Aplicación Web Automatizada de Facturas y Generación de Scripts SQL mediante Reconocimiento Óptico y Lenguajes Formales

A.M. Marin Alfonso[1], J.S. Villarraga Ariza[2], L.A. Medina Romero[3]

1. ammarin@unillanos.edu.co Cod: 160004521, *Ing. Sistemas.*
2. jsvillarraga@unillanos.edu.co Cod: 160004546, *Ing. Sistemas.*
3. luis.medina@unillanos.edu.co Cod: 160004146, *Ing. Sistemas.*

Facultad de Ciencias Básicas e Ingenierías.
Universidad de los Llanos, Villavicencio-Meta, Colombia

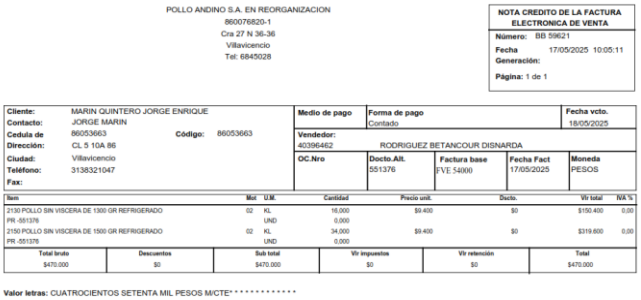
***Resumen*— Se presenta el desarrollo de una aplicación web implementada en Python y Streamlit, orientada a la automatización de la extracción y estructuración de datos provenientes de facturas en formato imagen o PDF. El sistema emplea reconocimiento óptico de caracteres (OCR), a través de EasyOCR y pdfplumber, así como técnicas de procesamiento de texto basadas en expresiones regulares, permitiendo la identificación precisa de información clave como datos del cliente, detalles de la factura y productos facturados. Las pruebas evidenciaron alta exactitud en la extracción de datos para documentos bien estructurados y de buena calidad visual. Se resalta la aplicabilidad de los lenguajes formales y los autómatas en la digitalización eficiente de documentos administrativos, contribuyendo a la mejora de procesos contables mediante la integración automática de información.**

Palabras clave: Expresiones regulares, extracción de texto, automatización, OCR, Streamlit.

I. Introducción

El presente trabajo describe el desarrollo de una aplicación web implementada en Python, utilizando la biblioteca Streamlit, cuyo objetivo es automatizar la extracción y estructuración de información contenida en facturas digitales o escaneadas (imágenes y archivos PDF). La aplicación integra técnicas de reconocimiento óptico de caracteres (OCR), mediante el motor Tesseract, para convertir el contenido visual de los documentos en texto digital. Posteriormente, se emplean expresiones regulares para identificar y segmentar los datos clave, incluyendo la información del cliente, detalles de la factura e ítems facturados, facilitando así la generación automática de scripts SQL para su inserción en bases de datos relacionales.

El desarrollo de este sistema se enmarca en el curso de Lenguajes de Programación y busca poner en práctica conceptos fundamentales de la teoría de la computación, especialmente aquellos relacionados con lenguajes formales y autómatas. Las expresiones regulares utilizadas corresponden a lenguajes regulares reconocidos por autómatas finitos deterministas, permitiendo la extracción precisa de estructuras específicas, como fechas, identificadores y montos. Si bien la solución demuestra eficacia en documentos de buena calidad y formato estándar, se identifican retos asociados a la variabilidad de los formatos y la calidad de los documentos, lo que motiva la incorporación de mecanismos de validación y retroalimentación para mejorar la robustez del sistema.



POLLO ANDINO S.A. EN REORGANIZACION
860076820-1
Cra 27 N 36-36
Villavicencio
Tel: 6845028

NOTA CREDITO DE LA FACTURA ELECTRONICA DE VENTA
Número: BB 59621
Fecha 17/05/2025 10:05:11
Generación:
Página: 1 de 1

Cliente:	MARIN QUINTERO JORGE ENRIQUE		Medio de pago	Forma de pago	Fecha vcto.
Contacto:	JORGE MARIN			Contado	18/05/2025
Cedula de	86053663	Código: 86053663	Vendedor:		
Dirección:	CL 5 10A 86		40396462	RODRIGUEZ BETANCOUR DISNARDA	
Ciudad:	Villavicencio		OC.Nro	Docto.Alt.	Factura base / Fecha Fact / Moneda
Teléfono:	3138321047			551376	FVE 54000 / 17/05/2025 / PESOS
Fax:					

