INFORME DE TRABAJO 1 (TP)

1ACC0218 – TEORÍA DE COMPILADORES - PRESENCIAL
Carrera de Ciencias de la Computación

Sección: 276
Docente: Peter Jonathan Montalvo García

Integrantes:

Código	Apellidos	Nombres
u202220230	Chipana Rios	Andre Angel
u202316342	Vásquez Trujillano	Edson Fabrizio

Ingreso al Github:: https://github.com/Britzio/TrabajoFinalCompiladores_Chipana_Vasquez

2025

ÍNDICE

ÍNDICE 2
Problemática y motivación 3
Objetivos 3
Gramática en ANTLR4 4
Referencias bibliográficas 9

Problemática y motivación

Los errores en la medicación de los pacientes pueden ser dañinos o incluso mortales. Según AMCP (s.f.), estos errores son los más comunes entre los errores médicos y perjudican a más de un millón de pacientes anualmente. Algunos de estos errores incluyen problemas con productos de la salud defectuosos, fallas en la administración de medicamentos, la deficiente educación del paciente ante el uso de medicamentos, errores en la prescripción dada por el especialista, entre otros.

Teniendo en cuenta este contexto, en los últimos años se han observado errores frecuentes en la prescripción de medicamentos debido a recetas ilegibles. Por ejemplo, Redacción Mag (2023) comparte un caso de un farmacéutico que no fue capaz de interpretar la receta médica de un cliente debido a que la letra era ilegible e incluso pudo estar escrita en jerga médica o abreviaturas específicas. Este caso, al igual que muchos otros, demuestra el desafío que enfrentan los farmacéuticos para comprender las indicaciones médicas de las recetas y proveer a las personas de los medicamentos correctos.

Ante esta problemática, proponemos una solución enfocada al desarrollo y uso de una gramática que sirva como validación para la prescripción de medicamentos. Esta gramática se implementará en el sistema del centro médico, permitiendo que el especialista de la salud ingrese la información de la receta médica o el tratamiento para su registro, de forma que cada dato registrado coincida con las reglas establecidas en la gramática. Por ejemplo, si se desea registrar el nombre de un medicamento, solo será aceptado si posee caracteres alfabéticos; en cambio, si se registra una dosis, el formato debe contener caracteres numéricos para la medida y caracteres alfabéticos para la unidad de medida.

Objetivos

El objetivo principal de este proyecto es solucionar o reducir los errores causados por prescripciones inadecuadas. Para ello, proponemos la creación de un lenguaje de programación que contenga reglas gramaticales destinadas a validar cada campo de la prescripción médica. Una vez la prescripción sea validada y registrada en el sistema del centro médico, la propia entidad puede entregar al paciente una versión impresa o digital, legible y clara, de su receta médica.

Además, como objetivo secundario, la propuesta contempla otras aplicaciones, tales como su integración con asistentes de voz para agilizar el registro de datos en comparación con el proceso manual tradicional, así como su uso en la generación automática de archivos para sistemas de base de datos (por ejemplo, en formatos .json, .csv, entre otros).

Gramática en ANTLR4

A fin de establecer las reglas gramaticales del lenguaje de programación, es necesario conocer los campos de una receta médica. Según el Instituto Nacional de Salud Mental (s.f.), la información que debe contener una receta médica es la siguiente:

Campo	**Tipo de carácteres**	**Descripción**
Paciente	Alfabético	Nombre completo del paciente
DNI	Numérico	Documento Nacional de Identidad o Código Único de Identificación de Extranjeros
Diagnóstico	Alfabético	Descripción de la enfermedad o estado de salud del paciente.
Medicamento	Alfabético	Nombre del producto farmacéutico recetado.
Concentración	Alfanumérico	Medida y unidad de medida de medicamento (por ejemplo, 500 mg).
Forma	Alfabético	Forma farmacéutica del medicamento (por ejemplo, tableta, jarabe, ampolleta, etc.).
Cantidad	Numérico	Cantidad total de medicamento que debe ingerir el paciente.
Expedición	Numérico y símbolos	Fecha de expedición en formato dd-mm-aa.
Caducidad	Numérico y símbolos	Fecha de vigencia de la receta en formato dd-mm-aa.
Indicación	Alfanumérico	Detalle de cómo y cuándo debe administrarse el medicamento
Vía	Alfabético	Vía de administración de medicamento.

