

**TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA**

**SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**

SEMESTRE:

AGOSTO-DICIEMBRE 2025

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA Y SERIE :

LENGUAJES Y AUTOMATAS 2

TÍTULO:

PROYECTO: TRADUCTOR DE MEDICAMENTOS

UNIDAD A EVALUAR:

2DA UNIDAD

NOMBRES Y NÚMEROS DE CONTROL DE LOS ALUMNOS:

MOJICA FAJARDO JOSE ANGEL 22210322

ZARZA MORALES JOSE DIEGO 22210366

NOMBRE DEL MAESTRO:

GAXIOLA VEGA LUIS ALFONSO

FECHA:

02 DE OCTUBRE DEL 2025

Índice

PROYECTO: TRADUCTOR DE MEDICAMENTOS.....	3
Información General.....	3
JUSTIFICACIÓN DEL PROYECTO.....	4
Relevancia Académica.....	4
Relevancia Social.....	4
ARQUITECTURA DEL SISTEMA.....	5
ESPECIFICACIONES TÉCNICAS.....	6
DEMOSTRACIÓN.....	7
ANÁLISIS LÉXICO.....	11
Tokens Definidos.....	11
ANÁLISIS SEMÁNTICO.....	12
ANÁLISIS SINTÁCTICO.....	13
Gramática Libre de Contexto (actualizada).....	13
Ejemplos válidos.....	14
Ejemplos inválidos.....	14
REGLAS SEMÁNTICAS.....	14
REGLAS DE PRODUCCIÓN.....	15
ÁRBOL SINTÁCTICO.....	16
Analizador lexico y sintactico.....	17
Códigos.....	20
MANUAL DE USUARIO.....	30
Tabla de Contenidos.....	30
Introducción.....	30
REQUISITOS DEL SISTEMA.....	31
INSTALACIÓN.....	31
INTERFAZ DEL SISTEMA.....	31
FUNCIONALIDADES PRINCIPALES.....	32
GUÍA DE USO PASO A PASO.....	32
MEDICAMENTOS DISPONIBLES.....	33
SOLUCIÓN DE PROBLEMAS.....	33
PREGUNTAS FRECUENTES.....	34
INFORMACIÓN DE CONTACTO Y CRÉDITOS.....	35
AVISO LEGAL.....	35

PROYECTO: TRADUCTOR DE MEDICAMENTOS

Información General

Proyecto: Traductor de Medicamentos usando Autómatas Finitos

Curso: Lenguajes y Autómatas 2

Objetivo: Desarrollar un sistema web que permita traducir nombres químicos de medicamentos a nombres comerciales, proporcionar información farmacológica, calcular dosis personalizadas y aplicar técnicas de análisis léxico, sintáctico y semántico.

JUSTIFICACIÓN DEL PROYECTO

Relevancia Académica

- Aplicación de autómatas finitos deterministas para reconocimiento de patrones.
- Implementación de gramáticas libres de contexto para validación de estructuras.
- Procesamiento de lenguaje natural enfocado al dominio farmacéutico.
- Ejemplo práctico de análisis léxico, sintáctico y semántico.

Relevancia Social

- Facilita la comprensión de prescripciones médicas.
- Reduce errores en la identificación de medicamentos.
- Brinda educación farmacológica básica al usuario.
- Mejora la accesibilidad a información médica.

ARQUITECTURA DEL SISTEMA

Componentes Principales

- **Interfaz Web:** Desarrollada en HTML, CSS y JavaScript.
- **Módulo Léxico:** Tokeniza entradas y normaliza texto.
- **Motor Sintáctico:** Aplica gramáticas libres de contexto para validar instrucciones.
- **Motor Semántico:** Comprueba relaciones válidas entre medicamentos, condiciones y alergias.
- **Base de Datos Local:** Diccionario interno de medicamentos.
- **Exportador:** Genera PDF de la receta mediante librerías JavaScript.

Flujo de Procesamiento

Entrada de Usuario



Normalización de Texto



Análisis Léxico



Reconocimiento de Patrones



Validación Sintáctica y Semántica



Consulta en Base de Datos



Presentación de Resultados

ESPECIFICACIONES TÉCNICAS

Elemento	Detalle
Lenguajes	HTML5, CSS3, JavaScript
Librerías	html2canvas, jsPDF
Autómata	AFD para reconocimiento de tokens
Gramática	Gramática libre de contexto
Almacenamiento	LocalStorage / SessionStorage
Salida	Interfaz + Exportación PDF
SO Compatible	Windows, Linux, macOS
Navegadores compatibles	Chrome, Firefox, Edge

DEMOSTRACIÓN

- **Versión inicial:** estructura básica, búsqueda directa.
1ra DEMOSTRACIÓN

Captura del programa antes de la revisión del 19 de septiembre

The screenshot shows a web browser window with the URL `localhost:5500/medicine-translator/`. The page has a light green background. At the top, a dark teal header contains the title 'Traductor de Medicamentos' and the subtitle 'Sistema basado en Autómatas Finitos'. Below the header, there is a search bar with the text 'paracetamol' and a teal 'BUSCAR' button. The results are displayed in a white box with a green border. The 'Resultado' section shows 'Nombre Químico: p-acetaminofenol' and 'Nombre Comercial: Paracetamol (Acetaminofén)'. The 'Información Farmacológica' section states: 'Analgésico y antipirético de uso común. Actúa inhibiendo la síntesis de prostaglandinas en el SNC. Indicado para dolor leve a moderado y fiebre.' At the bottom, there is a section titled 'Ejemplos de Búsqueda' with three cards for 'Paracetamol', 'Aspirina', and 'Ibuprofeno', each showing its chemical name.

