|  |  |
| --- | --- |
| **Proyecto 1 - manual Tecnico** | |
| 202111134 | Mario René Mérida Taracena |

# Descripción del Proyecto

El sistema AFDGraph es una herramienta visual para diseñar, analizar y validar Autómatas Finitos Deterministas (AFD). Sus funcionalidades ahora incluyen:

* Carga de múltiples AFDs desde archivos .lfp (incluyendo manejo de errores léxicos).
* Visualización de grafos con Graphviz (mejorado para soportar transiciones complejas).
* Validación de cadenas con retroalimentación detallada.
* Generación de reportes en JPEG/PNG con:
  + Tablas de tokens (lexemas, tipos, posición).
  + Errores léxicos/sintácticos detectados.

# Bases Lógicas de Programación

## POO:

* Clases principales:
  + AFD: Modelo ampliado con validación de integridad.
  + Scanner: Nuevo analizador léxico basado en AFD.
  + AnalizadorAFD: Parser sintáctico para múltiples autómatas.

## Estructuras de Datos:

* LinkedHashMap para preservar orden de estados/transiciones.
* HashSet para alfabeto y estados finales (evita duplicados).

## Manejo de Archivos:

* Soporte para archivos con errores (continúa procesando AFDs válidos).
* Generación de imágenes temporales para gráficos.

# Descripción de los Archivos

|  |  |
| --- | --- |
| Archivo | Nuevas Funcionalidades |
| AFD.java | Métodos para verificar integridad del autómata (estado inicial, transiciones completas). |
| AnalizadorAFD.java | Procesamiento de múltiples AFDs en un archivo. Detección de errores sintácticos. |
| Scanner.java | Implementación de AFD léxico para tokenización. Registro de errores léxicos. |
| GeneradorGrafo.java | Uso de archivos temporales para Graphviz. Soporte para visualización de errores en grafos. |
| InterfazGrafica.java | Panel de errores integrado. Selección múltiple de AFDs en combobox. |

