**Visión General del Sistema**

El propósito general de este artefacto de software es analizar un cierto documento (presumiblemente una fotocopia), verificar si corresponde a una identificación oficial (INE), para finalmente recuperar los datos valiosos tales como nombre, dirección domiciliaria, genero, CURP, Clave de elector.

**Justificación General**

Este software fue elaborado con la finalidad de recuperar de manera rápida los datos valiosos de una credencial de elector para así reducir los tiempos de captura de dichos datos.

**Justificación de los lenguajes y librerías de programación.**

Para este artefacto se decidió el uso del lenguaje de programación java en su versión 21, esto debido a la fiabilidad que proporciona el lenguaje, así como su uso regular en la industria de desarrollo de software, lo que permite que pueda ser adaptado a sistemas existentes con una dificultad mínima.

Para la manipulación de Imágenes se decidió por el uso de Opencv, el cual es una librería de visión por computadora de tipo de código abierto, el cual trae implementadas funciones que permiten de manera “relativamente” sencilla la identificación de rectángulos, la identificación de caras, la alteración de imágenes según diversos filtros, así como recortar la imagen propiamente.

Para el proceso de OCR (Optical Character Recognition), se decidió por el uso de la librería [Tesseract](https://www.bing.com/ck/a?!&&p=62e9e673350ec5eaJmltdHM9MTcwODIxNDQwMCZpZ3VpZD0zMmY4NDcyYS0xYjBlLTY4YTQtMzg3NC01NGMyMWFlMDY5ZjUmaW5zaWQ9NTkwMw&ptn=3&ver=2&hsh=3&fclid=32f8472a-1b0e-68a4-3874-54c21ae069f5&psq=Tesereact&u=a1aHR0cHM6Ly9iaW5nLmNvbS9hbGluay9saW5rP3VybD1odHRwcyUzYSUyZiUyZmdpdGh1Yi5jb20lMmZ0ZXNzZXJhY3Qtb2NyJTJmJnNvdXJjZT1zZXJwLXJyJmg9ZFNJSzhjcFprVjVMQnhPVENtaXdPc0E0VUZlckJidTk0NnJBJTJicUdaMXd3JTNkJnA9a2NvZmZjaWFsd2Vic2l0ZQ&ntb=1) debido a su confiabilidad ya que fue diseñada e implementada por Google, así como por su fácil uso y el hecho que por defecto, ya contiene entrenamientos de caracteres predeterminados, por lo que no es estrictamente necesario realizar un entrenamiento desde 0, lo que ahorra bastante tiempo de desarrollo.

Para el manejo de los paquetes anteriormente mencionados se utilizó Maven el cual es un gestor de paquetes, que permite la coordinación y el manejo de dependencias para así evitar el manejo manual de los archivos .jar de las librerías correspondientes.

Como punto final a este apartado cabe resaltar que el proyecto fue elaborado en netbeans, como un proyecto de netbeans, esto se menciona por que puede ser posible que otros editores de código no detecten correctamente la estructura del proyecto.

**Diseño general del sistema**

El sistema está separado en dos grandes secciones. La primera sección está relacionada con la identificación de la INE, así como el procesamiento necesario para la manipulación de la imagen y prepararla para la segunda fase, para términos prácticos se denominará esta fase como “FASE I”.

La fase siguiente corresponde al escaneado de la imagen mediante técnicas de OCR y la manipulación de los datos provenientes del OCR, para así presentar los datos de manera adecuada. Para términos prácticos denominaremos esta fase como “FASE II”.

Contexto: Cuando se realiza el proceso de OCR la función te retorna todo el texto que encuentra sin excepciones, no es posible saber exactamente que pedazo de este texto corresponde al nombre (u otro dato deseado), por lo que se emplearan técnicas de manejo de cadenas (que se explicaran formalmente más adelante) para devolver exactamente los datos que se solicitan de manera estructurada.

**FASE I**

Como se mencionó con anterioridad la fase I es la encargada de la manipulación y análisis de la imagen para determinar si corresponde a una credencial de elector INE, así como su preparación para la fase siguiente. A continuación, se hará mención de las clases que componen esta fase, para así posteriormente explicar método por método su funcionalidad, igualmente se justificaran ciertas decisiones de diseño que a simple vista pueden ser incoherentes.

