código y una breve explicación. Para finalizar, se muestran los diagramas de clase y de flujo.

II. ANÁLISIS Y DISEÑO

2.1. ENTRADAS

- **Datos de préstamos:** para que el programa funcione, se lee el archivo de "prestamos.lfa" que contiene los datos de los usuarios separados por comas y saltos de línea.
- **Definición de la clase préstamos:** se define una clase llamada préstamos que contiene los atributos que se vincularán a los datos del archivo de entrada.
- **Opciones del programa:** se definen las opciones que se colocarán en el menú, estas son cargar archivo, generar historial, listado de usuarios, listado de libros, estadísticas, generar prestamos vencidos y salir.

2.2. PROCESOS

- **Análisis léxico:** es el primer proceso del programa. Se analiza cada carácter del archivo leído y se verifican si son válidos o no. Se parte del alfabeto ya definido y se plantean condiciones dependiendo de si se encuentra o no el carácter. Así mismo se saltan líneas y se reconocen las comas que dividen los datos.
- **Verificación de las fechas:** en dicha parte se convierte el texto a un formato de fecha con ayuda de la importación de "datetime", así mismo después se verifica que cumpla con el formato y cantidad de caracteres.
- **Reportes:** con lo definido anteriormente se lee el archivo y se procesan los reportes. Se identifican los usuarios, sus libros prestados y las estadísticas. Así mismo, se verifica que haya un total de 6 campos por línea.
- **Almacenamiento de datos:** al leer y verificar todo, el programa almacena los datos de los prestamos temporalmente.
- **Main:** se define el main junto con sus opciones vinculando los métodos ya establecidos. Se usa un bucle while que finaliza al seleccionar la opción "7".
- **Verificación de archivo cargado:** Al ejecutar el programa se verifica a primera instancia que el archivo se haya cargado, de lo contrario no se tienen datos para leer.

2.3. SALIDAS

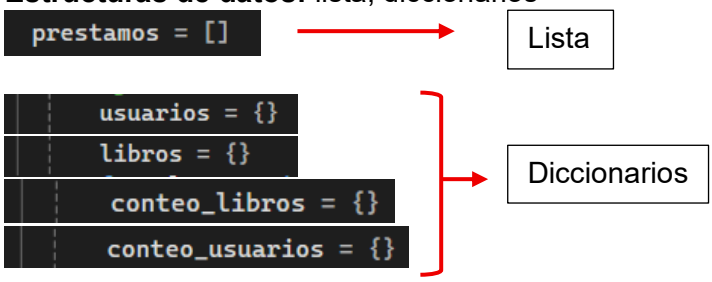
- **Menú:** se genera un menú para el usuario, en este se muestran las opciones disponibles. Cada opción devuelve un mensaje mientras por detrás se preparan las acciones para la salida de HTML.
- **Mensajes de verificación y error:** se muestran los mensajes de verificación en caso de que todo salga bien y de error si ocurre algo fuera de lugar como ingresar un carácter no permitido.
- **Salida de HTML:** el archivo HTML se construye en el proceso y al finalizar se crea una salida con los resultados. Se muestran las estadísticas de los préstamos, los préstamos vencidos y los reportes con el historial, listado de usuarios y libros prestados.

2.4. RESTRICCIONES

- **Pre-requisitos:** se necesita de un archivo ya hecho con los datos de los usuarios y libros.
- **Formato de archivo:** el archivo debe ser formato .lfa
- **Estructura del archivo:** para leer correctamente los datos, estos deben estar separados por comas (id_usuario, nombre_usuario, id_libro, titulo_libro, fecha_prestamo y fecha_devolucion) y tener 6 campos en total.
- **Caracteres:** los caracteres solo serán válidos si entran en el alfabeto (“ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789 ÁÉÍÓÚáéíóúÑñÜü ,.-@()”). Además, deben ser válidos bajo el formato UTF-8 de caracteres Unicode.
- **Fecha:** la fecha debe estar en formato yyyy-mm-dd.
- **Menú:** el menú solo posee 7 opciones. Si se selecciona una inexistente dará error y si se ingresa un carácter no numérico se avisará que es inválido.
- **Salida del reporte:** el formato de salida del reporte debe ser HTML.
- **Errores:** se deben manejar errores tanto para las entradas del usuario como para las acciones del programa.