En ese sentido, establecemos las reglas léxicas de la gramática siguiendo las convenciones descritas en la tabla.

Reglas léxicas originales:

```
PACIENTE : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ( ' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+)* ;
DNI : [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9];
DIAGNOSTICO : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ( ' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ )* ;
MEDICAMENTO : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ;
CONCENTRACION : [0-9]+ ' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ;
FORMA : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ;
```

CANTIDAD : [0-9]+ ;
EXPEDICION : [0-9][0-9] '-' [0-9][0-9] '-' [0-9][0-9] ;
CADUCIDAD : [0-9][0-9] '-' [0-9][0-9] '-' [0-9][0-9] ;
INDICACION : ([0-9]+ | [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ | ',' | '.' | ' ')+ ;
VIA : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ;

Sin embargo, estas reglas presentan definiciones muy similares o, en algunos casos, completamente idénticas. Esto genera ambigüedad léxica, ya que si una entrada cumple con las reglas léxicas de varios tokens, el analizador léxico no podrá identificar correctamente a qué token pertenece. Por ejemplo, si se ingresa la palabra “Hola”, el lexer no podrá determinar si corresponde al token PACIENTE, DIAGNOSTICO, MEDICAMENTO, FORMA, INDICACION o VIA, ya que todos permiten el ingreso de una sola palabra.

Para resolver este problema, optamos por establecer reglas más simples, usando descripciones más generales que eviten la ambigüedad y de forma que permitan establecer las reglas sintácticas esperadas.

Reglas léxicas:

PALABRAS : [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ (' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+)* ;
INT : [0-9] ;
CONCENTRACION : INT+ ' ' PALABRAS ;
FECHA : INT INT '-' INT INT '-' INT INT ;

Una vez establecidas las reglas léxicas, procedemos a describir las reglas sintácticas que dan sentido a nuestra gramática al crear definiciones de cómo se combinan los tokens para construir expresiones. A continuación, se muestran las subreglas sintácticas diseñadas para cada oración que debe reconocer el sistema:

Subreglas sintácticas:

- datosPacienteSentence:

  Arquitectura que describe la acción de ingresar el nombre completo y DNI de un paciente.

  Definición: 'Paciente:' WS* PALABRAS ',' WS* INT INT INT INT INT INT INT INT
  Ejemplo: “Paciente: José Raúl Ramírez Borgoña, 12345678”

- diagnosticoSentence:

  Arquitectura que describe la acción de ingresar el diagnóstico del paciente.

Definición: 'Diagnostico:' WS* PALABRAS
Ejemplo: “"Diagnostico: Arritmia severa”

- fechasSentence:

  Arquitectura que describe el ingreso de una fecha. En este caso, es la base para ingresar la fecha de expedición y la fecha de caducidad de la receta médica.

  Definición: 'Expede:' WS* FECHA WS* 'Caduce:' WS* FECHA
  Ejemplo: “Expede: 10-05-2025 Caduce: 18-10-2025”

- medicamentoSentence:

  Arquitectura que describe el ingreso de los datos de un medicamento en el siguiente orden: nombre, concentración, forma, vía, cantidad e indicación.

  Definición: 'Medicamento:' WS* PALABRAS ',' WS* CONCENTRACION ',' WS* PALABRAS ',' WS* PALABRAS ',' WS* INT+ ',' WS* PALABRAS '.'
  Ejemplo: "Medicamento: Paracetamol, 500 mg, Tableta, Oral, 5, Tomar cada ocho horas."

- registro:

  Arquitectura que une otras subreglas para generar una estructura de valor. Permite unir los datos de un paciente con su diagnóstico, las fechas de su receta médica y los medicamentos recetados.

  Definición: datosPacienteSentence diagnosticoSentence medicamentoSentence+ fechasSentence

Finalmente, establecemos unas reglas sintácticas simples para poner a prueba la gramática creada.

<u>Regla sintáctica:</u>

- prog:

  Arquitectura que recibe múltiples registros de recetas médicas.

  Definición: (registro)+ EOF

A continuación, se adjuntan imágenes de la gramática implementada en Google Colab, por medio del lenguaje de programación Python.