**ManejoImagenes.java** – Esta clase, como menciona su nombre es encargada de todas las operaciones relacionadas con imágenes, tales como encontrar coordenadas, dibujar rectángulos, detectar caras entre otras operaciones que serán descritas próximamente.

**Coordenadas.java** – Es la clase que sirve para guardar un objeto coordenada, el cual guarda las posiciones correspondientes a las 4 esquinas de una imagen.

**CoordenadasRojas.java** – Es la clase hija de Coordenadas, estas coordenadas hacen referencia explícita, a las coordenadas obtenidas cuando se encuentra un rectángulo con ciertas dimensiones. Se le llaman rojas debido a que cuando se dibujan sobre un archivo temporal el rectángulo encontrado, este es de color rojo.

**Punto.java** – Este objeto es el elemento atómico de una coordenada (el par ordenado X,Y), existe una relación de agregación con la clase Coordenadas.java.

**haarcascade\_frontalface\_default.xml** – Este archivo es el entrenamiento por defecto usado para la detección de caras frontales, puede ser nuevamente descargado en la siguiente liga. <https://github.com/opencv/opencv/blob/4.x/data/haarcascades/haarcascade_frontalface_default.xml>

**ImagenPrimerProceso.png** – Es la imagen de ejemplo de una entrada válida para la presente FASE.

### [ExcepcionesPropias.java](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/ExcepcionesPropias.java) – Una clase que hereda de la clase Exception y contiene un único método que recibe como argumento un mensaje de error, el cual lanza una excepción personalizada (proveniente de la clase padre) que únicamente contiene un mensaje.

### Ahora ya que conocemos todas las clases que conforman esta clase procederemos a explicarlas a profundidad, para la siguiente sección omitiremos a haarcascade\_frontalface\_default.xml, ImagenPrimerProceso.png y [ExcepcionesPropias.java](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/ExcepcionesPropias.java), debido a que no forman parte del algoritmo en general (Se incluye a [ExcepcionesPropias.java](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/ExcepcionesPropias.java), ya que en la descripción general se explicó toda su funcionalidad).

**ManejoImágenes.java**

Esta clase tiene un constructor únicamente inicializando un hash Set de Objetos Coordenada Roja, la razón de que se haya utilizado un hash set es que esta estructura no permite elementos duplicados, para esta funcionalidad de no permitir elementos duplicados se tuvo que sobrescribir los métodos equals() y hashcode() los cuales se encuentran explícitamente en la clase padre Coordenadas.java, de esta manera cualquier clase hija puede utilizar indirectamente estos métodos al entrar en el hash Set.

Todas las funciones únicamente manipulas imágenes en formato PNG.

**EncontrarContornos**

Esta función recibe como argumentos 2 cadenas, la primera cadena corresponde a la ruta absoluta donde se encuentra el archivo de entrada, véase la imagen de ejemplo(ImagenPrimerProceso.png). La segunda cadena corresponde a la ruta absoluta donde el usuario desea guardar la imagen de salida.

La imagen de salida, es una copia de la imagen original, sin embargo, tiene “dibujado” de color rojo cualquier rectángulo con ciertas dimensiones correspondientes al tamaño de una INE, además internamente se guardan como un objeto Coordenadas Rojas, todas aquellas coordenadas correspondientes a los rectángulos encontrados.

Cabe mencionar que cuando se dibujan los rectángulos, no se dibujan exactamente sobre la imagen rectangular, sino se deja cierto margen para garantizar que toda la credencial sea apreciable, debido a que puede estar ligeramente inclinada.

**DividirImagenesPorContorno**

Esta función, recibe por argumento la ruta absoluta correspondiente a la salida de EncontrarContornos, la función toma la imagen deseada, y según las coordenadas que analizo previamente las cuales siguen guardadas en Set<CoordenadasRojas> coordenadasSet, procede a recortar la imagen según los N rectángulos encontradas, devuelve como salida las imágenes recortadas, con la nomenclatura [nombre\_que\_se\_le\_proporciono\_al\_archivo\_de\_salida\_del proceso\_ EncontrarContornos]\_recortada\_[numero\_de\_iteracion\_de\_rectangulo\_recortado].png

Nota: Para las funciones de recorte en esta funcion se utilizaron métodos pertenecientes a la librería OpenCV.