Captura del programa con la actualización de lo que nos pidió

The screenshot shows the updated version of the web application. The layout is more complex. At the top, there is a 'Ficha del paciente' section with fields for 'Nombre' (with a sub-field 'Nombre completo'), 'Edad (años)' (with a sub-field 'Ej. 30'), 'Sexo' (with a dropdown 'Otro / Prefiere no decir'), 'Estatura (cm)' (with a sub-field 'Ej. 175'), 'Peso (kg)' (with a sub-field 'Ej. 70'), and 'Motivo / Condición' (with a dropdown 'Resfriado común / Gripe'). Below this, there is a section for 'Alergias (marcar las aplicables)' with checkboxes for 'Ninguna', 'Paracetamol', 'Ibuprofeno / AINEs', and 'Penicilinas', and a text input for 'Sulfonamidas Otro:'. A teal button labeled 'CALCULAR DOSIS Y RECOMENDACIONES' is positioned below the allergies section. The main header is a dark teal box with the title 'Traductor de Medicamentos' and the subtitle 'Sistema basado en Autómatas Finitos'. Below the header, there is a search bar with the placeholder text 'Ingrese nombre químico o comercial...' and a teal 'BUSCAR' button. At the bottom, there is a section titled 'Ejemplos de Búsqueda'.

- **Versión actual:** análisis de entrada, traducción, validaciones y exportación.

127.0.0.1:5500/ProgramaPrincipal/index.html

MEDCOG
Traductor de Medicamentos

MEDCOG — Tecnología al servicio de tu bienestar
Analizador (Estudiante)

"Tu salud, nuestra prioridad."

1 Cuenta — 2 Ficha — 3 Resultados

1 — Cuenta

Crea una cuenta o inicia sesión. Tu historial se guardará en esta cuenta en el navegador.

Usuario (email o nombre)
hjeemplo@correo.com

Contraseña
Contraseña

Crear cuenta Iniciar sesión

Sin sesión — crea o inicia sesión en el paso 1

Cómo usar

1. Crear cuenta o iniciar sesión (Paso 1).
2. Completar la ficha del paciente (Paso 2).
3. Ver recomendaciones y guardar historial (Paso 3).

"Decisiones médicas inteligentes."

En caso de duda, consulta a tu médico. Esta herramienta es informativa.

"La salud no espera. MedCOG responde."

Última actualización: Octubre 2025 — Versión 2.5 — "Tecnología al servicio de tu bienestar."

1 Cuenta — 2 Ficha — 3 Resultados

2 — Ficha del paciente

Llena los datos del paciente. Si no los sabes todos, ingresa los que tengas.

Nombre completo
Diego Zarza Morales

Edad (años)
22

Sexo
M

Estatura (cm)
180

Peso (kg)
110

Motivo / Condición
Dolor de garganta

Alergias (marcar si aplica)

Ninguna ☐

Paracetamol ☒

Ibuprofeno ☐

Aspirina ☐

Atrás Siguiente Ver recomendaciones

Conectado: diegopro123@gmail.com

Cómo usar

1. Crear cuenta o iniciar sesión (Paso 1).
2. Completar la ficha del paciente (Paso 2).
3. Ver recomendaciones y guardar historial (Paso 3).

"Decisiones médicas inteligentes."

En caso de duda, consulta a tu médico. Esta herramienta es informativa.



MEDCOG
Traductor de Medicamentos

MEDCOG — Tecnología al servicio de tu bienestar

Analizador (Estudiante)

Tu salud, nuestra prioridad.

1 Cuenta

2 Ficha

3 Resultados

3 — Recomendaciones

"Recomendaciones confiables al instante."

Sugerencias orientativas. Cada recomendación incluye un botón para leerla en voz alta.

Paracetamol (p-acetaminofenol)

Analgésico y antipirético • Dolor leve a moderado, fiebre

Dosis aproximada: 1100 - 1000 mg • Cada 4-6 horas

ADVERTENCIA: **alergia relacionada. NO administre sin consultar al médico.**

Escuchar

Agregar a selección

Atrás

Guardar historial en mi cuenta

Descargar historial (PDF)

Conectado:
diegopro123@gmail.com

Cómo usar

1. Crear cuenta o iniciar sesión (Paso 1).

2. Completar la ficha del paciente (Paso 2).

3. Ver recomendaciones y guardar historial (Paso 3).


"Decisiones médicas inteligentes."

En caso de duda, consulta a tu médico. Esta herramienta es informativa.

La salud no espera. MedCOG responde.

Última actualización: Octubre 2025 — Versión 2.5 — "Tecnología al servicio de tu bienestar."

9

**MEDCOG**
Decisiones médicas inteligentes

Receta Médica
Cuenta: diegopro123@gmail.com
Generado: 10/17/2025, 3:37:30 PM

Paciente
Nombre: Diego Zarza Morales
Edad: 22 años
Sexo: M

Medidas
Estatura: 180 cm
Peso: 110 kg
Condición: sore-throat

Prescripción

#	Medicamento	Dosis	Frecuencia	Duración
No hay medicamentos seleccionados.				

Firma y sello del médico

DISCLAIMER: Esta receta es orientativa y educativa. NO sustituye la evaluación médica. Antes de tomar cualquier medicamento, consulte con un profesional de la salud. En caso de urgencia, contacte servicios médicos.