Link del Google Colab:

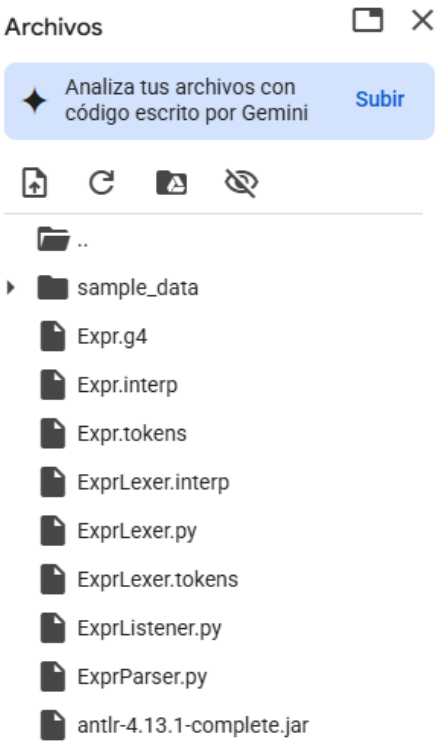
https://colab.research.google.com/drive/1I1O0Vng96My7Mp1aHi-eGVADGl71wcnM?usp=sharing

Gramática:

```
gramatica = """
grammar Expr;

// --- Reglas Sintácticas - Parser ---
prog:
    (registro)+ EOF
    ;

registro:
    datosPacienteSentence diagnosticoSentence medicamentoSentence+ fechasSentence
    ;

// --- SubReglas Sintácticas - Parser ---
datosPacienteSentence:
    'Paciente:' WS* PALABRAS ',' WS* INT INT INT INT INT INT INT INT
    ;

diagnosticoSentence:
    'Diagnostico:' WS* PALABRAS
    ;

fechasSentence:
    'Expede:' WS* FECHA WS* 'Caduce:' WS* FECHA
    ;

medicamentoSentence:
    'Medicamento:' WS* PALABRAS ',' WS* CONCENTRACION ',' WS* PALABRAS ',' WS* PALABRAS ',' WS* INT+ ',' WS* PALABRAS '.'
    ;

// --- Reglas Léxicas (Tokens) - Lexer ---
PALABRAS :   [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ( ' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ )* ;
INT :      [0-9] ;
CONCENTRACION : INT+ ' ' PALABRAS ;
FECHA :    INT INT '-' INT INT '-' INT INT ;

WS  : [ \\t\\r\\n]+ -> skip ;
"""

with open("Expr.g4", "w", encoding='utf-8', newline='\n') as f:
    f.write(gramatica)
```

Generar archivos G4:

```
[5]  # Genera varios archivos Lexer.py, Parser.py, .tokens, etc.
     #   Sirven para interpretar y ejecutar texto segun las reglas de la gramatica
     !java -jar antlr-4.13.1-complete.jar -Dlanguage=Python3 Expr.g4
```



Pruebas de gramática:

Se reconocen con éxito los ejemplos propuestos.

Gramática Final

```
grammar Expr;

// --- Reglas Sintácticas - Parser ---
prog:
    (registro)+ EOF
    ;

registro:
    datosPacienteSentence diagnosticoSentence medicamentoSentence+ fechasSentence
    ;

// --- SubReglas Sintácticas - Parser ---
datosPacienteSentence:
    'Paciente:' WS* PALABRAS ',' WS* INT INT INT INT INT INT INT INT
    ;

diagnosticoSentence:
    'Diagnostico:' WS* PALABRAS
    ;

fechasSentence:
    'Expede:' WS* FECHA WS* 'Caduce:' WS* FECHA
    ;

medicamentoSentence:
    'Medicamento:' WS* PALABRAS ',' WS* CONCENTRACION ',' WS* PALABRAS ',' WS* PALABRAS ',' WS* INT+ ',' WS* PALABRAS '.'
    ;

// --- Reglas Léxicas (Tokens) - Lexer ---
PALABRAS :   [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ ( ' ' [a-zA-ZáéíóúÁÉÍÓÚñÑ]+ )* ;
INT :      [0-9] ;
CONCENTRACION : INT+ ' ' PALABRAS ;
FECHA :    INT INT '-' INT INT '-' INT INT ;

WS  : [ \t\r\n]+ -> skip ;