**detectarRostros**

Como su nombre lo indica esta función es la encargada de devolver el número de caras que se encuentren en cierta imagen, recibe como argumento una cadena que corresponde a la ruta absoluta de una imagen que fue recortada previamente por DividirImagenesPorContorno.

Consideraciones Importantes:

* Esta función lanza una excepción si no encuentra suficientes caras
* Requiere que se modifique la ruta absoluta correspondiente al detector de caras el cual es haarcascade\_frontalface\_default.xml, anteriormente se proporcionó el link para descargar el archivo mencionado.

**RecortarImagen**

Como su nombre lo indica procede a recortar una imagen de un archivo especifico. Toma como argumento la ruta absoluta del **archivo original,** un objeto coordenadas y finalmente una ruta absoluta de salida.

Aclaración:

¿Por qué hay 2 funciones de recorte de imágenes?

La razón de que exista 2 funciones de recorte es debido a que cuando se aplicó la función **EncontrarContornos**, para que el proceso fuera exitoso se requirió aplicar ciertos filtros para que pudiera funcionar correctamente el procedimiento, sin embargo con cada proceso que le realizamos a la imagen, su calidad disminuyó, lo que provoca que los procesos de la FASE II no se puedan realizar adecuadamente, anudado a eso durante el desarrollo se detectó que la función incorporada de recorte de OpenCv, bajaba demasiado la calidad de la imagen. Aunque deterioraba la imagen se conservó la funcio ya que su ejecución es muy rápida por lo es idóneo cuando se requiere analizar muchas hojas.

El proceso actual **RecortarImagen** utiliza una librería perteneciente a Apache el cual, aunque tarda ligeramente más en ejecutarse mantiene fiel la calidad de la imagen original que recibe como argumento. Así como devuelve la imagen en la ruta absoluta que tú le indiques, lo que permite conectarla con otro proceso.

**PDFtoPNGConverter**

Como su nombre lo indica, convierte archivos PDF a imágenes en formato PNG, recibe como argumento la ruta absoluta de un archivo PDF, y devuelve según el directorio indicado en el argumento, las N páginas del documento convertidos en imágenes PNG, cabe aclarar que las imágenes PNG se devuelven por separado (individuales, una por cada página). Esta función no se utilizó en el código final, sin embargo, se dejó en caso de que los documentos existentes requieran ser transformados de formato.

# **Coordenadas.java**

Coordenadas es la clase la cual guarda objetos de tipo coordenadas, una coordenada está compuesta por 4 elementos.

* Punto esquinaSuperiorIzquierda
* Punto esquinaSuperiorDerecha
* Punto esquinaInferiorIzquierda
* Punto esquinaInferiorDerecha

La clase contiene los getters y setters correspondientes a los atributos anteriormente mencionados.

Contiene el método sobrescrito toString() por si se requiere imprimir en algún momento el objeto.

También contiene el método sobrescrito equals() pero en este caso se ha modificado ligeramente para tener una tolerancia de +/- 3 esto significa que si una medida tiene de largo 10 y la otra 13, las considerara iguales. Esto debido a que durante el proceso de recorte existen falsos positivos o duplicaciones de coordenadas, con este equals modificado garantizamos que no haya errores de ese tipo.

También contiene el método sobrescrito hashcode(), que se utiliza para representar de manera única a un objeto, inclusive si tienen exactamente los mismos atributos.

**CoordenadasRojas.java**

Es la clase hijo coordenadas. Contiene 2 constructores validos uno el cual es el constructor vacion, que inicializa la corrdenada en 0. El otro es el constructor que recibe 4 argumentos correspondientes a:

* Punto esquinaSuperiorIzquierda
* Punto esquinaSuperiorDerecha
* Punto esquinaInferiorIzquierda
* Punto esquinaInferiorDerecha

Para este constructor únicamente se llama al constructor padre con los argumentos antes mencionados.

Contiene el método setCoordenadas(), el cual asigna atributos de la coordenada a una coordenada, principalmente se utiliza para re escribir coordenadas creadas por el constructor vacío, utiliza setters provenientes de la clase padre.

Contiene un método toString(), que esta sobrescrito de la clase padre, con la única diferencia de que imprime una leyenda mencionando que las coordenadas son rojas.