"Tecnología al servicio de tu bienestar." — MEDCOG

ANÁLISIS LÉXICO

El analizador léxico convierte las entradas de texto en **tokens** que representan cada parte significativa de la instrucción.

Tokens Definidos

Token	Expresión Regular / Ejemplo	Descripción
IDENTIFICADOR	[a-zA-ZáéíóúÁÉÍÓÚñ]+	Nombre del paciente o palabras comunes
MEDICAMENTO	`(paracetamol	aspirina
CONDICION	`(resfriado	gripe
NUMERO	[0-9]+(\.[0-9]+)?	Dosis, edad, peso, frecuencia
UNIDAD	`(mg	g
OPERADOR	`(cada	por
SEPARADOR	`,`	:
INSTRUCCION	`(buscar	traducir
ALERGIA	`(ninguna	paracetamol
EOF	—	Fin de instrucción

ANÁLISIS SEMÁNTICO

El analizador semántico verifica la **coherencia lógica** de los tokens detectados:

1. El medicamento debe existir en la base de datos.
2. La condición debe tener tratamiento asociado.
3. No se recomienda un medicamento si el usuario es alérgico a él.
4. La dosis debe ser segura para la edad y peso ingresados.
5. La instrucción debe comenzar con una palabra reservada válida.
6. Los valores numéricos deben estar en rangos clínicos aceptables.
7. No se permiten palabras fuera del conjunto léxico definido.

ANÁLISIS SINTÁCTICO

La gramática define cómo deben organizarse los tokens para que una instrucción sea **válida estructuralmente**.

Gramática Libre de Contexto (actualizada)

<Programa> ::= <Instruccion> <Expresion>

<Instruccion> ::= "buscar" | "traducir" | "calcular" |
"exportar"

<Expresion> ::= <Medicamento>
| <Medicamento> <Condicion>
| <Medicamento> <Dosis> <Tiempo>

<Medicamento> ::= MEDICAMENTO

<Condicion> ::= "para" CONDICION

<Dosis> ::= NUMERO UNIDAD

<Tiempo> ::= "cada" NUMERO "horas"

<Paciente> ::= <Nombre> <Edad> <Peso> <Alergia>

<Nombre> ::= IDENTIFICADOR

<Edad> ::= NUMERO "años"

<Peso> ::= NUMERO "kg"

<Alergia> ::= ALERGIA

Ejemplos válidos

buscar paracetamol

traducir ibuprofeno para dolor

calcular paracetamol 500 mg cada 6 horas

Ejemplos inválidos

calcu aspirina 500 mg → error léxico

buscar 123 → error sintáctico

paracetamol cada 8 horas → falta instrucción

REGLAS SEMÁNTICAS

1. La condición debe existir en la base de datos.
2. El medicamento debe tener relación válida con la condición.
3. Si hay alergia → error semántico.
4. Dosis numérica debe estar en el rango permitido.
5. El tiempo debe expresarse en horas válidas.
6. Edad y peso deben estar en rango clínico.
7. La instrucción debe iniciar con palabra reservada.

REGLAS DE PRODUCCIÓN

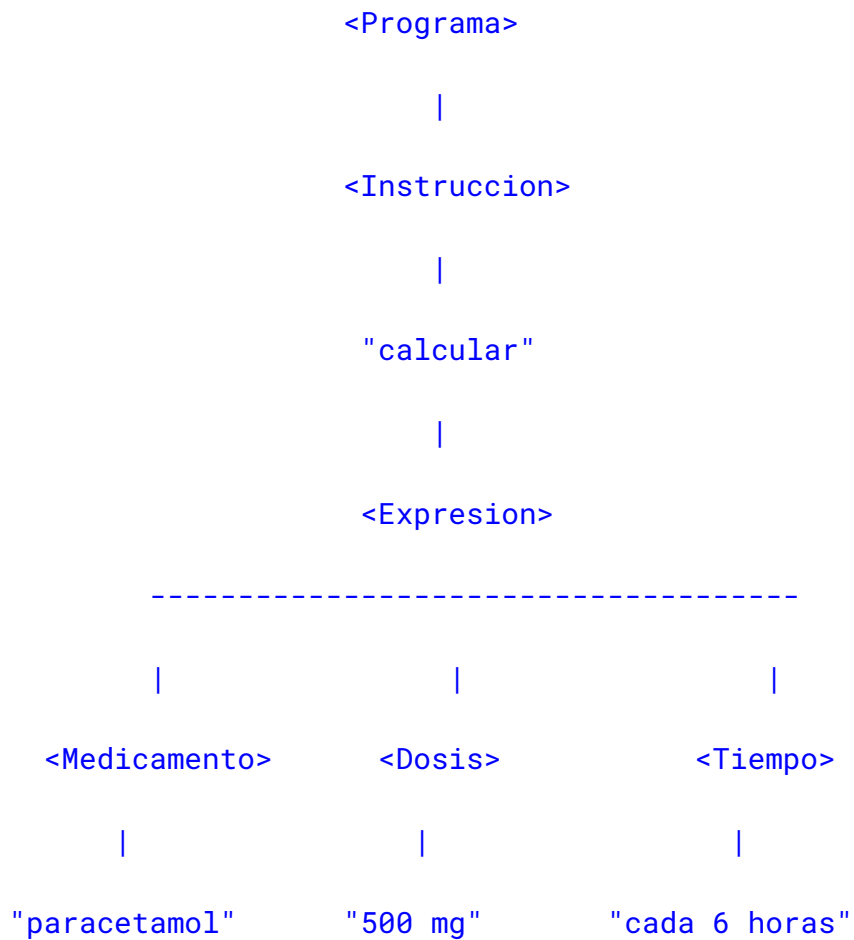
1. Programa → Instruccion Expresion
2. Instruccion → "buscar" | "traducir" | "calcular" | "exportar"
3. Expresion → Medicamento | Medicamento Condicion | Medicamento Dosis Tiempo
4. Medicamento → MEDICAMENTO
5. Condicion → "para" CONDICION
6. Dosis → NUMERO UNIDAD
7. Tiempo → "cada" NUMERO "horas"
8. Paciente → Nombre Edad Peso Alergia
9. Nombre → IDENTIFICADOR
10. Edad → NUMERO "años"
11. Peso → NUMERO "kg"
12. Alergia → ALERGIA

