HERRAMIENTA PARA EL DESARROLLO DE ELEMENTOS ESTÁTICOS EN HIGH PERFORMANCE HMI PARA LA EMPRESA INFODESIGN COLOMBIA

Camilo Andrés Echeverri Mondragón

Trabajo de grado en Automática industrial

Modalidad: Práctica profesional

Director: Msc. Oscar Amaury Rojas Alvarado

Asesor de la empresa: Ing. Henry Erik Pérez Valdeblanquez

Universidad del Cauca

Facultad de ingeniería Electrónica y telecomunicaciones

Programa de Ingeniería en automática industrial

Popayán, junio 2023

Camilo Andrés Echeverri Mondragón

HERRAMIENTA PARA EL DESARROLLO DE ELEMENTOS ESTÁTICOS EN HIGH PERFORMANCE HMI PARA LA EMPRESA INFODESIGN COLOMBIA

Informe presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del título de

Ingeniero en:

Automática Industrial

Director: Msc. Oscar Amaury Rojas Alvarado

Popayán, Cauca

2023

Nota de Aceptación: _____

_____

_____

_____

_____

_____

Firma del director _____

Firma del jurado _____

Firma del jurado _____

Popayán, junio de 2023.

Dedicatoria

A mis padres quienes con todo su esfuerzo, amor y dedicación no dudaron en ningún momento otorgarme la oportunidad tan valiosa de realizar mis estudios, oportunidad que ellos no tuvieron y quizás en algún momento la anhelaron, ellos quienes en los altos y bajos de este camino siempre me apoyaron con palabras de aliento e impulsaron cada paso para alcanzar tan anhelada meta.

Finalmente, y no menos importante a todas las personas que de alguna manera u otra han contribuido para lograr llegar a este punto.

Agradecimientos

Agradezco a mi familia que con su apoyo incondicional me ayudaran alcanzar y muchas otras metas, a mis compañeros, pero por encima de todo amigos Carlos y Laura con quien pase incontables horas entre estudio, risas y preocupaciones, fueron el mejor equipo de trabajo, no elegiría otro si tuviera la opción y espero que sigan presentes en mi vida.

A Paula, mi novia, quien vio todo el desarrollo de este trabajo, todas las derrotas y victorias, los momentos de inspiración y quizás más importante en los momentos de desolación donde me apoyo y siempre me animaba a seguir.

Al Ingeniero Oscar, mi director de trabajo de grado siempre con una excelente disposición para resolver mis dudas y quien en varias ocasiones me tuvo en cuenta para ciertas oportunidades.

Tabla de contenido

Lista de figuras

LISTA DE TABLAS

Introducción

La inteligencia artificial (IA) ha emergido como un campo apasionante y transformador en la intersección de la ciencia de la computación, las matemáticas y la ingeniería. En su esencia, la IA se esfuerza por dotar a las máquinas con la capacidad de razonar, aprender y tomar decisiones de manera similar a como lo hacen los seres humanos. Los avances en el campo de la IA especialmente en el aprendizaje profundo han desencadenado un rápido cambio de paradigma en la industria. Empresas de diversos sectores han encontrado aplicaciones innovadoras de la IA para mejorar la eficiencia, la personalización y la toma de decisiones, desde la automatización de tareas repetitivas hasta la creación de sistemas de detección de imágenes altamente precisas. [1][11]

La IA analiza imágenes con gran precisión, ofreciendo información valiosa en aplicaciones industriales diversas. En manufactura, detecta defectos en productos mejorando la calidad y costos. En logística, sistemas de visión permiten realizar un seguimiento eficiente, optimizando la cadena de suministro y gestión de inventario. Además, en atención médica, la detección de imágenes posibilita diagnósticos más precisos y rápidos. Estos avances evidencian la creciente relevancia de la IA en industrias y su capacidad para transformar procesos desafiantes. [2][4][12]

A raíz de todas estas aplicaciones en la industria, surge la idea en colaboración con la empresa Infodesign de integrar la inteligencia artificial en el proceso de desarrollo. Esto se debe al considerable volumen de trabajo en el área de diseño, donde se reciben miles de Diagramas de Instrumentación y Proceso (P&ID), los cuales esperan ser transformados en Interfaces Humano-Máquina (HMIs) que cumplan con la normativa de alto rendimiento para su desarrollo.

Se implementó la metodología ágil Scrum para llevar a cabo el desarrollo del proyecto. Se desglosó en 12 sprints en un lapso de 3 meses, lo que posibilitó realizar un seguimiento preciso del progreso, así como la gestión eficaz de obstáculos y contingencias que surgieron a lo largo del proceso.