Aclaración: ¿Por qué se hace la distinción de color de coordenadas?

El sistema de detección de INE tiene amplio margen de mejora, supongamos que en un futuro quisiera recortar únicamente la cara del ciudadano que aparece en la INE, si todo se manejara con coordenadas sería un completo caos saber, a que elemento hacen referencia cada coordenada, por lo tanto, se realiza esta distinción, quien sabe en un futuro las coordenadas azules podrían hacer acto de presencia en el programa informático.

**Punto.java**

Únicamente contiene los atributos X, Y correspondientes a una posición en el plano cartesiano, la clase contiene getters y setters de los respectivos atributos, así como la sobre escritura del método toString(), para imprimir de manera elegante la coordenada deseada.

FASE II

Como se mencionó con anterioridad la fase II es la encargada de la realización del OCR formalmente, así como la manipulación de las cadenas para conseguir el contenido deseado. Esta fase contiene las siguientes clases:

ClaveDeElector.java – Clase de apoyo que sirve para manipular los caracteres individuales de una clave de elector de manera mucho más clara y transparente.

ImagenSegundoProceso.png – Imagen de ejemplo de la entrada idónea para esta fase.

ManejoCadenas.java – La presente clase contiene todos los métodos necesarios para extraer los datos valiosos que se encontraron durante el proceso de OCR

OCR.java – La clase contiene métodos puramente orientados al reconocimiento de caracteres.

PersonalNE.java – Clase que genera objetos “ciudadano” donde se guardan los datos encontrados en la INE, por si es necesario realizar más operaciones con los datos encontrados.

### [spa.traineddata](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/spa.traineddata) - Archivo que contiene el entrenamiento necesario para el reconocimiento de caracteres. Se puede obtener del siguiente link: <https://github.com/tesseract-ocr/tessdata/blob/main/spa.traineddata>

### Ahora ya que conocemos todas las clases que conforman esta clase procederemos a explicarlas a profundidad, para la siguiente sección omitiremos a [spa.traineddata](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/spa.traineddata) y ImagenSegundoProceso.png, ya que no forman parte del algoritmo del programa.

**OCR.java**

La clase OCR, contiene un constructor vacío, el cual es el encargado de instanciar a la clase Tesseract y proveer de las pre configuraciones necesarias para el correcto funcionamiento del reconocimiento de caracteres.

Nota: Es necesario modificar el testdataPath según la ruta absoluta en la que se encuentre en tu equipo de cómputo.

Cuenta con un método denominado “MetodoAlternativo”, el cual llama al método doOCRWithBufferedImage el cual es el que realiza formalmente el proceso de OCR, la razón de esto es que el método doOCRWithBufferedImage es privado, por lo que requiere de un método que sirva “como puerta de entrada” para poder ejecutarlo.

**PersonaINE.java**

Esta clase sirve como apoyo para poder guardar los elementos obtenidos de los procesos que se encuentran en la clase de ManejoCadenas.java, al ser una clase únicamente para guardar datos cuenta con los siguientes atributos, así como su respectiva función toString(), para que se imprima de forma elegante.

* primerApellido;
* segundoApellido;
* nombre;
* domicilio;
* claveElector;
* curp;
* fechaNacimiento;
* numeroEntidadFederativa;
* sexo;

**ClaveDeElector.java**

La clave es una cadena compuesta de 18 caracteres los cuales cada carácter representa ciertos datos, por ejemplo:

1. **Primer y segundo dígito**: Representan las dos consonantes iniciales del primer apellido.
2. **Tercer y cuarto dígito**: Corresponden a las dos consonantes iniciales del segundo apellido.
3. **Quinto y sexto dígito**: Indican las dos consonantes iniciales del nombre del elector.
4. **Del séptimo al duodécimo dígito**: Representan la fecha de nacimiento (dos dígitos para el año, dos para el mes y dos para el día).
5. **Décimo tercer y décimo cuarto dígito**: Son el número de la entidad federativa donde nació el titular.
6. **Décimo quinto dígito**: Es una letra que identifica el género (por ejemplo, “M” para masculino o “F” para femenino).
7. **Décimo sexto, décimo séptimo y décimo octavo dígito**: Forman la homoclave, compuesta por tres dígitos asignados aleatoriamente por el INE.