```

Arquitectura del compilador

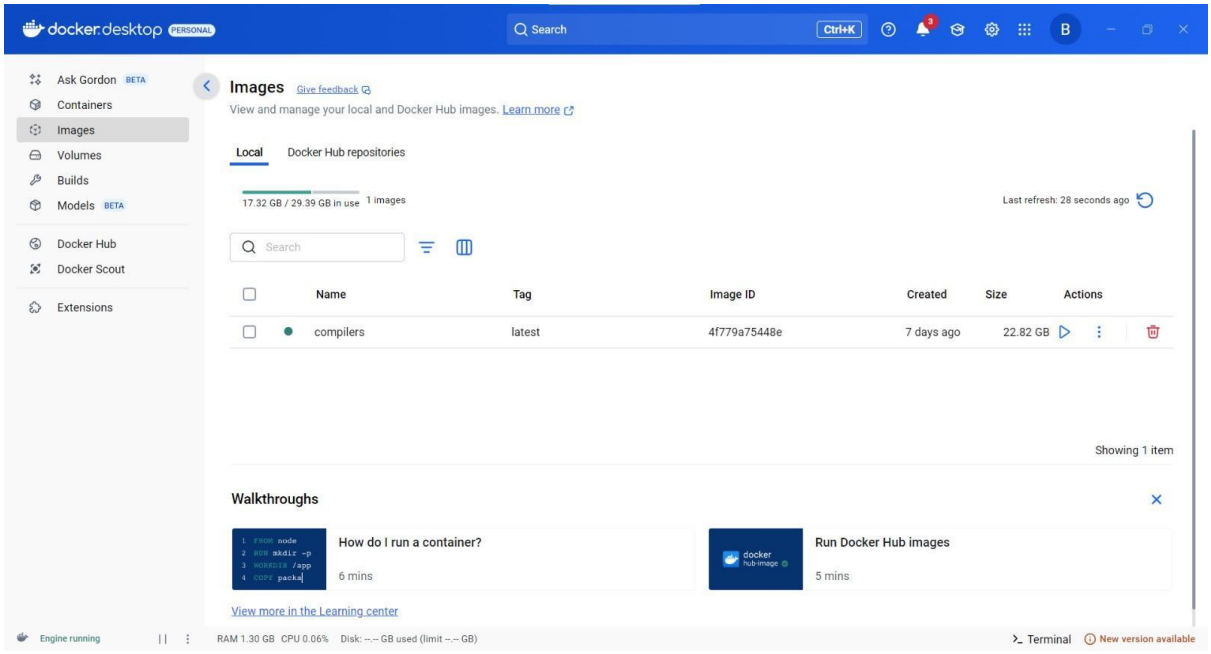
Como parte de la arquitectura del compilador, se implementó un entorno de ejecución aislado utilizando **Docker**, con el fin de garantizar la portabilidad, reproducibilidad y control sobre las herramientas necesarias para compilar y ejecutar la gramática desarrollada con **ANTLR4**.

Imagen Docker personalizada

Para ello, se construyó una imagen personalizada llamada “compilers”, que incluye:

- El lenguaje de programación Python y sus dependencias.
- El runtime y herramientas de ANTLR4 necesarias para el análisis léxico y sintáctico.
- Scripts de validación y formateo de salidas que conforman el driver del compilador.

Esta imagen funciona como el **entorno base del compilador**, asegurando que todos los componentes estén correctamente configurados sin importar el sistema operativo o las librerías del entorno local del usuario.