Capítulo 1

1. Contextualización del Problema

1.1. Planteamiento del problema

1.1.1. Descripción del problema.

Los diagramas P&ID son representaciones gráficas, que se utilizan para mostrar el funcionamiento de un proceso industrial y también la forma en la cual se interconectan los distintos componentes involucrados. Estos diagramas son muy detallados, además, contienen información acerca de los instrumentos, las tuberías, los equipos y los componentes de control que se utilizan en el proceso.[13]

Mientras que un HMI es una interfaz que permite a los operadores controlar y monitorear el proceso de manera efectiva. [14][15]

El desafío en este proyecto radica en implementar una red neuronal para identificar ciertos objetos de interés y posteriormente crear una plantilla para el diseño de HMIs a partir de diagramas P&ID, tomando en cuenta la complejidad por la cantidad de información que contienen estos diagramas. La inteligencia artificial debe ser capaz de analizar el diagrama P&ID y extraer la información necesaria para diseñar dicha plantilla.

Para lograr esto, se necesitan algoritmos de procesamiento de imágenes y de reconocimiento de patrones que puedan identificar los distintos componentes presentes en el diagrama. Una vez que se ha diseñado la plantilla HMI, se debe realizar un proceso extra de adecuamiento por parte de los diseñadores para lograr un HMI totalmente funcional que cumpla con todos requerimientos para el proceso que ha sido diseñada.

1.1.2. Justificación del problema

InfoDesign toma como principal actividad económica el desarrollo de herramientas y sistemas informáticos para el sector Oil & Gas, entre los servicios ofertados se encuentran el diseño e implementación de interfaces humano-máquina de alto desempeño (High performance HMI), desarrollo de software para el manejo de información en tiempo real, entre otras. En la industria Oil & Gas es de vital importancia la correcta representación de los elementos, conductos, tuberías y lazos de control al momento de realizar la implementación en las interfaces humano-máquina debido a que en este tipo de procesos los pequeños detalles marcan la calidad y un correcto producto final.

El desarrollo de HMI es un proceso complejo que requiere de habilidades especializadas y conocimientos técnicos avanzados, ya que involucra diversas etapas como levantamiento de la información, contratación, diseño, pruebas, concertación con el cliente, entrenamiento de personal y puesta en funcionamiento. Por otra parte, el diseño de los despliegues debe estar regido por la norma ANSI ISA 101, la cual tiene como principal función permitir que los operadores logren observar sin esfuerzo o confusión los cambios en las variables del proceso representado y de esta manera evitar accidentes o perdidas en la producción, para lograr este objetivo la normativa define ciertos parámetros como: color, tamaño de fuente, forma de equipos, conexiones, grosor de líneas, entre otros. Además, la norma también determina diferentes niveles para las pantallas siendo los más relevantes al momento de diseñar el nivel dos “High Level Process Display” o “Schematic Overview” y el nivel 3 “System detail displays”, estas pantallas son normalmente elaboradas como una representación de los diagramas P&ID, que se utilizan para representar el proceso industrial, estos suelen ser muy detallados y contener una gran cantidad de información, lo que dificulta el diseño de una interfaz efectiva, debido a esto para una correcta interpretación de los P&ID es necesario contar con personal especializado que pueda identificar y filtrar los elementos necesarios para una representación adecuada en un HMI.

Para abordar esta necesidad y simplificar el proceso de diseño, se plantea la creación de una aplicación utilizando el lenguaje de programación Python. Esta herramienta aprovechará la inteligencia artificial (IA) para automatizar la identificación y filtrado de elementos estáticos presentes en los P&ID. Una vez completado este análisis, se generará un dibujo vectorial que reflejará los diversos componentes delineados en los diagramas. Esta innovación optimizará significativamente el proceso de diseño, eliminando la necesidad de llevar a cabo tareas repetitivas de manera manual.

1.1.3. Objetivos del estudio

Objetivo general

Implementar un prototipo para la creación de elementos estáticos no parametrizables en interfaces humano-máquina de alto rendimiento basados en la representación gráfica de los diagramas de tubería e instrumentación.

Objetivos específicos

- Generar una inteligencia artificial capaz de identificar elementos estáticos, tales como líneas de proceso, control y equipos para ser indexados en una base de datos para su posterior tratamiento.
- Diseñar un aplicativo inteligente capaz de realizar una librería el dibujo vectorial de manera autónoma de los elementos identificados.
- Evaluar la confiabilidad de la herramienta comparando sus resultados con los que puede generar un diseñador de HMI.

Capítulo 2

2. Marco teórico

2.1. Marco conceptual y teórico

En este capítulo se abordan algunos conceptos necesarios con el fin de ayudar a comprender de manera adecuada el presente trabajo.

