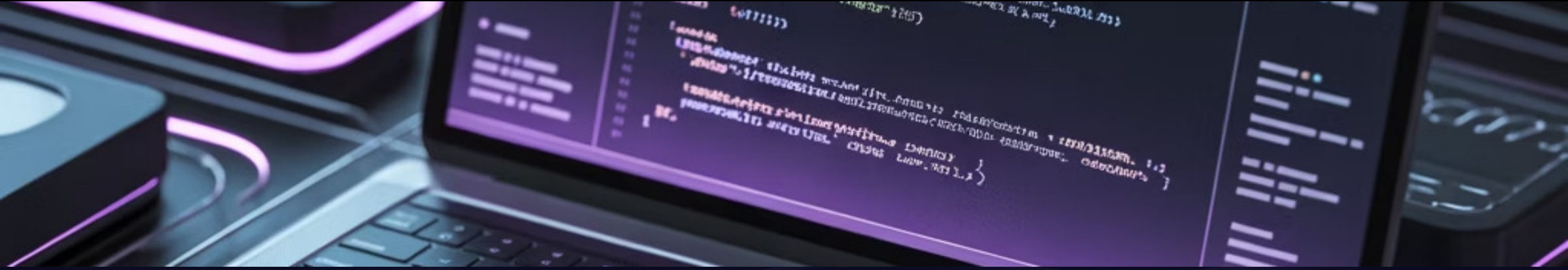
EVIDENCIA PROGRAMACIÓN

Hecho por:

Alvaro Limon Curiel - A01645224

Cristóbal de Jesús Valenzuela Medina - A01646802



Problemática a solucionar

Captura manual poco confiable

Errores por acentos, "ñ", partículas en apellidos, errores de tecleo, y formatos de fecha.

Tiempos elevados y retrabajo

Verificación manual y correcciones repetidas en procesos de alta demanda (KYC, registro de clientes, onboarding).

Necesidad de estandarización

Generar automáticamente un **RFC** con reglas claras y sin diacríticos, manteniendo evidencia y trazabilidad.

Objetivo del proyecto

Tomar una **imagen de pasaporte**, extraer **datos estructurados** (nombres, apellidos, fecha) y producir un **RFC con formato correcto**.

Flujo sencillo de nuestro programa

1. **Subir imagen**
   (Colab) → files.upload()
2. **Leer imagen**
   (OpenCV) → cv2.imread() y mostrarla con plt.imshow()
3. **Leer MRZ**
   (PassportEye) → read_mrz(filename)
4. **Parsear datos**
   names, surname, date_of_birth (YYMMDD)
5. **Separar apellidos**
   paterno, materno (de surname)
6. **Limpiar textos**
   (unicodedata) → sin acentos, ñ→n, sin partículas, MAYÚSCULAS
7. **Generar RFC**
   L1 (paterno), L2 (vocal interna paterno), L3 (materno/X), L4 (nombre/X) + YYMMDD + homoclave **ficticia**
8. **Validar mínimos**
   si faltan nombre/paterno/fecha, no genera RFC
9. **Mostrar salida**
   imprime Nombre(s), Apellidos, Fecha y RFC

Filtros utilizados (librerías)

cv2 (OpenCV)

Uso: lectura de la imagen (cv2.imread), conversión BGR→RGB para mostrar con matplotlib, y base para preprocesos si se necesitan (grises, binarización, resize).

Aporta: **estabilidad** del OCR cuando la imagen no es ideal (ruido, reflejos).

matplotlib.pyplot (plt)

Uso: plt.imshow(...), plt.title("Pasaporte"), plt.axis("off").

Aporta: evidencia visual del documento, útil para **demo y depuración**.

passporteye.read_mrz

Uso: read_mrz(filename) → mrz.to_dict() con names, surname, date_of_birth.

Aporta: **extracción estructurada y confiable**; menor tasa de error que un OCR genérico para estos campos.

random

Uso: homoclave **ficticia** de 3 caracteres: random.choices(string.ascii_uppercase + string.digits, k=3).

Aporta: simula la parte final para **prototipo**. (En productivo se reemplaza por algoritmo SAT.)

Filtros utilizados (librerías)

string

Uso: string.ascii_uppercase, string.digits para componer homoclave y validar caracteres permitidos.

Aporta: control sobre el **alfabeto válido** del token.

unicodedata

Uso: normalize('NFD') + encode/decode para **eliminar diacríticos**.

Aporta: textos **limpios y compatibles** con las reglas del RFC (sin acentos).

google.colab.files

Uso: files.upload() para subir la imagen y probar directamente desde el navegador.

Aporta: flujo rápido de **prototipado** sin construir una UI o backend.