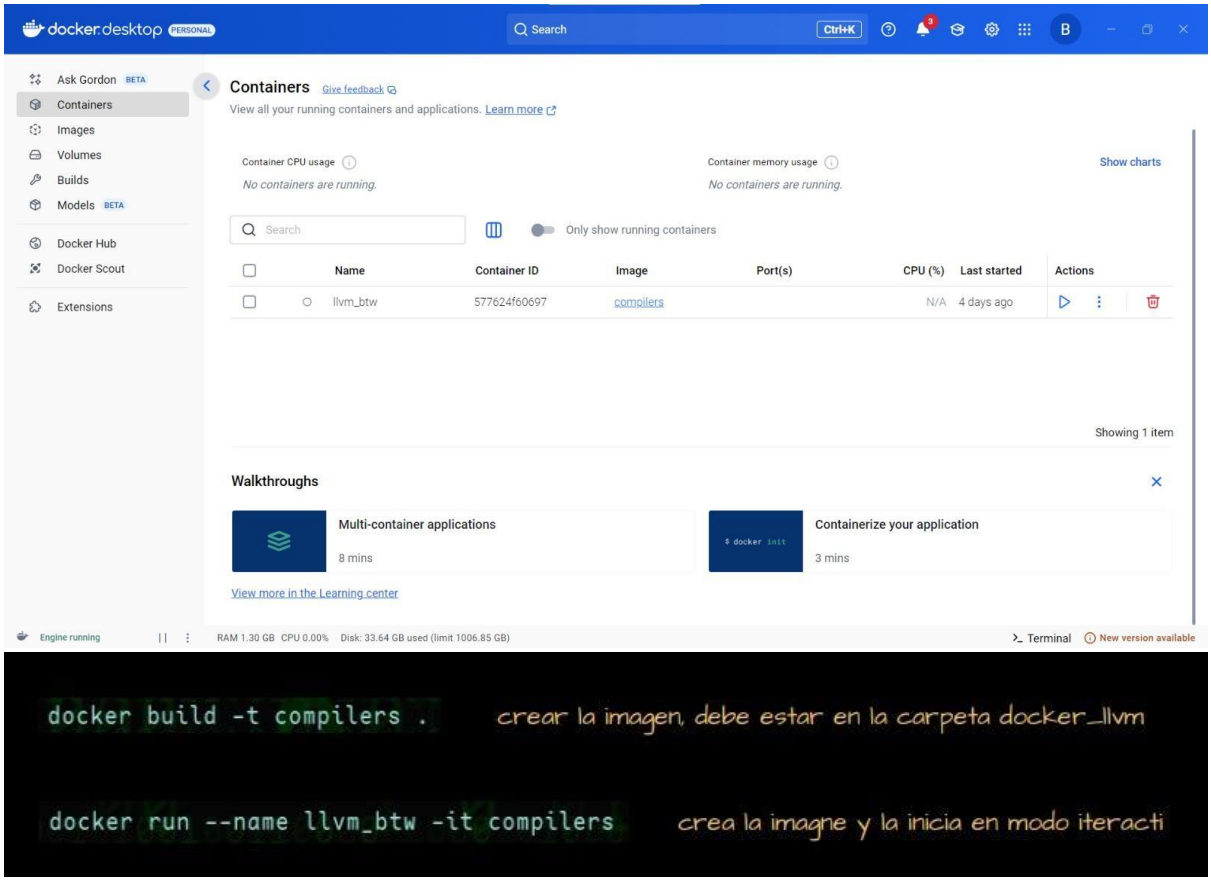


Contenedor de ejecución del compilador

A partir de esta imagen, se ejecuta un contenedor que representa la instancia activa del compilador. Este contenedor permite:

- Ingresar entradas simuladas (recetas médicas).
- Analizar su estructura gramatical conforme a la gramática definida.
- Validar su sintaxis y generar una salida estructurada (recetas legibles y resumen automatizado).

Este enfoque modular permite que el compilador sea fácilmente desplegado, probado o incluso escalado, en caso de que se requiera su integración con otros sistemas médicos o clínicos. Además, facilita la separación entre la construcción (build) del compilador y su ejecución, lo cual sigue los principios de una arquitectura compiladora moderna.



Plan de validación

Para validar la efectividad de la gramática desarrollada en ANTLR4, se diseñó un conjunto de pruebas que simulan distintas entradas que un especialista de salud podría ingresar en el sistema. El objetivo es verificar que la gramática:

1. **Reconozca correctamente entradas válidas** (conforme a las reglas sintácticas y léxicas definidas).

2. **Rechace entradas inválidas o malformadas**, como campos incompletos, formatos erróneos o datos ambiguos.

3. Una funcionalidad clave añadida al driver del compilador es la generación de una salida estructurada y legible, que presenta:
    - Los datos de cada paciente con su diagnóstico y lista de medicamentos de forma clara y ordenada.
    - Indicaciones, dosis, forma y frecuencia de administración expresadas en texto natural.

- Un resumen final automatizado que muestra el número total de pacientes procesados, el diagnóstico más frecuente y la cantidad de unidades recetadas por cada medicamento.