Los principales conceptos y planteamientos teóricos utilizados son:

2.1.1. Diagrama P & ID (Piping and Instrumentation Diagram)

Es una representación gráfica de un proceso industrial que muestra el flujo de los fluidos y los equipos involucrados en el proceso, junto con los instrumentos de medición y control. (Ludwig, 2009).

2.1.2. Inteligencia artificial (IA)

Es la capacidad de las máquinas de imitar la inteligencia humana y realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de patrones, el aprendizaje automático y la toma de decisiones. (Russell & Norvig, 2010).

2.1.3. Aprendizaje profundo (Deep Learning)

Es un subcampo del aprendizaje automático que se basa en redes neuronales artificiales para aprender patrones y relaciones complejas en los datos de entrada. (Goodfellow et al., 2016).

2.1.4. Redes neuronales convolucionales (Convolutional Neural Networks, CNN)

Son un tipo de red neuronal artificial que se utiliza para procesamiento de imágenes y otros datos que tienen una estructura espacial. Las CNN utilizan filtros convolucionales para extraer características de las imágenes de entrada. (LeCun et al., 1998).

2.1.5. YOLO (You Only Look Once)

Es un algoritmo de detección de objetos en tiempo real que utiliza una red neuronal convolucional para identificar objetos en una imagen. YOLO divide la imagen de entrada en una cuadrícula y predice la clase y la posición de los objetos en cada celda de la cuadrícula. (Redmon et al., 2016).

2.1.6. OCR (Reconocimiento óptico de caracteres)

Tecnología que permite a las computadoras leer y comprender texto impreso en imágenes. El OCR utiliza algoritmos de procesamiento de imágenes para identificar los caracteres en la imagen y convertirlos en texto legible por la máquina. (Smith, 2007).

2.1.7. Transferencia de aprendizaje

Técnica utilizada en el aprendizaje automático que consiste en utilizar un modelo pre-entrenado para una tarea y adaptarlo para otra tarea relacionada. La transferencia de aprendizaje es útil cuando se tiene un conjunto limitado de datos para la tarea de interés. (Pan & Yang, 2010).

2.1.8. Base de datos relacional

Una base de datos que se organiza en tablas relacionales, donde cada tabla tiene una clave primaria y puede tener una o más claves externas que la relacionan con otras tablas

2.1.9. SQL (Structured Query Language)

Un lenguaje de programación utilizado para comunicarse con bases de datos relacionales. SQL se utiliza para crear, actualizar y consultar datos almacenados en una base de datos. (Silberschatz, Korth & Sudarshan, 2010)

2.1.10. Norma ANSI/ISA-101

La norma ANSI/ISA-101, titulada "Interfaces Humanas-Máquina para Sistemas de Automatización de Procesos", es una guía exhaustiva desarrollada por la Sociedad Internacional de Automatización (ISA). Su enfoque se centra en el diseño e implementación de interfaces gráficas del operador (HMI) en entornos industriales y de automatización.

El propósito principal de esta guía es abordar los desafíos asociados con la visualización y comprensión de datos complejos en sistemas de control industrial. Ofrece directrices específicas para lograr una presentación eficaz de información crítica en las pantallas de HMI, con el objetivo de optimizar la toma de decisiones y mejorar la eficiencia operativa. Para lograrlo, establece principios de diseño que incluyen la organización coherente de información, el uso de colores y símbolos claros, asimismo la disposición intuitiva de elementos.

La norma también considera la interacción humano-máquina, poniendo énfasis en la forma en que los operadores se relacionan con la interfaz gráfica. Se abordan consideraciones ergonómicas para asegurar que la interfaz sea fácil de navegar y que las acciones necesarias sean intuitivas y rápidas. Además, se destaca la importancia de proporcionar información contextual adecuada y alertas claras para situaciones de riesgo o cambios inesperados en el proceso.

La relevancia de la ANSI/ISA-101 es especialmente notable en entornos donde la seguridad y la eficiencia son esenciales, como en la industria petrolera, química, plantas de energía y manufactura. Al presentar información de manera clara y coherente, la guía busca reducir la carga cognitiva de los operadores, contribuyendo a prevenir errores y mejorar el rendimiento operativo. (Norma ANSI/ISA-101, 2015).

2.2. Contribución del trabajo

En InfoDesign Colombia manejan volúmenes de trabajo elevado y metas por tiempo que se deben cumplir según un cronograma establecido, generando así una carga laboral no deseada que desencadena en errores humanos por fatiga como: omitir textos, malas conexiones y en general una representación fallida del proceso. Para combatir este problema se propone la implementación de una inteligencia artificial que sirva como apoyo para los diseñadores generando una plantilla con los elementos más recurrentes en cada diseño.

Con esta herramienta se espera reducir el tiempo de desarrollo de cada HMI favoreciendo así a cada diseñador, otorgando facilidad al momento de enfrentar cada gráfico y cumplir con las metas propuestas en menor tiempo y menos desgaste.