Más adelante estos datos serán vitales para el proceso de detección de nombre dirección, genero, etc.

¿Por qué es una clase separada, no hubieras hecho agregación de la clase PersonaINE?.

Hubiera sido completamente valido sin embargo no proporcionaría suficiente claridad al desarrollador, por ejemplo caracteres[5] no sabría el desarrollador que este carácter está asociado a segundaLetraNombre.

La clase contiene un constructor que recibe como parámetro una cadena correspondiente a la clave de elector. Luego se procede a recorrer carácter por carácter la cadena para así asignar los elementos al arreglo correspondiente de tal manera que, cada carácter tiene asociado un getter que menciona de manera explícita de que se trata.

primeraLetraPrimerApellido = caracteres[0];

public char getPrimeraLetraPrimerApellido() {

return primeraLetraPrimerApellido;

}

Aunque parece un poco tedioso a largo plazo esta pequeña clase nos será de gran utilidad para todos los procesos siguientes.

# **ManejoCadenas.java**

Esta sin dudas es una de las clases más importantes de nuestro programa, ya que es la que nos va a devolver los datos concretos que requerimos.

Posee un constructor que recibe 2 parametros, la cadenaOriginal la cual es el producto de las funciones de OCR y el objeto PersonaINE ciudadano el cual va a servir para guardar los datos recuperados.

En el constructor se va a inicializar una LinkedList que va a ser el encargado de guardar todas las sub cadenas pertenecientes a la cadenaOriginal, así como también se va a inicializar un ArrayList donde se guardan ciertos Datos Vitales los cuales son la sub cadena correspondiente a la CURP y la sub cadena correspondiente a la clave de Elector.

¿Por qué utilizas LinkedList y un ArrayList?

Se utiliza ArrayList para los datos Vitales debido a que, aunque sabemos que en la INE solo existen 2 sub cadenas que corresponden a la CURP y la clave de Elector, puede darse el caso (fatídico) donde se encuentre otra cadena de 18 caracteres, si fuese un Array simple, correríamos con el riesgo que se dispare una excepción por desbordamiento del Arreglo, más adelante en el método EncontrarClaveElector toma más sentido esta decisión.

Se utiliza Linked List ya que hay una sección de código donde si se encuentra una sub cadena que cumple ciertas condiciones, la guardas en un atributo de un Objeto INE, y procedes a eliminarlo. LinkedList nos ofrece la habilidad de eliminar elementos mientras se itera sobre él, sin peligro de disparar una excepción.

Aclarado este punto procedemos a explicar la funcionalidad de cada método.

**LimpiarCadenaValoresBasura** – Recibe como argumento una cadena (Presumiblemente la cadena producto del proceso de OCR) y procede a eliminar todos aquellos valores que no son letras mayúsculas ni números.

**VerificarINE** – Itera sobre la cadena para encontrar ciertas palabras clave que corresponden a una INE, tales como VOTAR, ELECTORAL, CREDENCIAL, etc; dispara una excepción si no encuentra ninguna de estas palabras;

**GuardarSubcadenasConMasDeUnDigito** – Guarda únicamente las subcadenas las cuales contengan más de un carácter.

**BuscarCURPyClaveElector** – Guarda en el ArrayList datosVitales, aquellas subcadenas las cuales contengan exactamente 18 caracteres.

**EncontrarClaveElector** – Dispara una excepción si ArrayList datosVitales tiene un valor de elementos diferente a 2, utiliza una expresión regular para determinar si los primeros 6 caracteres de uno de los elementos de datosVitales, corresponden a letras mayúscula en caso que sea cierto asigna dicho valor a un nuevo Objeto de tipo ClaveDeElector, análogamente salva tanto la CURP y la clave de elector en la instancia ciudadano previamente inicializada en el constructor.

**EliminarElementosBasura** – Es la función que itera sobre los elementos LinkedList<String> arregloElementosOCR, con tal de eliminar aquellos elementos que contengan ciertas cadenas predeterminadas o aquellas que posean 4 números seguidos. Esto con tal de eliminar aquellas subcadenas que no agreguen valor.