# Estructura del Proyecto

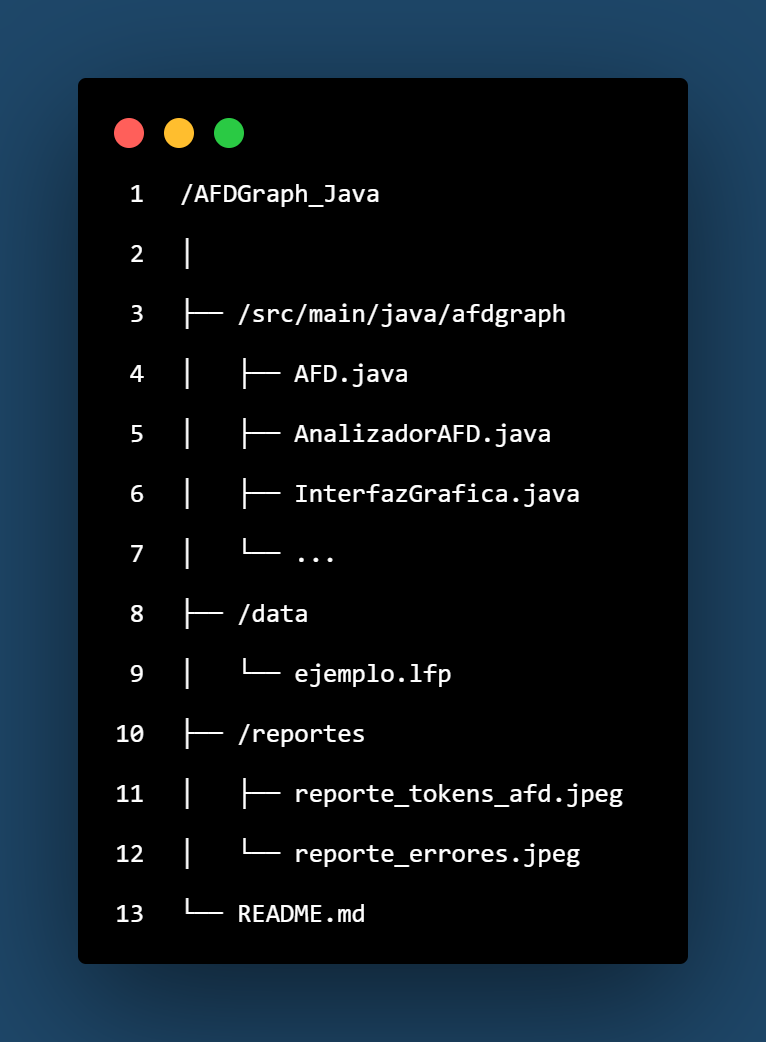


Figura 1. Estructura del proyecto  
Fuente: Elaboración Propia (2025)

# Librerías Empleadas

## Novedades:

* java.nio.file.Paths: Para manejo seguro de rutas en generación de reportes.
* java.text.SimpleDateFormat: Timestamps en nombres de reportes.
* java.util.concurrent: Procesamiento asíncrono de Graphviz.

## Uso de Graphviz:

* Requiere instalación previa del paquete dot.
* Comando: dot -Tpng archivo.dot -o salida.png.

# Cómo se Generaron los Reportes

1. **Tokens**:
   * Tabla con lexemas, tokens y posiciones (línea/columna).
2. **Errores**:
   * Lista de caracteres no válidos detectados.

# Compilación y Ejecución

## Compilación:

Ejecuta el siguiente comando en la terminal:

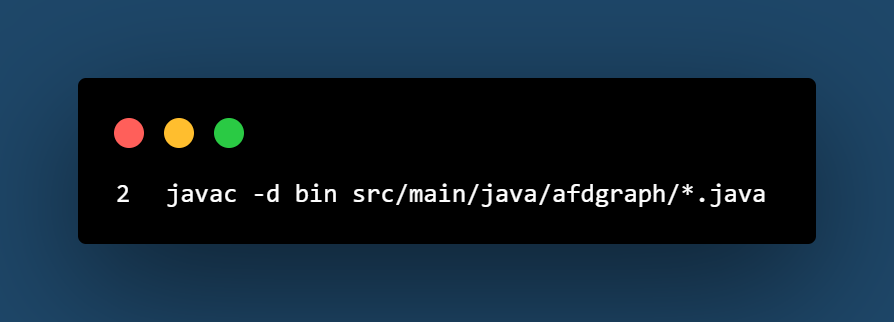


Figura 2. Compilación del Proyecto  
Fuente: Elaboración Propia (2025)

# Ejecución:

Ejecuta el programa con el siguiente comando:

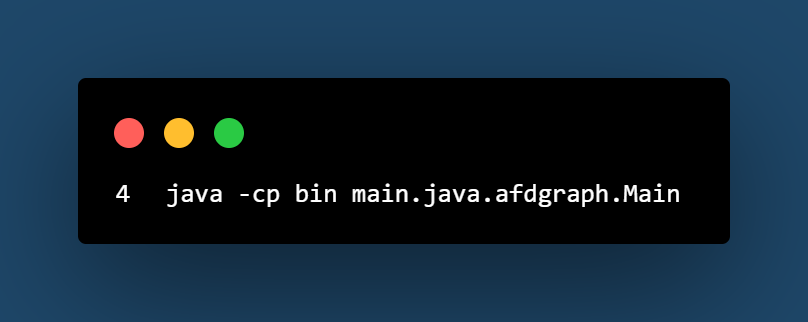


Figura 3. Ejecución del Proyecto  
Fuente: Elaboración Propia (2025)

# Consideraciones Técnicas

## Validación de Cadenas:

* El sistema ignora caracteres inválidos pero los reporta.
* Los AFDs incompletos se marcan pero no detienen la ejecución.

## Excepciones:

* Manejo de errores en carga de archivos y Graphviz.

## Extensibilidad:

* Diseñado para añadir nuevos tipos de autómatas (AFN, PDA).
* Plantilla de reportes adaptable (JSON/HTML futuro).