Por último, es importante destacar que la herramienta tiene un Dataset inicial y dirigido a un proyecto en específico, especialmente para una planta de una refinería de petróleo, aun así este Dataset se podría actualizar con cientos de elementos nuevos que se deseen identificar en nuevos procesos, de hecho, todo este desarrollo apunta a ser una herramienta polivalente que sea capaz de identificar elementos de diferentes industrias y cada vez generar una plantilla más cercana al diseño completo que podría crear un diseñador profesional de HMI..

Capítulo 3

3. Metodología

Este capítulo desarrolla la metodología utilizada para cumplir los objetivos planteados, describiendo detalladamente cada una de las fases del proyecto.

3.1. Enfoque

El desarrollo de este proyecto se basó principalmente en la metodología de trabajo SCRUM, conocida por su enfoque en entregas incrementales y una planificación iterativa. Si bien se realizó una planificación detallada al inicio del proyecto, SCRUM permite la inspección y adaptación continua, lo que significa, que se pueden realizar ajustes y mejoras en cada sprint. Esta flexibilidad garantiza que el proceso pueda adaptarse según las necesidades y los aprendizajes obtenidos en cada etapa, permitiendo una evolución constante del proyecto.

Además de la metodología SCRUM, es importante destacar que el proyecto también incorporó un enfoque cuantitativo para evaluar la confiabilidad de los resultados de la herramienta. Se generaron diversos indicadores de desempeño que proporcionan una medida objetiva y numérica para evaluar la confiabilidad de la herramienta en función de los objetivos establecidos. Estos indicadores permitieron tener una visión clara y cuantificable del rendimiento de la herramienta, brindando información valiosa para la toma de decisiones y la mejora continua, la combinación de la metodología SCRUM y el enfoque cuantitativo permitió un desarrollo ágil y orientado a los resultados.

3.2. Desarrollo de actividades

3.2.1. Fase 1: Identificar elementos estáticos

Esta fase se divide en dos partes fundamentales. La primera actividad planteada consiste en la creación de un algoritmo de visión artificial en Python, que permita identificar ciertos objetos en un Diagrama P&ID. Tras llevar a cabo un par de reuniones con el equipo de dirección de

proyectos, se llega a un acuerdo para desarrollar una primera versión de la herramienta. Esta versión deberá identificar todos los objetos que se consideran comunes en el desarrollo de HMIs para el proyecto en curso.

Para ello, se procede a crear una lista de requerimientos en la que se especifican los elementos estáticos que se deben identificar:

- Entradas y salidas del diagrama, también conocidas como banderas.
- Válvulas (opcional).
- Hornos.
- Intercambiadores de calor.
- Sistemas de ventilación.
- Separadores de crudo.
- Tanques de almacenamiento.
- Generadores de vapor.
- Turbinas.
- Líneas de conexión.

Es importante destacar que algunos de los elementos mencionados en la lista anterior tienen subtipos asociados. Esto significa que existen variaciones para dichos objetos que, aunque cumplen la misma función o alguna similar, requieren una representación gráfica diferente en términos de tamaño o forma. Estos cambios son relevantes debido a una solicitud del cliente, quien ha dejado claro que estas distinciones son necesarias para lograr una mejor comprensión de la interfaz por parte de los operadores de planta.

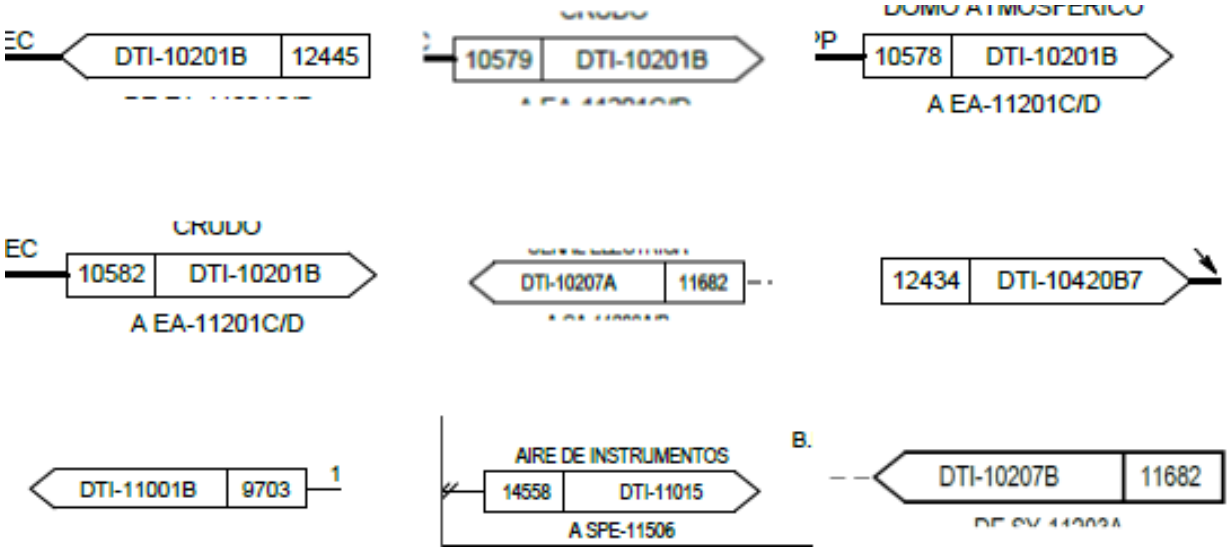
Se estima un tiempo de 3 semanas para el desarrollo de esta actividad. Durante este período, se llevará a cabo un avance semanal y se realizarán las modificaciones pertinentes al proceso. A continuación, se presentan los avances por semana.