ÁRBOL SINTÁCTICO

Entrada:

calcular paracetamol 500 mg cada 6 horas

Árbol:



Analizador lexico y sintactico

Imágenes del analizador

Trabajo Adjunto — Analizador

Volver

Editor Léxico Sintáctico Reglas

Lenguaje: Aritmético (expresiones) ▼

Editor — pega el código

```
// Ejemplos: prueba los tres modos
// arithmetic: (3 + 4) * 2 == 14
// assignment: x = (a + 3) * 2;
x = (a + 3) * 2;
```

Generar tokens Generar árbol Choose File No file chosen Cargar ejemplo

Descripción:

Consiste en analizar código mediante tres modos: léxico, sintáctico y de reglas.

El usuario puede pegar o escribir código en el editor y luego generar tokens para ver el análisis léxico o generar el árbol para observar la estructura sintáctica. Además, se pueden cargar ejemplos o archivos con código para analizarlos automáticamente.

Trabajo Adjunto — Analizador

Volver

Editor Léxico Sintáctico Reglas

Lenguaje: Aritmético (expresiones) ▼

Resultados del Análisis Léxico

#	Token	Lexema	Pos
---	-------	--------	-----

Descripción:

En esta sección, el programa procesa el código ingresado en el editor y descompone el texto en tokens, que son las unidades básicas del lenguaje (por ejemplo, identificadores, operadores, números o símbolos).

Los resultados se muestran en una tabla con columnas como "Token", "Lexema" y "Pos", donde se indican el tipo de token, el valor reconocido y su posición dentro del código.

En este caso, la tabla aparece vacía, lo que sugiere que aún no se ha ejecutado el análisis o no se detectaron tokens válidos.


Trabajo Adjunto — Analizador			
<div> <div>Editor</div> <div>Léxico</div> <div>Sintáctico</div> <div>Reglas</div> </div> <div>Lenguaje: Aritmético (expresiones) ▼</div>			
Resultados del Análisis Léxico			
#	Token	Lexema	Pos
1	SYM	/	0
2	SYM	/	1
3	IDENT	Ejemplos	3
4	UNKNOWN	:	11
5	IDENT	prueba	13
6	IDENT	los	20
7	IDENT	tres	24
8	IDENT	modos	29
9	SYM	/	35
10	SYM	/	36
11	IDENT	arithmetic	38
12	UNKNOWN	:	48
13	SYM	(50
14	NUMBER	3	51
15	SYM	+	53
16	NUMBER	4	55
17	SYM)	56
18	SYM	*	58
19	NUMBER	2	60
20	OP	==	62
21	NUMBER	14	65

Descripción:

El sistema toma el texto ingresado (en este caso, las líneas de ejemplo con operaciones y comentarios) y lo divide en tokens, que son las unidades más pequeñas del lenguaje. Cada token se clasifica y se muestra en una tabla con cuatro columnas:

- **#**: número de orden del token.
- **Token**: tipo de elemento detectado (por ejemplo, símbolo, identificador, número u operador).
- **Lexema**: valor exacto del texto que fue reconocido.
- **Pos**: posición donde aparece el lexema dentro del código original.

Por ejemplo, se identifican símbolos como “/”, identificadores como “Ejemplos” o “prueba”, números como “3” y “4”, y operadores como “==”.

 **Trabajo Adjunto — Analizador** Volver

Editor Léxico Sintáctico **Reglas**

Lenguaje: Aritmético (expresiones) ▼

Reglas de Gramática y Tokens


```
// Gramática de ejemplo (editable)
<program> ::= { <stmt> }
<stmt> ::= <ident> "=" <expr> ";" | <expr> ";"
<expr> ::= <term> { ("+"|"-" ) <term> }
<term> ::= <factor> { ("*"|" /" ) <factor> }
<factor> ::= NUMBER | IDENT | "(" <expr> ")"
```

Guardar reglas Restaurar por defecto

Descripción:

En esta sección, el programa permite ver o editar las reglas gramaticales que describen la estructura del lenguaje que se está analizando. Dichas reglas indican cómo se forman las expresiones, asignaciones y factores a partir de combinaciones de tokens.

En otras palabras, este módulo define la lógica sintáctica que el analizador usa para construir el árbol de análisis. Gracias a estas reglas, el sistema puede reconocer si el código ingresado cumple con la sintaxis correcta y cómo deben agruparse sus componentes.