**EncontrarNombres** – Aquí es donde hace presencia los atributos de la clase ClaveDeElector. Se iterará sobre todos los elementos del arreglo, por ejemplo:

Si se encuentra con una sub cadena que contenga el carácter de elector.getPrimeraLetraPrimerApellido() y elector.getSegundaLetraPrimerApellido(). Consideraremos que esa sub cadena corresponde al primer apellido. La misma lógica se aplica para el segundo apellido y el nombre.

**EncontrarSexo** – Recupera el char asociado al texto de la instancia Sexo del Objeto Clave de elector, finalmente se agregan al objeto ciudadano(PersonalINE).

**EncontrarEntidadFederativa** – Recupera los chars asociados a la entidad federativa correspondientes al Objeto de la clave de elector, finalmente se agregan al objeto ciudadano(PersonalINE).

**EncontrarFechaNacimiento** – Del objeto generado de la clave de elector, devuelve los valores asociados al dia, mes y año de nacimiento, y los formatea para que sean una fecha completa en formato dd-MM-yyyy, finalmente se agregan al objeto ciudadano(PersonalINE).

**EncontrarDomicilio** – Hallar el domicilio en una INE no es tarea sencilla, existen infinitas combinaciones de dirección, por lo que es imposible generalizarlos, por lo tanto, procederemos a explicar detalladamente este metodo.

Nota: Este es el último método que se debe ejecutar antes de terminar toda la fase.

* Sabemos que, en una INE, la cadena del domicilio se encuentra inmediatamente a la derecha de la palabra DOMICILIO, cuando se encuentre la pablara que contenga la sub cadena DOMI, se guardara el índice correspondiente denominado posicionDomicilio
* Sabemos que el domicilio de una INE siempre termina con tres letras mayúsculas, por ejmplo: YUC, también vamos a buscar el índice de la subcadena que únicamente contenga 3 letras mayúsculas, denominado posicionEstado.
* Ahora dentro de la LinkedList vamos a recuperar todos los elelemntos que se encuentren entre el índice de posicionDomicilio y el índice de posicionEstado.
* Removeremos del Linked List todos los elementos que encontramos.
* Todos los elementos que recuperamos los guardaremos en el campo correspondiente del objeto de PersonaINE.

**DevolverElementosNoIdentificados** – Devuelve todos los elementos sobrantes, aquellas sub cadenas que no pudieron ser analizadas o no pertenecen a ningún atributo.

**dividirPorSaltosDeLinea** – Divide una cadena mediante sus saltos de línea y lo guarda en un arreglo de Strings

**verificarCuatroNumeros** – Mediante expresiones regulares verifica si una sub cadena contiene 4 numeros seguidos.

**RepararCURP** – El OCR no es perfecto algunas veces detecta los “O” como 0, lo cual es erróneo, sabemos que hay ciertos caracteres dentro del CURP que corresponden a números, entonces realiza el cambio únicamente si están en los dígitos correspondientes a los números (Fecha de Nacimiento).

**RepararClaveElector** - El OCR no es perfecto algunas veces detecta los “O” como 0, lo cual es erróneo, sabemos que hay ciertos caracteres dentro de la Clave de Elector que corresponden a números, entonces realiza el cambio únicamente si están en los dígitos correspondientes a los números (Fecha de Nacimiento).

**unirElementos** – Concatena los elementos de un LinkedList<String> para devolver una única cadena.

**Main.java**

Es el archivo maestro que desencadena tanto las ETAPAS I y II. En el respectivo orden para que realicen sus funciones adecuadas, el orden en que se llaman las funciones no se debe alterar ya que provocaría errores inesperados.