Semana 1: Segmentación

El proceso comienza segmentando y etiquetando los elementos que se desean identificar, como válvulas, tanques, banderas de entrada y salida, intercambiadores, y todos los demás descritos en la lista de requerimientos. El primer paso consiste en realizar la segmentación de los elementos, es decir, inspeccionar cada P&ID. Es importante tener en cuenta que se dispone de un número limitado de P&ID, ya que esta herramienta, en su primera versión, está diseñada para una planta específica. Durante la inspección, se busca generar una biblioteca o conjunto de datos de imágenes (Dataset) para cada elemento. Esto implica encontrar variaciones específicas del elemento, como etiquetas, elementos acompañantes, numeraciones o texto dentro del objeto. De esta manera se conforma la biblioteca de imágenes, como se muestra en las siguientes figuras:

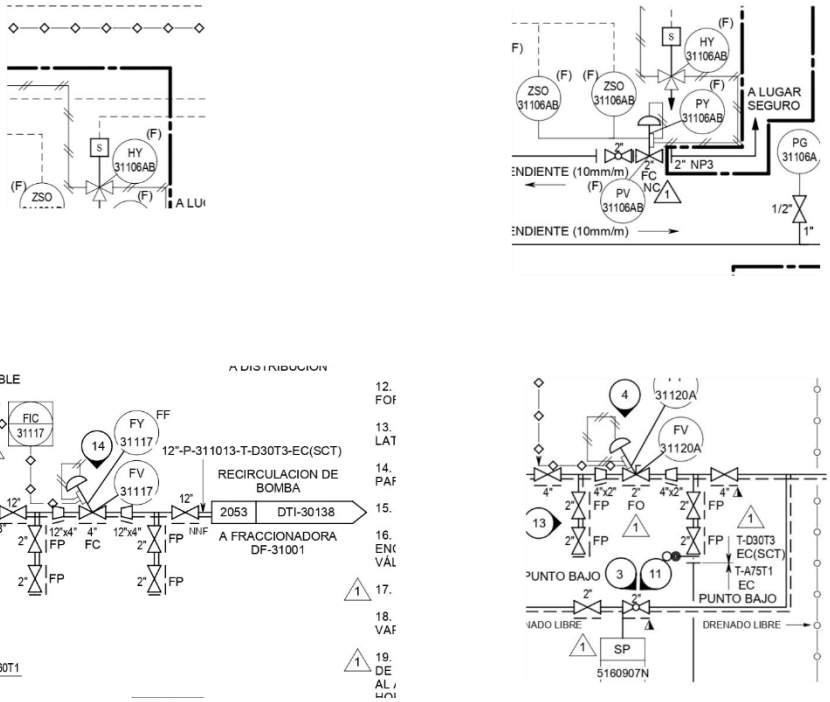
Banderas

Figura 1. Dataset de imágenes de entradas y salidas.