 **Trabajo Adjunto — Analizador** Volver

Editor Léxico **Sintáctico** Reglas Lenguaje: Aritmético (expresiones) ▼

Árbol Sintáctico Generado

```
Program
ExpressionStatement
Binary (=)
  Identifier: x
  Binary (*)
    Paren
      Binary (+)
        Identifier: a
        Number: 3
      Number: 2
```

Descripción:

El funcionamiento que se muestra en esta imagen corresponde al módulo sintáctico del analizador, donde el sistema genera y visualiza el árbol sintáctico del código ingresado.

En esta parte, el programa interpreta la estructura jerárquica de la expresión según las reglas gramaticales definidas previamente. Cada nodo del árbol representa una parte del código, como operadores, identificadores o números, mostrando cómo se combinan para formar una sentencia válida.

Códigos

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1"
/>
  <title>Trabajo Adjunto – Analizador (Standalone)</title>
  <link rel="stylesheet" href="analyzer.css" />
</head>
<body>
  <div class="modal-window" role="dialog" aria-label="Trabajo
Adjunto">
    <div class="modal-header">
      <div style="display:flex;align-items:center;gap:12px;">
        
        <div class="modal-title">Trabajo Adjunto – Analizador</div>
      </div>
      <div class="modal-actions">
        <a href="../../../ProgramaPrincipal/index.html" class="ghost"
title="Volver a la app">Volver</a>
      </div>
    </div>

    <div class="modal-body">
      <div class="top-row">
        <div class="tabs">
          <button id="tabEditor" class="black-tab
active-tab">Editor</button>
          <button id="tabLexer" class="black-tab">Léxico</button>
          <button id="tabParser"
class="black-tab">Sintáctico</button>
          <button id="tabGrammar" class="black-tab">Reglas</button>
        </div>

        <div class="lang-select">
          <label for="langMode">Lenguaje:</label>
          <select id="langMode">
            <option value="arithmetic">Aritmético
```

```

(expresiones)</option>
    <option value="assignment">Asignaciones</option>
    <option value="custom">Otro (editable)</option>
</select>
</div>
</div>

<div id="screens" class="screens">
    <section id="screenEditor" class="screen active">
        <h3>Editor - pega el código</h3>
        <textarea id="sourceEditor" spellcheck="false">// Ejemplos:
prueba los tres modos
// arithmetic: (3 + 4) * 2 == 14
// assignment: x = (a + 3) * 2;
x = (a + 3) * 2;
</textarea>
        <div class="controls">
            <button id="runLex" class="black-btn">Generar
tokens</button>
            <button id="runParse" class="black-btn">Generar
árbol</button>
            <input id="fileInput" type="file" accept=".txt,.cpp" />
            <button id="loadExample" class="ghost">Cargar
ejemplo</button>
        </div>
    </section>

    <section id="screenLexer" class="screen" hidden>
        <h3>Resultados del Análisis Léxico</h3>
        <table id="tokensTable" class="tokens-table">

<thead><tr><th>#</th><th>Token</th><th>Lexema</th><th>Pos</th></tr></thead>

        <tbody></tbody>
        </table>
    </section>

    <section id="screenParser" class="screen" hidden>
        <h3>Árbol Sintáctico Generado</h3>
        <div id="treeArea" class="tree-area"
aria-live="polite"></div>
    </section>

```

```

    <section id="screenGrammar" class="screen" hidden>
      <h3>Reglas de Gramática y Tokens</h3>
      <textarea id="grammarEditor"
class="grammar-editor"></textarea>
      <div class="controls">
        <button id="saveGrammar" class="black-btn">Guardar
reglas</button>
        <button id="resetGrammar" class="ghost">Restaurar por
defecto</button>
      </div>
    </section>
  </div>
</div>
</div>

<script src="analyzer.js"></script>
</body>
</html>

```

```

:root{ --primary:#1F5C9A; --ivory:#F8F5ED; --black:#000;
--muted:#4d6b68; --card:#fff; --shadow:0 20px 50px rgba(0,0,0,0.15);
font-family:"Segoe UI", Roboto, Arial, sans-serif; }
*{box-sizing:border-box}
body{ margin:0; background:
linear-gradient(180deg,var(--ivory),#f4f6f7); display:flex;
align-items:center; justify-content:center; min-height:100vh;
padding:24px; }
.modal-window{ width:95%; max-width:1100px; background:var(--card);
border-radius:10px; box-shadow:var(--shadow); overflow:hidden; }
.modal-header{ background:var(--primary); color:#fff; padding:14px
18px; display:flex; justify-content:space-between;
align-items:center; }
.modal-title{ font-weight:800; font-size:1.1rem; }
.modal-actions .ghost{ background:transparent;border:1px solid
rgba(255,255,255,0.25); color:#fff; padding:8px 10px;
border-radius:6px; text-decoration:none; }
.modal-body{ padding:16px; }
.top-row{ display:flex; justify-content:space-between;
align-items:center; gap:12px; margin-bottom:12px; }

```