2.5. ESTRUCTURAS USADAS

- **Estructuras de datos:** lista, diccionarios



- **Estructuras de control:** if, else, elif, try, except, while

```
#si no esta en el alfabeto y este no es un salto devuelve el error
if x in ("\n","\r"):
    continue #continua al siguiente bloque
if x not in Alfabeto_permitido:
    #si no esta en el alfabeto tira el error
    print(f"Error en linea {linea_error}, indice: {i} caracter no permitido {x}")
    valido=False
```

```
elif opc ==2:
    if ArchivoSeCargo == True:
        f.write(historial_prestamos())
        print("Historial de reportes guardado correctamente.\n")
    else:
        print("No se ha inicializado aun el archivo\n")
```

```python
def ValidarFecha(fecha: str) -> bool:
    try:
        datetime.strptime(fecha, "%Y-%m-%d")
        return True
    except ValueError:
        return False
```

```python
while(opc!=7):
```

- **Estructuras de clases:** clase prestamos

```python
class Prestamos:
    #clase prestamos
    def __init__(self, id_usuario, nombre_usuario, id_libro, titulo_libro, fecha_prestamo, fecha_devolucion):
        self.id_usuario = id_usuario
        self.nombre_usuario =  nombre_usuario
        self.id_libro=id_libro
        self.titulo_libro=titulo_libro
        self.fecha_prestamo=fecha_prestamo
        self.fecha_devolucion=fecha_devolucion
```

2.6. ERRORES

- Validación del alfabeto.

```python
def Validar_Alfabeto(linea:str, linea_error:int ):
    valido = True
    for i, x in enumerate(linea, start=1):
        #si no esta en el alfabeto y este no es un salto devuelve el error
        if x in ("\n","\r"):
            continue #continua al siguiente bloque
        if x not in Alfabeto_permitido:
            #si no esta en el alfabeto tira el error
            print(f"Error en linea {linea_error}, indice: {i} caracter no permitido {x}")
            valido=False

    return valido
```

- Validación de fecha. Se usa ValueError porque se dan valores incorrectos.

```python
def ValidarFecha(fecha: str) -> bool:
    try:
        datetime.strptime(fecha, "%Y-%m-%d")
        return True
    except ValueError:
        return False
```

- Validación de cantidad de campos (se necesitan 6, separados por comas)

```python
if len(partes)!=6: #verifica que haya los 6 datos esperados
    print(f"Error en la linea {linea_num}, cantidad de campos incorrectos")
    continue #si es incorrecto solo tira el error y sigue a el siguiente bloque
```

- Validar que el usuario ingrese caracteres permitidos en el menú.

```python
print("Ingrese la opcion que desea realizar:\n1.Cargar Archivo\n2.General
try:
    opc = int(input("Eleccion: "))
except ValueError:
    print("Error intento ingresar un simbolo invalido\n")
```

2.7. MÓDULOS

- **Datetime:** como única importación se usa “from datetime import datetime” que pertenece a la biblioteca estándar de Python.

III. ANÁLISIS LÉXICO

A continuación, se muestran los fragmentos de código en los que se trabaja el análisis léxico del proyecto.

1. Validar_Alfabeto

```python
#alfabeto que va a aceptar
Alfabeto = ("ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz"
"0123456789"
"ÁÉÍÓÚáéíóúÑñÜü"
" ,.-@()"
)

Alfabeto_permitido = set(Alfabeto)
```



```python
def Validar_Alfabeto(linea:str, linea_error:int ):
    valido = True
    for i, x in enumerate(linea, start=1):
        #si no esta en el alfabeto y este no es un salto devuelve el error
        if x in ("\n","\r"):
            continue #continua al siguiente bloque
        if x not in Alfabeto_permitido:
            #si no esta en el alfabeto tira el error
            print(f"Error en linea {linea_error}, indice: {i} caracter no permitido {x}")
            valido=False

    return valido
```

Se comienza definiendo los caracteres permitidos que se podrán escanear en el archivo de referencia. En este se incluyen letras mayúsculas y minúsculas junto con números, caracteres usados en el español (tildes, diéresis y la letra ñ) y símbolos especiales.

Posterior a eso se define una función para validar el alfabeto ya establecido. En el bucle “for” se empiezan a leer los caracteres. En la primera condición se continúa si se encontró el carácter mientras a la vez se ignoran saltos de línea mientras que en la siguiente se rechaza el carácter no admitido y se muestra un mensaje de error.

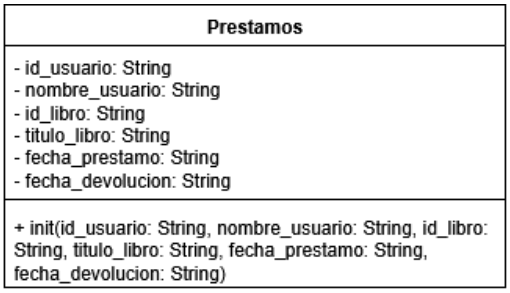
2. Cargar_Prestamos

```python
def cargar_prestamos(archivo):
    with open(archivo,"r", encoding="utf-8") as f:
        linea_num = 0;
        for linea in f:
            linea_num+=1
            linea = linea.strip()

            if not Validar_Alfabeto(linea, linea_num):
                continue #continua a la siguiente linea

            partes = linea.split(",") #separa id, nombre, idlibro etc

            if len(partes)!=6: #verifica que haya los 6 datos esperados
                print(f"Error en la linea {linea_num}, cantidad de campos incorrectos")
                continue #si es incorrecto solo tira el error y sigue a el siguiente bloque

            #asignamos las partes
            id_usuario, nombre_usuario, id_libro, titulo_libro, fecha_prestamo, fecha_devolucion = partes

            if not ValidarFecha(fecha_prestamo) or not ValidarFecha(fecha_devolucion):
                print(f"Error en la linea {linea_num}, formato de fecha incorrecto")
                continue

            p = Prestamos(id_usuario.strip(), nombre_usuario.strip(), id_libro.strip(), titulo_libro.strip(),
                          fecha_prestamo.strip(), fecha_devolucion.strip())


            prestamos.append(p)
```