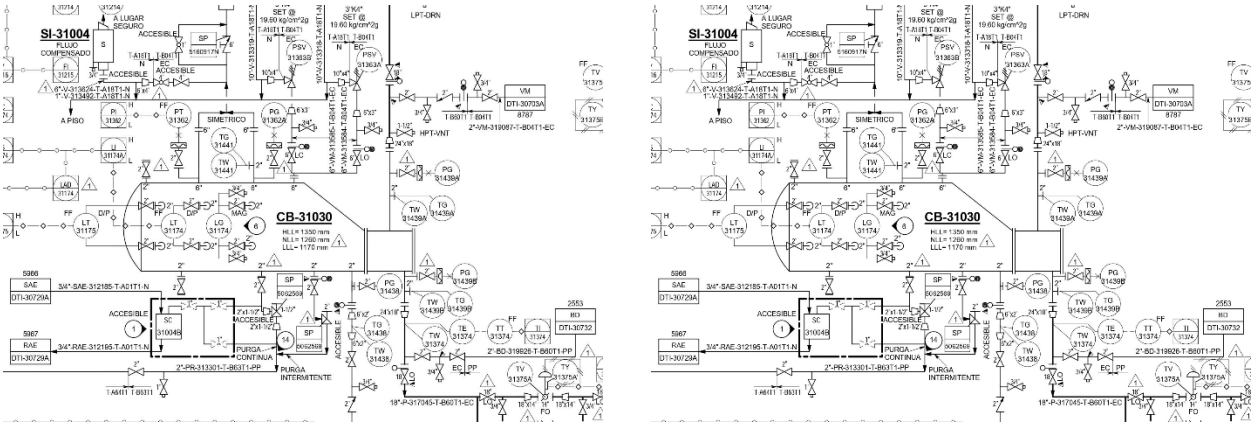
Válvulas

Figura 2. Dataset de válvulas.



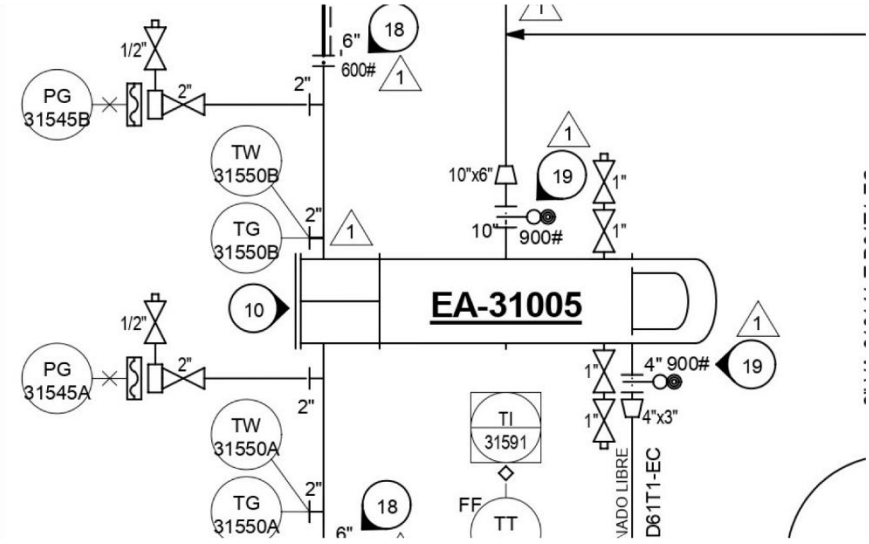
Generadores de vapor

Figura 3. Dataset de generadores de vapor



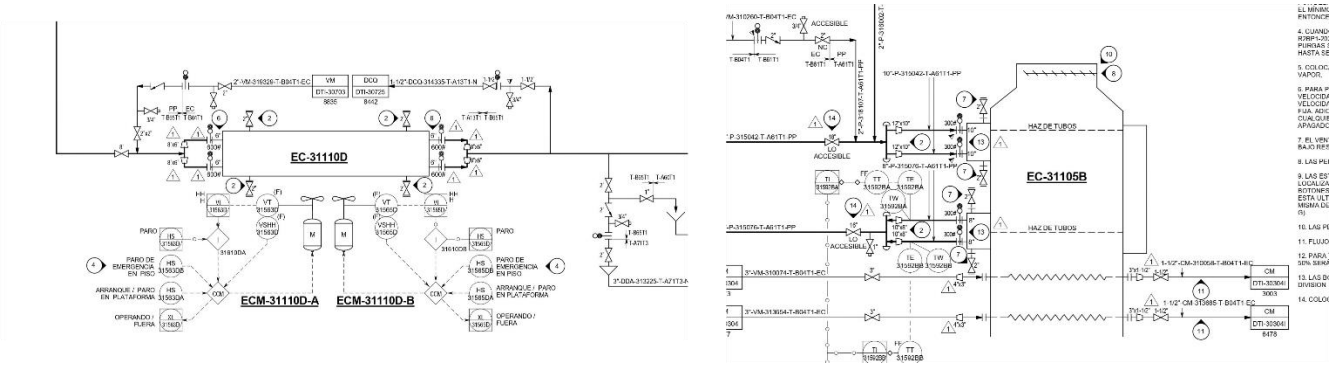
Intercambiadores de calor

Figura 4. Dataset intercambiadores de calor.