Item	Mol	U.M.	Cantidad	Precio unit.	Docto.	Vlr total	IVA %
2130 POLLO SIN VISCERA DE 1300 GR REFRIGERADO PR -551376	02	KL / UND	16,000 / 0,000	$9.400	$0	$150.400	0,00
2150 POLLO SIN VISCERA DE 1500 GR REFRIGERADO PR -551376	02	KL / UND	34,000 / 0,000	$9.400	$0	$319.600	0,00

Total bruto	Descuentos	Sub total	Vlr impuestos	Vlr retención	Total
$470.000	$0	$470.000	$0	$0	$470.000

Valor letras: CUATROCIENTOS SETENTA MIL PESOS M/CTE* * * * * * * * * * * *

Fig. 1. *Estructura de factura en formato PDF (facturaSavicol).*

II. Marco Teórico

A. Python y Streamlit

Python es un lenguaje de alto nivel ampliamente utilizado para el desarrollo de aplicaciones de procesamiento de datos y prototipos rápidos. Streamlit, por su parte, es un framework orientado a la creación de aplicaciones web interactivas de manera sencilla, permitiendo la integración eficiente de componentes de entrada, visualización y procesamiento de datos.

B. Reconocimiento Óptico de Caracteres (OCR)

El OCR es una tecnología que convierte texto contenido en imágenes o documentos escaneados en texto digital editable. En este proyecto, se implementa EasyOCR, una librería basada en PyTorch que soporta múltiples idiomas y utiliza modelos de aprendizaje profundo para mejorar la precisión del reconocimiento [4][5]. Para el procesamiento de archivos PDF, se emplea pdfplumber, una herramienta que permite extraer texto y metadatos de documentos en este formato [6].

C. Expresiones Regulares y Lenguajes Formales

Las expresiones regulares (ER) constituyen una herramienta fundamental para la búsqueda y extracción de patrones dentro de cadenas de texto. Desde el punto de vista teórico, las ER corresponden a lenguajes regulares dentro de la jerarquía de Chomsky, los cuales pueden ser reconocidos por autómatas finitos deterministas (AFD). En este proyecto, las ER permiten modelar los formatos esperados de campos como fechas, identificadores fiscales, montos y descripciones de productos presentes en las facturas.

D. Autómatas y Máquinas Abstractas

La teoría de autómatas provee modelos formales para el reconocimiento de lenguajes. En particular, los autómatas finitos son utilizados en el procesamiento de texto mediante ER. Además, el flujo general de la aplicación puede ser representado como una máquina de Moore o Mealy, donde las salidas (extracción de datos, advertencias, generación de SQL) dependen de los estados y las entradas (archivos, texto, patrones).

III. METODOLOGÍA

El desarrollo del sistema se llevó a cabo en Python, integrando múltiples componentes y bibliotecas para cumplir con los requerimientos funcionales del proyecto. A continuación, se describen las etapas principales del proceso:

A. Diseño de la interfaz web

Se implemento una interfaz interactiva con **Streamlit**. Esta interfaz permite al usuario cargar un archivo de factura en formato imagen (.jpg, .png) o PDF. Proporcionando retroalimentación visual en cada etapa del procesamiento.

B. Extracción de Texto mediante OCR

Utilizando Pytesseract, se realiza la conversión de las imágenes o PDFs en texto plano. Esto transforma la información contenida en los documentos en una cadena de texto susceptible de procesamiento computacional.

C. Procesamiento y extracción de información con Expresiones Regulares

Se diseñaron patrones de ER para identificar y extraer los campos relevantes de la factura: datos del cliente, número y fecha de la factura, subtotal, IVA, total e ítems facturados. Estos patrones corresponden a lenguajes regulares y son evaluados mediante los algoritmos de búsqueda y coincidencia de la librería `re` de Python. Por ejemplo:

1) *Clientes: ID, nombre y dirección.*
2) *Facturas: número de factura, fecha, subtotal, IVA y total.*
3) *Ítems de factura: lista de productos con su descripción, cantidad y precio.*

Las ER se definieron considerando patrones comunes en facturas.

D. Generación automática de script SQL

A partir de los datos extraídos e identificados, el sistema estructura automáticamente sentencias SQL tipo INSERT INTO listas para ser ejecutadas en una base de datos relacional en las tablas correspondientes (clientes, facturas, items_factura).