```

.tabs{ display:flex; gap:8px; }
.black-tab{ background:#000;color:#fff;border:none;padding:8px
10px;border-radius:6px;cursor:pointer;font-weight:700 }
.active-tab{ box-shadow:0 8px 20px rgba(0,0,0,0.18);
transform:translateY(-2px) }
.lang-select{ color:#0b3b65 }
.screens .screen{ padding:8px 0; }
.screen h3{ margin:6px 0 10px 0; color:var(--primary) }
#sourceEditor{ width:100%; min-height:180px; border-radius:8px;
border:1px solid #e6eef2; padding:10px; font-family:monospace;
font-size:14px; background:#fbfdff; }
.controls{ margin-top:10px; display:flex; gap:8px;
align-items:center; flex-wrap:wrap; }
.black-btn{ background:#000;color:#fff;border:none;padding:10px
14px;border-radius:8px; cursor:pointer; font-weight:700 }
.ghost{ background:transparent;border:1px solid
#dcdcdc;padding:8px;border-radius:8px; cursor:pointer }
.tokens-table{ width:100%; border-collapse:collapse; margin-top:8px;
background:#fff; border-radius:6px; overflow:hidden }
.tokens-table th, .tokens-table td{ padding:10px; border-bottom:1px
solid #f0f4f3; text-align:left; font-family:monospace }
.tree-area{ min-height:260px; border-radius:8px; padding:12px;
background:linear-gradient(180deg,#fff,#fbfffe); border:1px dashed
rgba(31,92,154,0.12) }
.grammar-editor{ width:100%; min-height:200px; border-radius:8px;
padding:10px; border:1px solid #e6eef2; font-family:monospace }
@media (max-width:700px){ .modal-window{ width:100%; } .top-row{
flex-direction:column; align-items:flex-start; gap:8px; } }

```

```

// Standalone analyzer (lexer + parser) - open
Analizador/analyzer.html
function escapeHtml(s){ return
(s||'').toString().replace(/&/g,'&amp;').replace(/</g,'&lt;').replace
(/>/g,'&gt;'); }

const tabEditor = document.getElementById('tabEditor');
const tabLexer = document.getElementById('tabLexer');
const tabParser = document.getElementById('tabParser');
const tabGrammar= document.getElementById('tabGrammar');

```



```
const screenEditor = document.getElementById('screenEditor');
const screenLexer = document.getElementById('screenLexer');
const screenParser = document.getElementById('screenParser');
const screenGrammar = document.getElementById('screenGrammar');

const sourceEditor = document.getElementById('sourceEditor');
const runLex = document.getElementById('runLex');
const runParse = document.getElementById('runParse');
const tokensTableBody = document.querySelector('#tokensTable tbody');
const treeArea = document.getElementById('treeArea');
const grammarEditor = document.getElementById('grammarEditor');
const saveGrammar = document.getElementById('saveGrammar');
const resetGrammar = document.getElementById('resetGrammar');
const fileInput = document.getElementById('fileInput');
const langMode = document.getElementById('langMode');
const loadExample = document.getElementById('loadExample');

function showScreen(name) {
  [screenEditor, screenLexer, screenParser,
screenGrammar].forEach(s => s.hidden = true);
  [tabEditor, tabLexer, tabParser, tabGrammar].forEach(t =>
t.classList.remove('active-tab'));
  if(name === 'editor') { screenEditor.hidden=false;
tabEditor.classList.add('active-tab'); sourceEditor.focus(); }
  if(name === 'lexer') { screenLexer.hidden=false;
tabLexer.classList.add('active-tab'); }
  if(name === 'parser') { screenParser.hidden=false;
tabParser.classList.add('active-tab'); }
  if(name === 'grammar') { screenGrammar.hidden=false;
tabGrammar.classList.add('active-tab'); }
}
tabEditor.addEventListener('click', () => showScreen('editor'));
tabLexer.addEventListener('click', () => showScreen('lexer'));
tabParser.addEventListener('click', () => showScreen('parser'));
tabGrammar.addEventListener('click', () => showScreen('grammar'));

fileInput?.addEventListener('change', (ev) => {
  const f = ev.target.files && ev.target.files[0];
  if(!f) return;
  const r = new FileReader();
  r.onload = () => sourceEditor.value = r.result;
  r.readAsText(f);
});
```

```
});

loadExample?.addEventListener('click', () => {
  const mode = langMode.value;
  if(mode === 'arithmetic') sourceEditor.value = '(3 + 4) * 2 == 14';
  else if(mode === 'assignment') sourceEditor.value = 'x = (a + 3) *
2;';
  else sourceEditor.value = '// ejemplo\nx = (b + 5) * 2;';
});

function tokenize(src){
  const tokens=[]; let i=0;
  const isLetter = c=>/[a-zA-Z_]/.test(c); const isDigit =
c=>/[0-9]/.test(c);
  while(i<src.length){
    const ch = src[i];
    if(/\s/.test(ch)){ i++; continue; }
    if(isLetter(ch)){ let j=i+1; while(j<src.length &&
/[a-zA-Z0-9_]/.test(src[j])) j++;
tokens.push({type:'IDENT',lexeme:src.slice(i,j),pos:i}); i=j;
continue; }
    if(isDigit(ch)){ let j=i+1; while(j<src.length &&
/[0-9\./]/.test(src[j])) j++;
tokens.push({type:'NUMBER',lexeme:src.slice(i,j),pos:i}); i=j;
continue; }
    const two = src.substr(i,2);
    if(['==','!=','<=','>','=','&&','||'].includes(two)){
tokens.push({type:'OP',lexeme:two,pos:i}); i+=2; continue; }
    if(['+','-','*','/','%','(',')','{','}','<','>'].includes(ch)){
tokens.push({type:'SYM',lexeme:ch,pos:i}); i++; continue; }
    tokens.push({type:'UNKNOWN',lexeme:ch,pos:i}); i++;
  }
  return tokens;
}

function renderTokens(tokens){
  tokensTableBody.innerHTML=''; tokens.forEach((t,idx)=>{ const
tr=document.createElement('tr'); tr.innerHTML =
`<td>${idx+1}</td><td>${t.type}</td><td>${escapeHtml(t.lexeme)}</td><
td>${t.pos}</td>`; tokensTableBody.appendChild(tr); });
}
```