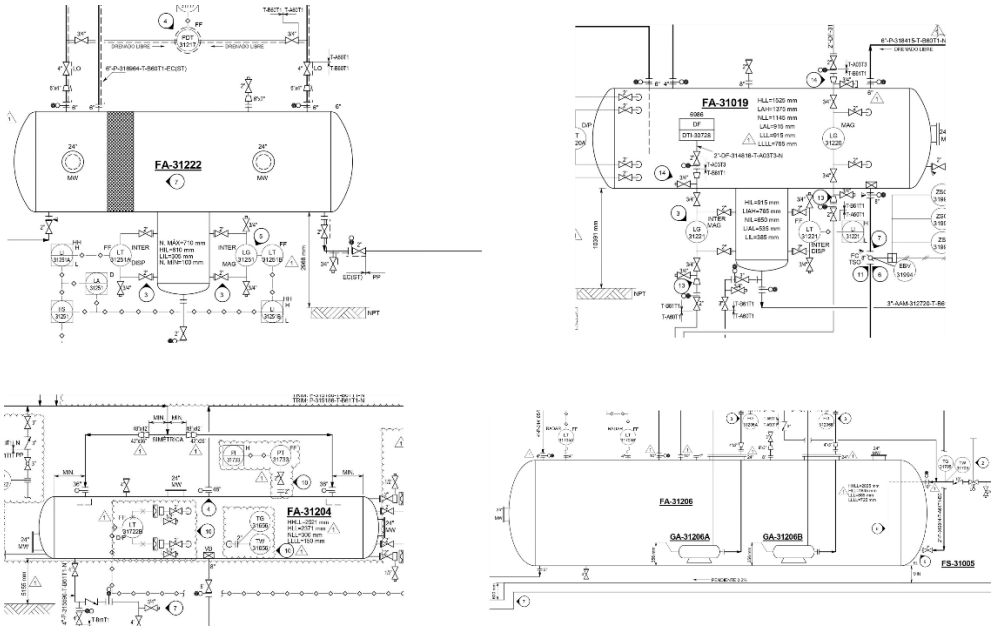
Sistema de ventilación

Figura 5. Dataset ventilación.