Metodología del plan de validación

- Se elaboró una colección de **casos de prueba positivos** (entradas válidas) y **casos de prueba negativos** (entradas con errores).
- Cada entrada fue procesada por el lexer y parser generado con ANTLR4 en Google Colab.
- Se documentó el resultado de cada prueba: aceptada o rechazada.
- Se registraron los errores detectados y se ajustaron las reglas si era necesario.

Caso	Entrada	Resultado
1	Paciente: Mario Chavez Raul, 12347478 Diagnostico: Tos fuerte Medicamento: Paracetamol, 500 mg, Tableta, Oral, 5, Tomar cada ocho horas. Expede: 02-05-25 Caduce: 01-10-25 Paciente: Carlos Méndez Ruiz, 11223344 Diagnostico: Sinusitis crónica Medicamento: Cefalexina, 500 mg, Tableta, Oral, 14, Tomar cada ocho horas. Medicamento: Loratadina, 10 mg, Tableta, Oral, 7, Tomar una vez al día. Medicamento: Salbutamol, 100 mcg, Inhalador, Inhalatoria, 1, Usar cada seis horas si hay dificultad respiratoria. Expede: 05-06-25 Caduce: 05-08-25	Aceptado **Respuesta**: (prog (registro (datosPacienteSentence Paciente: Mario Chavez Raul , 1 2 3 4 7 4 7 8) (diagnosticoSentence Diagnostico: Tos fuerte) (medicamentoSentence Medicamento: Paracetamol , 500 mg , Tableta , Oral , 5 , Tomar cada ocho horas .) (fechasSentence Expede: 02-05-25 Caduce: 01-10-25)) (registro (datosPacienteSentence Paciente: Carlos Méndez Ruiz , 1 1 2 2 3 3 4 4) (diagnosticoSentence Diagnostico: Sinusitis crónica) (medicamentoSentence Medicamento: Cefalexina , 500 mg , Tableta , Oral , 1 4 , Tomar cada ocho horas .) (medicamentoSentence Medicamento: Loratadina , 10 mg , Tableta , Oral , 7 , Tomar una vez al día .) (medicamentoSentence Medicamento: Salbutamol , 100 mcg , Inhalador , Inhalatoria , 1 , Usar cada seis horas si hay dificultad respiratoria .) (fechasSentence Expede: 05-06-25 Caduce: 05-08-25)) <EOF>)
2	Paciente: Juan Pérez Gómez, 12345678 Diagnostico: Gripe común Medicamento: Paracetamol, 500 mg, Tableta, Oral, 5, Tomar cada ocho horas. Medicamento: Ibuprofeno, 10 mg, Tableta, Oral, 5, Tomar cada cinco horas. Expede: 01-05-25 Caduce: 01-06-25l	Aceptado **Respuesta**: (prog (registro (datosPacienteSentence Paciente: Juan Pérez Gómez , 1 2 3 4 5 6 7 8) (diagnosticoSentence Diagnostico: Gripe común) (medicamentoSentence Medicamento: Paracetamol , 500 mg , Tableta , Oral , 5 , Tomar cada ocho horas .) (medicamentoSentence Medicamento: Ibuprofeno , 10 mg , Tableta , Oral

		, 5 , Tomar cada cinco horas .) (fechasSentence Expede: 01-05-25 Caduce: 01-06-25)) <EOF>)
3	Paciente: Juan Pérez Gómez, 12345678 Medicamento: Paracetamol, 500 mg, Tableta, Oral, 5, Tomar cada ocho horas. Diagnostico: Gripe común Medicamento: Ibuprofeno, 10 mg, Tableta, Oral, 5, Tomar cada cinco horas. Expede: 01-05-25 Caduce: 01-06-25	Rechazado **Error**: line 2:26 mismatched input 'Medicamento:' expecting 'Diagnostico:' line 3:26 extraneous input 'Diagnostico:' expecting {'Expede:', 'Medicamento:'} **Explicación**: Las líneas de medicamento deben ir después del diagnóstico.
4	Paciente: Luis Alberto Gómez, 556688 Diagnostico: Infección urinaria Medicamento: Fenazopiridina, 100 mg, Tableta, Oral, p, Tomar cada ocho horas por tres días. Expede: 15-06-25 Caduce: 15-07-25	Rechazado **Error**: line 2:26 mismatched input 'Diagnostico:' expecting INT line 3:78 mismatched input 'p' expecting {INT, WS} **Explicación**: El número de DNI espera 8 dígitos y la cantidad de medicamento debe ser entero.