```

runLex?.addEventListener('click', ()=>{ const src =
sourceEditor.value||''; const toks = tokenize(src);
renderTokens(toks); showScreen('lexer'); });

function tokenizeForParser(src) {
  const out=[]; let i=0; const isLetter=c=>/[a-zA-Z_]/.test(c); const
isDigit=c=>/[0-9]/.test(c);
  while(i<src.length){
    const ch=src[i];
    if(/\s/.test(ch)){ i++; continue; }
    if(isLetter(ch)){ let j=i+1; while(j<src.length &&
/[a-zA-Z0-9_]/.test(src[j])) j++;
    out.push({type:'IDENT',value:src.slice(i,j)}); i=j; continue; }
    if(isDigit(ch)){ let j=i+1; while(j<src.length &&
/[0-9\.]\/.test(src[j])) j++;
    out.push({type:'NUMBER',value:src.slice(i,j)}); i=j; continue; }
    if(src.substr(i,2)==='=='){ out.push({type:'OP',value:'=='});
i+=2; continue; }
    if('+-*/'.includes(ch)){ out.push({type:'OP',value:ch}); i++;
continue; }
    if('=;(){}[],<>'.includes(ch)){ out.push({type:'SYM',value:ch});
i++; continue; }
    i++;
  }
  return out;
}

function parseSource(source, mode='arithmetic'){
  const tokens = tokenizeForParser(source); let pos=0;
  function peek(){ return tokens[pos]||null; } function consume(){
return tokens[pos++]||null; }
  function expect(type,val){ const t=peek();
if(!t||t.type!==type|| (val!==undefined&&t.value!==val)) throw new
Error(`se esperaba ${type}${val?(' '+val):''} en pos ${pos}`); return
consume(); }

  function parseFactor(){ const t=peek(); if(!t) throw new
Error('Factor inesperado EOF'); if(t.type==='NUMBER'){ consume();
return {type:'NumberLiteral',value:t.value}; } if(t.type==='IDENT'){
consume(); return {type:'Identifier',name:t.value}; }
if(t.type==='SYM'&&t.value==='('){ consume(); const e=parseExpr();
expect('SYM',')'); return {type:'Paren',expr:e}; } throw new

```

```

Error('Factor inválido cerca de '+JSON.stringify(t)); }
function parseTerm(){ let node=parseFactor();
while(peek() && peek().type==='OP' && (peek().value==='*' || peek().value===
='/')){ const op=consume().value; const right=parseFactor();
node={type:'BinaryExpression',operator:op,left:node,right}; } return
node; }

function parseExpr(){ let node=parseTerm();
while(peek() && peek().type==='OP' && (peek().value==='+' || peek().value===
='-')){ const op=consume().value; const right=parseTerm();
node={type:'BinaryExpression',operator:op,left:node,right}; }
if(peek() && ((peek().type==='OP' && peek().value==='=') || (peek().type===
'SYM' && peek().value==='='))){ const op=consume().value; const
right=parseExpr();
node={type:'BinaryExpression',operator:op,left:node,right}; } return
node; }

function parseAssignment(){ const id=expect('IDENT');
expect('SYM','='); const expr=parseExpr();
if(peek() && peek().type==='SYM' && peek().value===';') consume(); return
{type:'Assignment',id:id.value,expr}; }

function parseExpressionStmt(){ const expr=parseExpr();
if(peek() && peek().type==='SYM' && peek().value===';') consume(); return
{type:'ExpressionStatement',expr}; }

const body=[];
if(mode==='assignment'){ while(pos<tokens.length){ if(peek() &&
peek().type==='IDENT' && tokens[pos+1] && tokens[pos+1].type==='SYM'
&& tokens[pos+1].value==='=') body.push(parseAssignment()); else
body.push(parseExpressionStmt()); } }
else { while(pos<tokens.length) body.push(parseExpressionStmt()); }
return {type:'Program', body};
}

function createNodeElement(node){
const el=document.createElement('div'); el.style.marginLeft='8px';
el.style.padding='4px 0';
if(!node){ el.textContent='(empty)'; return el; }
const title=document.createElement('div');
title.style.fontWeight='700';
switch(node.type){
case 'Program': title.textContent='Program';
el.appendChild(title); node.body.forEach(c=>

```

```

el.appendChild(createNodeElement(c)); return el;
    case 'Assignment': title.textContent=`Assignment → ${node.id}`;
el.appendChild(title); el.appendChild(createNodeElement(node.expr));
return el;
    case 'ExpressionStatement':
title.textContent='ExpressionStatement'; el.appendChild(title);
el.appendChild(createNodeElement(node.expr)); return el;
    case 'BinaryExpression': title.textContent=`Binary
(${node.operator})`; el.appendChild(title);
el.appendChild(createNodeElement(node.left));
el.appendChild(createNodeElement(node.right)); return el;
    case 'NumberLiteral': title.textContent=`Number: ${node.value}`;
el.appendChild(title); return el;
    case 'Identifier': title.textContent=`Identifier: ${node.name}`;
el.appendChild(title); return el;
    case 'Paren': title.textContent='Paren'; el.appendChild(title);
el.appendChild(createNodeElement(node.expr)); return el;
    default: title.textContent=JSON.stringify(node);
el.appendChild(title); return el;
}
}

runParse.addEventListener('click', () => {
    const src = sourceEditor.value||''; const mode = langMode.value ||
'arithmetic';
    treeArea.innerHTML='';
    try{ const ast = parseSource(src, mode);
treeArea.appendChild(createNodeElement(ast)); showScreen('parser'); }
catch(e){ treeArea.innerHTML = `<div
style="color:#b00000;font-weight:700;">Error al parsear:
${escapeHtml(String(e.message||e))}</div>`; showScreen('parser'); }
});

const defaultGrammar = `// Gramática de ejemplo (editable)
<program> ::= { <stmt> }
<stmt> ::= <ident> "=" <expr> ";" | <expr> ";"
<expr> ::= <term> { ("+"|"-" ) <term> }
<term> ::= <factor> { ("*"|" / ") <factor> }
<factor> ::= NUMBER | IDENT | "(" <expr> ")"
`;
grammarEditor.value = localStorage.getItem('analyzer_grammar') ||
defaultGrammar;

```