E. Manejo de Errores y Validación:

Dado que el proceso de extracción está sujeto a posibles fallos (por ejemplo, cuando no se reconoce correctamente un carácter o se omite un campo), se incorporó una validación básica. Si el patrón, no es reconocido, el sistema indica al usuario qué dato no pudo ser extraído, permitiendo corregir manualmente la información faltante o ajustar la factura, asegurando así la integridad de los datos antes de su incorporación a la base de datos relacional.

IV. RESULTADOS Y ANÁLISIS

Esta sección detalla los resultados obtenidos de la aplicación web automatizada para extracción y conversión de imágenes JPG, PNG o PDF. En cada una de sus fases propuestas

A. Evaluación del Proceso de Extracción y Generación

Para validar la efectividad del sistema, se realizaron pruebas con facturas reales en formato imagen (.jpg, .png) y PDF. El flujo consistió en la carga del archivo, la extracción automática de texto mediante OCR o pdfplumber, la identificación de campos relevantes con expresiones regulares y la generación del script SQL correspondiente. En todos los casos, la aplicación presentó al usuario una vista previa del documento, el script SQL generado y advertencias sobre posibles errores de extracción.

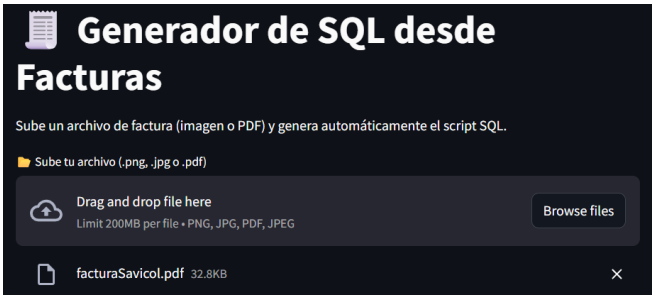


Fig. 2. Interfaz de subir factura en formato png, jpg, jpeg o pdf..

En este caso, a fines de demostración de las fases, se utilizó la facturaSavicol.pdf.

B. Ejemplo de Extracción de Texto y SQL Generado

1) Factura PDF

Texto extraído resumido de Fig 1.

> POLLO ANDINO S.A. EN REORGANIZACION NOTA CREDITO DE LA FACTURA
> 860076820-1 ELECTRONICA DE VENTA
> Cra 27 N 36-36 Número: BB 59621
> Fecha 17/05/2025 10:05:11
> Cliente: MARIN QUINTERO JORGE ENRIQUE ...
> ...
> Total bruto Descuentos Sub total Vlr impuestos Vlr retención Total
> $470.000 $0 $470.000 $0 $0 $470.000
> Valor letras: CUATROCIENTOS SETENTA MIL PESOS M/CTE

2) Fragmento de código de extracción en PDF (parser_pdf.py)

```python
import re

def extraer_datos_pdf(texto):
    datos = {
        "cliente": {},
        "empresa": {},
        "factura": {},
        "items": []
    }
    errores_extraccion = {}

    with open(texto, 'r', encoding='utf-8') as archivo:
        texto = archivo.read()
```

Fig. 3. Listas de datos extraídos de facturas PDF a partir de ER.

```python
    # ================== CLIENTE ====================
    cliente = re.search(r'Cliente:\s*(.*?)\s+Medio de pago', texto)
    cedula = re.search(r'Cedula de\s+(\d{6,15})', texto)
    direccion = re.search(r'Dirección:\s*(.*?)\s+\d', texto)
    telefono = re.search(r'Teléfono:\s*(\d+)', texto)
    codigo = re.search(r'Código:\s*(\d+)', texto)
```

Fig. 4. ER para extraer y validar datos para insertar en lista cliente.

```python
# ================== EMPRESA ====================
nit = re.search(r'LA FACTURA\s*[\r\n]*([0-9]{7,10}-\d)', texto, re.IGNORECASE)
nombre_empresa = re.search(r'^([A-Z ]+ S\.A\.)', texto, re.MULTILINE)
direccion_empresa = re.search(r'Cra\s+\d+\s+N\s+\d+-\d+', texto)
```

Fig. 5. ER para extraer y validar datos para insertar en lista empresa.

```python
# ================== FACTURA ====================
factura_id = re.search(r'Número:\s*BB\s*(\d+)', texto)
fecha = re.search(r'Fecha\s+(\d{2}/\d{2}/\d{4})', texto)
total = re.search(r'Total\s*\$([0-9.,]+)', texto, re.IGNORECASE)
```

Fig. 6. ER parar extraer y validar datos para insertar en lista factura.