Resultados de la validación

- Reconocimiento y validación exitosa de entradas bien estructuradas: La gramática logró validar el 100 % de las prescripciones que seguían el formato definido, incluyendo campos como el nombre del paciente, diagnóstico, fechas y detalles del medicamento.

- Ambigüedad léxica resuelta: Se corrigieron los problemas de ambigüedad léxica presentes en versiones anteriores mediante la simplificación y generalización de los tokens, lo que permitió una identificación más precisa de cada campo.

- Rechazo adecuado de entradas incorrectas: Las entradas mal formateadas o incompletas fueron correctamente rechazadas por el sistema, cumpliendo con el objetivo de evitar errores en el registro de datos.

- Generación automática de salida estructurada: A diferencia de versiones previas, el sistema ahora genera un formato de salida legible, ordenado y coherente, mostrando de manera clara la información de cada paciente, sus recetas y datos médicos relevantes.

- Análisis de datos en tiempo real: Como parte del procesamiento, el sistema produce un resumen final que identifica automáticamente:

    - El número total de pacientes registrados.
    - El diagnóstico más frecuente.
    - El total de unidades recetadas por medicamento.

Este resultado evidencia que la propuesta no solo es capaz de validar prescripciones, sino también de procesar y presentar información médica de forma automatizada, clara y útil para la toma de decisiones en entornos clínicos.

Conclusiones

1. La prescripción médica ilegible representa un riesgo crítico para la seguridad del paciente, ya que puede generar errores en la administración de medicamentos, lo cual está documentado como una de las principales causas de incidentes médicos a nivel mundial.

2. El desarrollo de una gramática formal mediante ANTLR4 se presenta como una solución innovadora para validar de manera automática las prescripciones médicas, asegurando que los campos requeridos cumplan con un formato correcto y legible.

3. La implementación de reglas léxicas y sintácticas específicas permite estandarizar la información médica, evitando ambigüedades y mejorando la precisión en la interpretación de recetas por parte de los profesionales de salud y farmacéuticos.

4. El diseño de sub reglas sintácticas para cada componente de la receta médica contribuye a un modelo robusto y extensible, que no solo valida campos como nombre, diagnóstico, y fecha, sino que también detalla la forma, dosis y vía de administración del medicamento.

5. La propuesta tiene aplicaciones prácticas más allá de la validación de recetas, ya que podría integrarse con tecnologías como asistentes de voz o sistemas de gestión de datos médicos, mejorando la eficiencia operativa en centros de salud.

6. Reducir los errores de prescripción mediante un sistema automatizado mejora la calidad del servicio de salud, facilita la labor médica y farmacéutica, y protege la integridad de los pacientes.

7. El sistema basado en gramática implementada con ANTLR4 fue validado satisfactoriamente con entradas reales. La correcta detección de errores, el formato claro y el procesamiento automatizado de datos confirman que esta solución es funcional, escalable y útil para centros de salud que buscan reducir errores en la prescripción médica.

Referencias bibliográficas

Academy of Managed Care Pharmacy (AMCP). (s.f.) Medication Errors. *AMCP*. Recuperado el 12 de mayo de 2025, de https://www-amcp-org.translate.goog/concepts-managed-care-pharmacy/medication-errors?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge#:~:text=Common%20causes%20of%20medication%20error,and%20lack%20of%20patient%20education.

Redacción Mag. (30 de diciembre de 2023). Un farmacéutico pide ayuda para entender lo que dice esta receta: ¿logras descifrarlo?. *El Comercio*. https://elcomercio.pe/mag/virales/video-viral-farmaceutico-pide-ayuda-para-entender-lo-que-dice-esta-receta-espana-nnda-nnrt-noticia/?ref=ecr

Instituto Nacional de Salud Mental. (s.f.). Requisitos que debe contener una receta médica. *Ministerio de Salud.* Recuperado el 14 de mayo de 2025, de https://www.insm.gob.pe/comites/farmacoterapeutico/archivos/170707%20REQUISITOS%20QUE%20DEBE%20CONTER%20UNA%20RECETA.pdf