**Limitaciones/Restricciones:**

* Para empezar actualmente únicamente soporta archivos PNG, aunque dentro del código se encuentra disponible la función de convertir un documento PDF a múltiples imágenes PNG, no está implementada la lógica de iterar sobre cada una de las imágenes provenientes del archivo PDF y realizar el proceso de detección de INE, por lo que actualmente únicamente soporta una página a la vez.
* Actualmente el sistema no realiza una eliminación de los archivos temporales generados del proceso, suponiendo que se vuelve a ejecutar el proceso las imágenes temporales únicamente se sobrescribirían, lo que podría ocasionar problemas si se desearan conservar ciertos archivos.
* El proceso de OCR tiende a fallar, esto se puede deber a múltiples razones: Durante el proceso de implementación se usó un entrenamiento predeterminado para el reconocimiento de caracteres, actualmente se desconoce qué tan intensivo fue ese entrenamiento así como se cuestiona la calidad del mismo, sin embargo se usó para fines prácticos, ya que es capaz de detectar texto en casos simples, ( Con casos simples nos referimos que la imagen se encuentre completamente alineada, que sea un escaneado original, etc.). Para mejoras futuras se recomienda la realización de un entrenamiento personalizado.
* Por las razones anteriormente mencionadas es que el proceso de detección de INE, así como el proceso de OCR se encuentran separados, esto debido a que no se puede garantizar que una hoja escaneada que contenga ambas caras de la INE pueda ser detectada adecuadamente sus datos que contiene.
* Durante el proceso de OCR el domicilio no puede ser detectado con completa exactitud, para este proceso se realizó una generalización de los casos que podrían contener la dirección de una persona, sin embargo, no se puede garantizar que la dirección sea adecuada.
* Análogamente tampoco se puede garantizar que el nombre pueda ser recuperado por completo, como se mencionó anteriormente conocemos que ciertos caracteres correspondientes a la clave de elector consisten en caracteres que pertenecen a los nombres, sin embargo, después de múltiples pruebas se comprobó que en el caso de segundos nombres dichos no se pueden recuperar. De hecho, al finalizar el proceso queda como una cadena sin analizar, a continuación, un ejemplo:

Nombre Completo: PABLO ERNESTO BAEZA LARA

Nombre Recuperado: PABLO BAEZA LARA

Cadena sin recuperar: ERNESTO

* Existen ciertos datos que no se pueden recuperar de la INE, tales como el año de registro, la sección electoral, así como la vigencia de la presente credencial.
* El sistema únicamente analiza la cara frontal de la INE ósea la que contiene las 2 fotografías del ciudadano.
* El programa no es capaz de detectar si una INE es verídica, ósea le es incapaz de detectar si es una falsificación.
* Para el proceso de OCR, el programa falla si la INE se encuentra de cabeza, ósea boca abajo.
* Para el proceso de detección de imágenes, específicamente en la detección de rectángulos si la INE se encuentra con una imagen con rotación mayor a 20 grados centígrados, la detección del rectángulo puede ser no del todo exitosa.
* Para el proceso inicial de detección de INEs (FASE 1), únicamente se probó con imágenes del tamaño de una hoja editadas de tal manera que contuvieran un recorte de imagen correspondiente a las 2 caras de la INE, se puntualiza que no se probó con fotocopias reales.
* Para el proceso de OCR el cual fue minuciosamente probado se llegó a la conclusión que únicamente funciona con una entrada que pertenezca a un escaneado de la cara frontal de la INE original, ósea directo del escáner, cuando se intentó con imágenes generadas por la FASE I, así como por fotocopias, los resultados fueron nada aceptables, se vuelve a puntualizar que la razón de esto puede ser el archivo de entrenamiento proporcionado, el cual se vuelve a mencionar que es un entrenamiento por defecto recuperado de internet, para entornos de producción se recomienda un entrenamiento a la medida.
* Se puntualiza de nueva cuenta que el presente software no está preparado para entornos de producción reales, ya que no ofrece la suficiente confiabilidad tanto en la recuperación de datos como en la detección de las imágenes, únicamente fue posible probarla con 3 credenciales de elector.
* Actualmente el software no admite entrada de datos, la manera en que se le “asigna” la ruta absoluta de la imagen deseada es mediante modificación directa del código lo cual es inaceptable para el usuario (No vamos a poner al usuario a modificar código).
* Para reafirmar el punto anterior el presente programa tampoco cuenta con una interfaz gráfica de usuario (GUI) implementada.
* El programa no es del todo portable, aunque el proyecto se encuentra con un manejador de dependencias denominado “MAVEN”, esto no evita que ciertos archivos tales como [spa.traineddata](https://github.com/PabloBaeza56/OCR-DISENIO/blob/main/src/main/java/main/spa.traineddata), así como, haarcascade\_frontalface\_default.xml. Se tengan que reasignar sus rutas absolutas en el equipo de cómputo de destino, la modificación se requiere hacer directamente en código, lo que es inaceptable para un usuario final.