Si bien dicha parte no pertenece al análisis léxico, se referencia el uso de la función “Validar_Alfabeto”. En este caso, después de leer el archivo, si se encuentra alguna línea no válida, se rechaza y se continúa con la otra.

IV. DIAGRAMAS

1. Diagrama de clases



Clase única de préstamos. En esta se definen todos los datos que se usarán en los métodos.

2. Diagrama de flujo

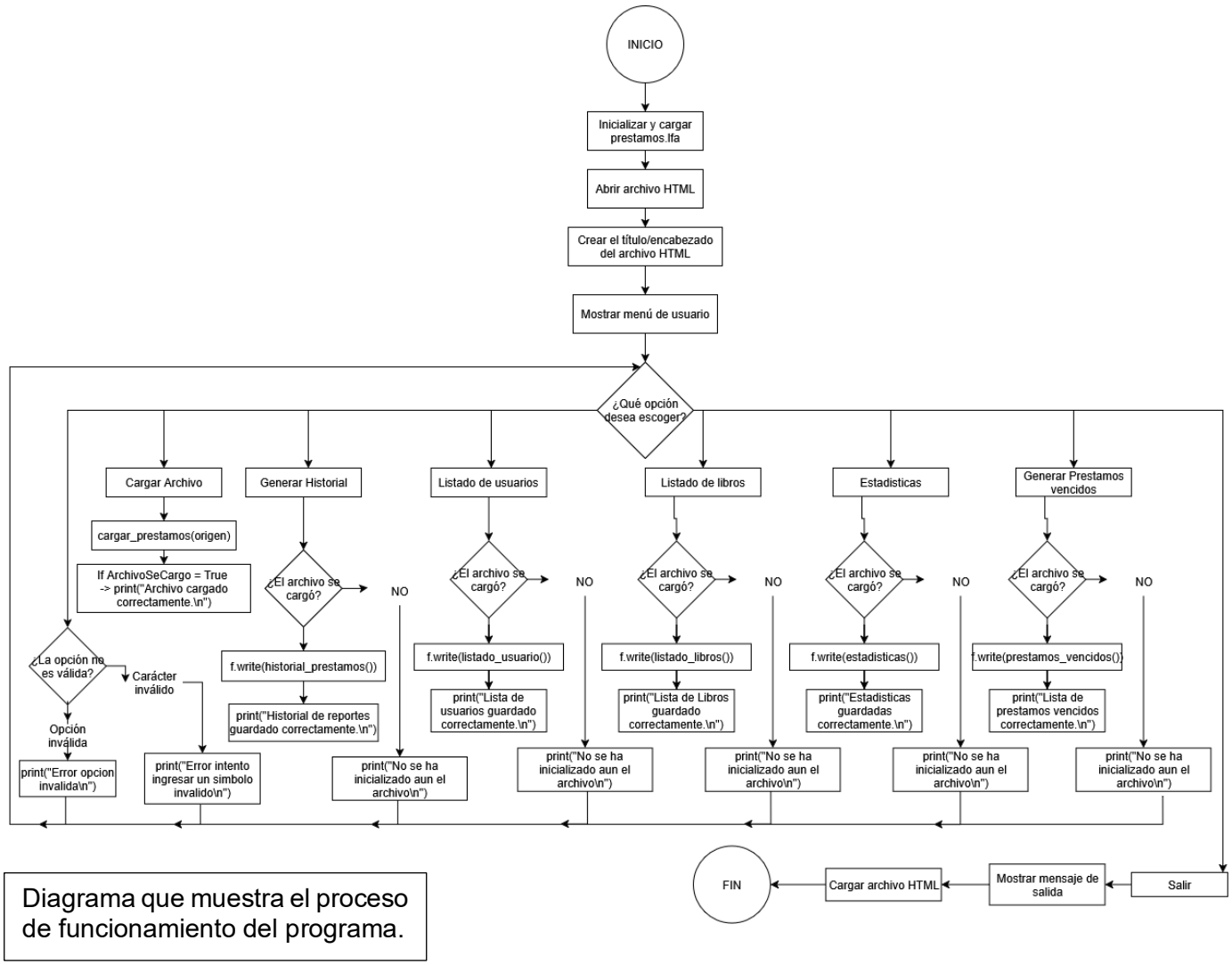


Diagrama que muestra el proceso de funcionamiento del programa.