```
saveGrammar?.addEventListener('click', ()=> {  
  localStorage.setItem('analyzer_grammar', grammarEditor.value);  
  alert('Reglas guardadas.');
```



```
resetGrammar?.addEventListener('click', ()=> { grammarEditor.value =  
  defaultGrammar; localStorage.removeItem('analyzer_grammar');  
  alert('Reglas restauradas.');
```



```
showScreen('editor');
```

MANUAL DE USUARIO

Tabla de Contenidos

- Introducción
- Requisitos del Sistema
- Instalación
- Interfaz del Sistema
- Funcionalidades Principales
- Guía de Uso Paso a Paso
- Medicamentos Disponibles
- Solución de Problemas
- Preguntas Frecuentes
- Información de Contacto y Créditos
- Aviso Legal

Introducción

El Traductor de Medicamentos es un sistema basado en autómatas finitos deterministas (AFD) que permite:

- Traducir nombres químicos a nombres comerciales.
- Proporcionar información farmacológica.
- Calcular dosis personalizadas según datos del paciente.
- Validar instrucciones mediante análisis léxico, sintáctico y semántico.

REQUISITOS DEL SISTEMA

Mínimos:

- Navegador moderno (Chrome 90+, Firefox 88+, Edge 90+)
- JavaScript habilitado
- Resolución mínima 1024x768

Archivos necesarios:

proyecto/

├─ index.html

├─ script.js

└─ styles.css

INSTALACIÓN

1. **Modo Local:** Descargar los archivos y abrir `index.html` en un navegador.
2. **Servidor Web:** Subir los archivos a un servidor y acceder mediante URL.

INTERFAZ DEL SISTEMA

1. **Encabezado** — Título y subtítulo.
2. **Ficha del Paciente** — Datos personales y médicos.
3. **Buscador de Medicamentos** — Entrada de texto.
4. **Resultados** — Medicamento, dosis y advertencias.
5. **Ejemplos** — Búsquedas rápidas.

FUNCIONALIDADES PRINCIPALES

- Búsqueda inteligente de medicamentos.
- Traducción química → comercial.
- Validación de condiciones y alergias.
- Cálculo de dosis por edad y peso.
- Exportación de resultados a PDF.

GUÍA DE USO PASO A PASO

1. Ingresar datos del paciente.
2. Seleccionar condición médica.
3. Indicar alergias si existen.
4. Buscar o seleccionar medicamento.
5. Calcular y visualizar dosis.
6. Exportar la receta en PDF.

MEDICAMENTOS DISPONIBLES

Nombre Comercial	Principio Activo	Uso	Dosis Adulto
Paracetamol	p-acetaminofenol	Analgésico, antipirético	500–1000 mg / 4–6 h
Aspirina	Ácido acetilsalicílico	Dolor / inflamación	500–1000 mg / 4–6 h
Ibuprofeno	Ácido isobutilfenil	Dolor / inflamación	200–400 mg / 4–6 h
Omeprazol	Omeprazol	Úlceras, reflujo	20–40 mg / 24 h
Amoxicilina	Amoxicilina	Infecciones bacterianas	500–875 mg / 8 h
Cetirizina	Cetirizina	Alergias	10 mg / 24 h

SOLUCIÓN DE PROBLEMAS

Problema: Medicamento no encontrado

- Verificar ortografía o usar nombre comercial.
- Revisar si el medicamento existe en la base de datos.

Problema: No se calcula la dosis

- Verificar que se haya ingresado edad y peso válidos.

Problema: Alergia no reconocida

- Solo se permiten alergias definidas en el sistema.

Problema: Botón no responde

- Revisar si JavaScript está habilitado.
- Recargar la página.

PREGUNTAS FRECUENTES

- ¿Funciona sin internet? Sí, localmente.
- ¿Se guardan mis datos? No, todo es local.
- ¿Puedo automedicarme con esto? No, solo es educativo.
- ¿Puedo agregar más medicamentos? Sí, editando `script.js`.
- ¿Puedo imprimir los resultados? Sí, desde el navegador o exportar a PDF.

INFORMACIÓN DE CONTACTO Y CRÉDITOS

Proyecto: Traductor de Medicamentos con Autómatas Finitos

Curso: Lenguajes y Autómatas 2

Institución: Tecnológico Nacional de México — Instituto Tecnológico de Tijuana

Desarrolladores:

- Mojica Fajardo José Ángel — 22210322
- Zarza Morales José Diego — 22210366

Profesor: Gaxiola Vega Luis Alfonso

AVISO LEGAL

Este sistema es un proyecto académico con fines educativos. La información mostrada no sustituye la consulta médica profesional. Ante cualquier condición médica, consulte a un especialista.