**Requisitos NO Funcionales (Atributos de calidad):**

Actualmente el sistema se encuentra en fase de Prueba de Concepto, como un producto de alta calidad generado por un Ingeniero de software se espera que se cumplan la gran mayoría de requisitos NO funcionales (Básicos) tales como seguridad, mantenibilidad, portabilidad, etc. Sin embargo, a continuación, se mencionarán aquellos requisitos no funcionales con los que actualmente cuenta el sistema.

* El sistema de software cuenta con métodos atómicos capaces de ser reutilizados en proyectos análogos al actual.
* El sistema cuenta con métodos modulares los cuales respetan el principio de responsabilidad única.

**Requisitos Funcionales:**

**Etapa de detección de Imágenes de credenciales de Elector(INE):**

* El sistema es capaz de recibir una imagen en formato PNG como entrada, detectar si cuenta con formas rectangulares de ciertas dimensiones correspondientes a una identificación oficial y encerrar en un rectángulo de color rojo dicha imagen rectangular encontrada.
* El sistema es capaz de encontrar en una imagen en formato PNG, la N cantidad de rectángulos rojos previamente encontrados, para posteriormente recortar individualmente cada rectángulo de color rojo y guardarlo cada uno de los rectángulos encontrados como un archivo nuevo en formato PNG.
* El sistema es capaz de dada una imagen en formato PNG, previamente recortada a la forma de un rectángulo, detectar la N cantidad de caras humanas que dicha contenga.
* El sistema, de manera mucho más precisa y sin perder la calidad de una imagen original, es capaz de recortar cierta imagen con coordenadas previamente encontradas (Primer Requisito Funcional).

**Etapa de procesamiento óptico de caracteres:**

* El sistema mediante métodos predeterminados de reconocimiento óptico de caracteres es capaz de analizar una imagen producto de la primera fase (La cual presumiblemente es la cara frontal de una INE), para así realizar el análisis correspondiente el cual devuelva una cadena perteneciente a los caracteres encontrados.
* El sistema es capaz dado una cadena (Resultante del requisito funcional inmediato anterior) dividir dicha cadena en un arreglo de sub cadenas, la división se realizar mediante espacios en blanco.
* Dada la cadena resultante del requisito funcional inmediato anterior, conserva toda sub cadena que se encuentre en letras mayúsculas, o sea un numero natural. Todas las demás sub cadenas que no cumplan con alguna de las 2 características anteriormente mencionadas serán eliminadas de la cadena final de salida.
* El sistema es capaz de determinar cuál de las sub cadenas (producto del requisito funcional inmediato anterior) contienen exactamente 18 caracteres, de esas sub cadenas con tales características, el sistema detecta si esta pertenece a una CURP o a una clave de elector.
* El sistema manipulando los caracteres individuales de la clave de elector, iterando sobre el arreglo de subcadenas, encontrara las subcadenas que contengan los caracteres que pertenezcan a la subcadena según el criterio de búsqueda, por ejemplo:

La clave es una cadena compuesta de 18 caracteres los cuales cada carácter representa ciertos datos:

* **Primer y segundo dígito**: Representan las dos consonantes iniciales del primer apellido.

Para encontrar la subcadena nombre iteraremos hasta encontrar una subcadena que contengan el primer y el segundo digito de la clave de elector, con esto determinaremos que dicha subcadena pertenece al primer apellido, la lógica mencionada se repite para el segundo apellido y el primer nombre, con los caracteres de la clave de elector que le corresponden.

* El sistema manipulando la cadena de la clave de elector correspondiente recupera el carácter que corresponde al género del ciudadano.
* El sistema manipulando la cadena de la clave de elector correspondiente recupera los caracteres asociados a la fecha de nacimiento y devuelve una fecha en formato dia/mes/año.
* El sistema manipulando la cadena de la clave de elector correspondiente recupera los caracteres que corresponden a la entidad federativa del ciudadano y los concatena
* El sistema manipulando las subcadenas restante recupera los elementos que se encuentre entre alguna subcadena que contenga la cadena “DOMI” y alguna subcadena que contenga 3 letras mayúsculas seguidas. (Las cuales representan el estado de residencia del ciudadano.