Notar que, el texto fue extraído correctamente, manteniendo su sintaxis, léxica y semántica original. Sin embargo, se obtuvo que, el nombre de la empresa este mezclado con la especificidad de la factura. Esto ocurre, porque tiene en cuenta un recorrido lineal de izquierda a derecha, línea por línea. (Revisar Texto extraído).

3) Script SQL Generado

```sql
CREATE TABLE IF NOT EXISTS clientes (
    id INT PRIMARY KEY,
    nombre VARCHAR(100),
    documento VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS empresas (
    nit VARCHAR(20) PRIMARY KEY,
    nombre VARCHAR(100),
    direccion VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS facturas (
    id INT PRIMARY KEY,
    cliente_id INT,
    empresa_nit VARCHAR(20),
    fecha DATE,
    total DECIMAL(12,2),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id),
    FOREIGN KEY (empresa_nit) REFERENCES empresas(nit)
);

CREATE TABLE IF NOT EXISTS items_factura (
    id SERIAL PRIMARY KEY,
    factura_id INT,
    descripción VARCHAR(255),
    cantidad DECIMAL(10,2),
    precio_unitario DECIMAL(12,2),
    FOREIGN KEY (factura_id) REFERENCES facturas(id)
);
```

Fig. 7. Estructura Script SQL para base de datos relacional.

Este script está acompañado de sentencias SQL generadas a partir de un analizador léxico. Observar Fig.3-6.

```sql
INSERT INTO clientes (id, nombre, documento) VALUES (
    9035, 'Marin Quintero Jorge Enrique', '86053663');

INSERT INTO empresas (nombre, direccion, nit) VALUES (
    'Pollo Andino S.A.', 'Cra 27 N 36-36', '860076820-1');

INSERT INTO facturas (id, cliente_id, empresa_nit, fecha, total) VALUES (
    1, 9035, '860076820-1', '17-05-2025', 470000);

INSERT INTO items_factura (factura_id, descripción, cantidad, precio_unitario) VAL
    1, 'Pollo Sin Viscera De 1300 Gr Refrigerado', 16.000, 9400);

INSERT INTO items_factura (factura_id, descripción, cantidad, precio_unitario) VAL
    1, 'Pollo Sin Viscera De 1500 Gr Refrigerado', 34.000, 9400);
```

Fig. 7. Parte 2. Script SQL. Sentencias INSERT INTO.

Fig. 8. Botón para descargar el script de factura SQL.

Al presionar el botón de “Descargar factura_generada.sql” Fig. 8. se obtuvo la estructura mencionada en Fig. 7

V. *DISCUSIÓN*.

A. Precisión y limitaciones

En facturas bien estructuradas y con buena calidad de imagen, la extracción es altamente precisa. Los campos suelen detectarse correctamente, y los datos se insertan en la base de datos sin errores.

La precisión disminuye si la factura tiene baja resolución, presenta manchas o si los campos varían mucho respecto a los patrones esperados por las expresiones regulares. En estos casos, los valores no detectados se marcan como `"NA"`, y se notifica al usuario para su corrección manual.

B. Expresiones Regulares (ER):

Los resultados confirman que las ER empleadas funcionan como un autómata finito determinista (AFD), aceptando solo cadenas que cumplen la gramática regular definida.

VI. CONCLUSIONES

La solución desarrollada demuestra cómo los fundamentos de la teoría de la computación son aplicables en la práctica del procesamiento automático de documentos. El uso de expresiones regulares, gramáticas y tipos de datos es esencial para extraer información de forma precisa y confiable, mientras que la integración de OCR y autómatas en la cadena de procesamiento permite automatizar tareas que tradicionalmente requerían intervención manual. Futuras mejoras pueden incluir el uso de gramáticas más complejas (por ejemplo, para manejar tablas de ítems con estructuras recursivas) y la incorporación de modelos de aprendizaje automático para el reconocimiento de estructuras aún más variadas.

REFERENCIAS

[1] Jurado Málaga, E. (2008). *Teoría de Autómatas y Lenguajes Formales*. Universidad de Extremadura.
[2] Python Software Foundation, “Python Documentation,” [Online]. Available: https://www.python.org/doc/
[3] Streamlit Inc., “Streamlit Documentation,” [Online]. Available: https://docs.streamlit.io/
[4] Jaided AI, “EasyOCR Documentation,” [Online]. Available: https://www.jaided.ai/easyocr/PyTorch Documentation.
[5] [6] J. Vine, “pdfplumber Documentation,” [Online]. Available: https://github.com/jsvine/pdfplumber