Arquitectura técnica

Implementación de la respuesta

Diseño y Funcionamiento Detallado

Extracción del archivo

Ingesta de imagen (Colab + OpenCV)

El usuario sube la foto/escaneo; validamos que abre y la mostramos.

Lectura MRZ (PassportEye)

Decodificamos la zona legible por máquina (MRZ) conforme **ICAO** para obtener datos confiables.

```python
# === CARGAR IMAGEN ===
uploaded = files.upload()
filename = list(uploaded.keys())[0]
image = cv2.imread(filename)
if image is None:
    raise FileNotFoundError(f"No se pudo abrir la imagen: {filename}")

plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Pasaporte")
plt.axis("off")

# === EXTRAER MRZ ===
mrz = read_mrz(filename)
if not mrz:
    print("No se pudo leer la MRZ.")
    exit()

data = mrz.to_dict()
nombre = data.get("names", "")
apellidos = data.get("surname", "")
fecha_nac = data.get("date_of_birth", "")
```

```python
def limpiar_texto(texto):
    """Convierte a mayúsculas, quita acentos, cambia ñ→n y elimina partículas."""
    if not texto:
        return ""
    texto = texto.lower()
    # Eliminar acentos y ñ
    texto = unicodedata.normalize('NFD', texto).encode('ascii', 'ignore').decode('utf-8')
    texto = texto.replace('ñ', 'n')
    # Eliminar partículas comunes
    particulas = [' de ', ' del ', ' de la ', ' la ', ' los ', ' las ']
    for p in particulas:
        texto = texto.replace(p, ' ')
    return texto.strip().upper()
```

Procesamiento de datos

Parsing de campos

Extraemos names, surname, date_of_birth (YYMMDD) del objeto MRZ.

Normalización lingüística

Convertimos a mayúsculas, quitamos acentos (NFD), mapeamos **ñ→n** y eliminamos partículas ("de", "del", "de la", "la/los/las").

```python
def generar_rfc(paterno, materno, nombre, fecha):
    """Genera un RFC ficticio con formato básico."""
    paterno, materno, nombre = map(limpiar_texto, [paterno, materno, nombre])
    L1 = paterno[0] if paterno else 'X'
    L2 = next((ch for ch in paterno[1:] if ch in 'AEIOU'), 'X')
    L3 = materno[0] if materno else 'X'
    L4 = nombre[0] if nombre else 'X'
    homoclave = ''.join(random.choices(string.ascii_uppercase + string.digits, k=3))
    return f"{L1}{L2}{L3}{L4}{fecha}{homoclave}"
```

Generación de RFC (función generar_rfc)

- **L1**

  Primera letra del **apellido paterno** (o "X" si falta).

- **L2**

  Primera vocal interna del apellido paterno (o "X" si no existe).

- **L3**

  Primera letra del **apellido materno** (o "X" si falta).

- **L4**

  Primera letra del **nombre** (o "X" si falta).

- **Fecha**

  YYMMDD de la MRZ.

- **Homoclave (prototipo)**

  3 caracteres aleatorios [A-Z0-9]. *(En producción: cálculo oficial SAT.)*

Validación y salida

Validación mínima

Si faltan nombre, paterno o fecha, se notifica que no se puede generar el RFC.

Salida y evidencia

Impresión de datos y RFC en consola + visualización de la imagen.

```
# Separar apellidos
partes = apellidos.split()
paterno = partes[0] if len(partes) >= 1 else ""
materno = partes[1] if len(partes) >= 2 else ""

# === MOSTRAR DATOS ===
print(f"\nNombre(s): {nombre or 'No encontrado'}")
print(f"Apellido paterno: {paterno or 'No encontrado'}")
print(f"Apellido materno: {materno or 'No encontrado'}")

# === GENERAR RFC ===
if nombre and paterno and fecha_nac:
    rfc = generar_rfc(paterno, materno, nombre, fecha_nac)
    print(f"\nRFC generado: {rfc}")
else:
    print("No se pudo generar el RFC por falta de datos.")
```

Archivos de entrada y salida

Archivos de entrada

Imagen de pasaporte (ej. pasaporte1.png): foto o escaneo con buena iluminación y resolución.

Archivos de salida

Consola:

RESULTADOS:
NOMBRE(S): Abigail
APELLIDO PATERNO: Medina
APELLIDO MATERNO: Perez

RFC GENERADO: MEPA810117L30

Conclusiones

- **Robustez**

  MRZ ofrece mejor exactitud que OCR genérico para nombres y fecha.

- **Limpieza lingüística confiable**

  unicodedata resuelve acentos, ñ y partículas; el RFC sale **limpio y consistente**.

- **Prototipo funcional**

  RFC con estructura correcta; homoclave **de demo** (aleatoria).

- **Escalable y auditable**

  El pipeline es modular; fácil añadir validaciones, logs y formatos de exportación.