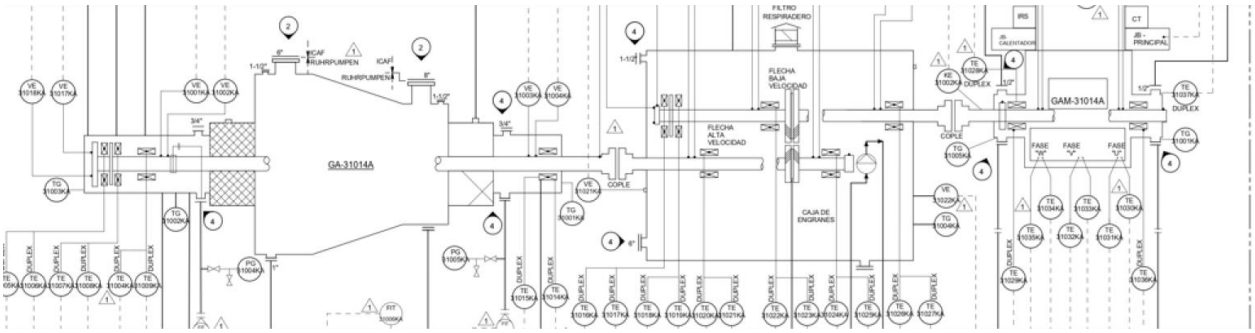
Separadores de crudo

Figura 6. Dataset separadores de crudo



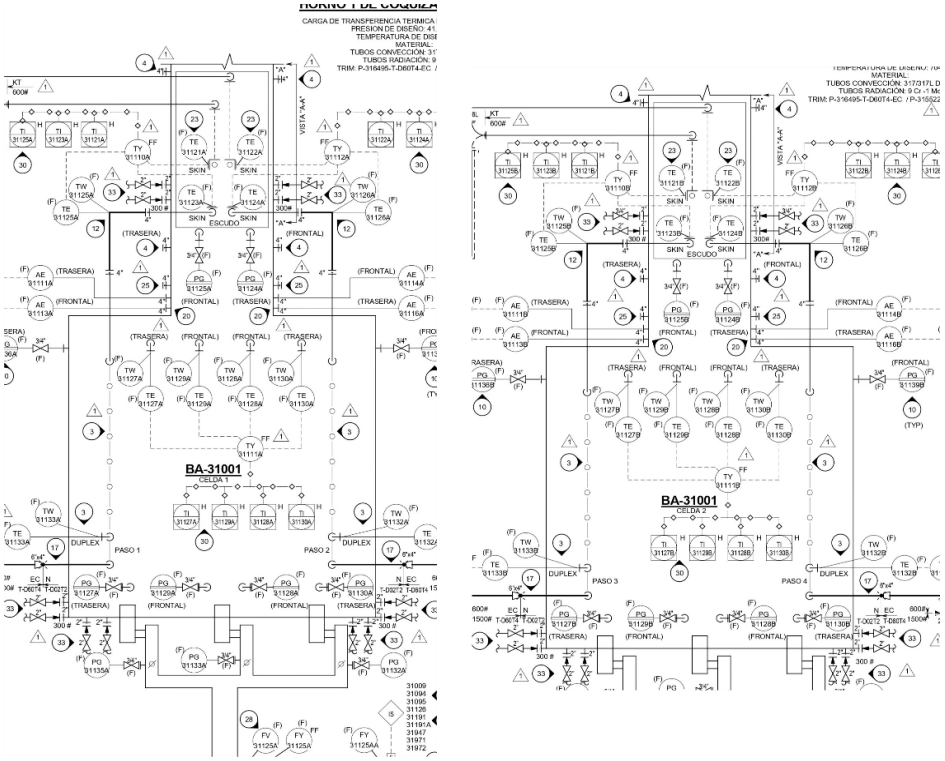
Turbinas

Figura 7. Dataset turbinas



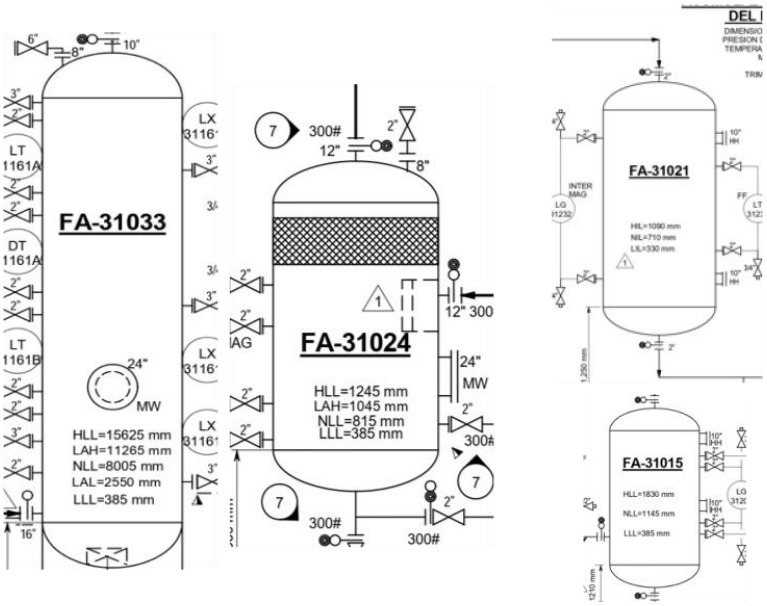
Hornos

Figura 8. Dataset hornos



Tanques

Figura 9. Dataset tanques



Nota: Es de vital importancia aclarar que ninguno de los datasets está compuesto solo por las imágenes que se logran apreciar en su figura correspondiente, esta figura funciona a manera de ejemplo, todos los dataset del presente trabajo serán adjuntados en la carpeta final del proyecto para así evitar la saturación del documento con cientos de imágenes.

Al realizar la revisión de la primera semana se llega a la conclusión de que el dataset para cada tipo de elemento era muy limitado y queda planteado realizar un ensayo con este dataset y analizar algunos resultados preliminares. Cabe mencionar que se empieza desde esta semana la investigación del mejor modelo para realizar la detección de los objetos.

Semana 2: Etiquetado

Una vez que todas las imágenes están segmentadas, se requiere realizar el etiquetado. Este proceso debe realizarse de forma manual y existen varias formas de etiquetar imágenes, como crear un documento con código, utilizar software especializado o aprovechar alternativas web. Todas estas opciones tienen el mismo objetivo: generar un archivo de texto con las coordenadas del elemento etiquetado.

En este trabajo, se optó por utilizar una alternativa web llamada MAKESENSE debido a su facilidad de acceso, gratuidad y su interfaz de usuario intuitiva. Si bien existen otras herramientas web disponibles, las características mencionadas hacen de MAKESENSE la opción adecuada para este proceso en particular.

El funcionamiento de esta herramienta es muy sencillo. Se cargan las imágenes que se desean etiquetar **(Fig. 10)** y luego se crean las etiquetas correspondientes para cada elemento **(Fig. 11)**. A continuación, se dibuja un rectángulo que abarca el área de interés seleccionada **(Fig. 12)**. Por último, se descarga un archivo .txt por cada imagen, el cual contiene cinco columnas: el ID de la etiqueta, los valores ymin, xmin, ymax y xmax **(Fig. 13)**.

A continuación, se muestra la interfaz de usuario de la herramienta:

Figura 10. Interfaz makesense para carga de imágenes

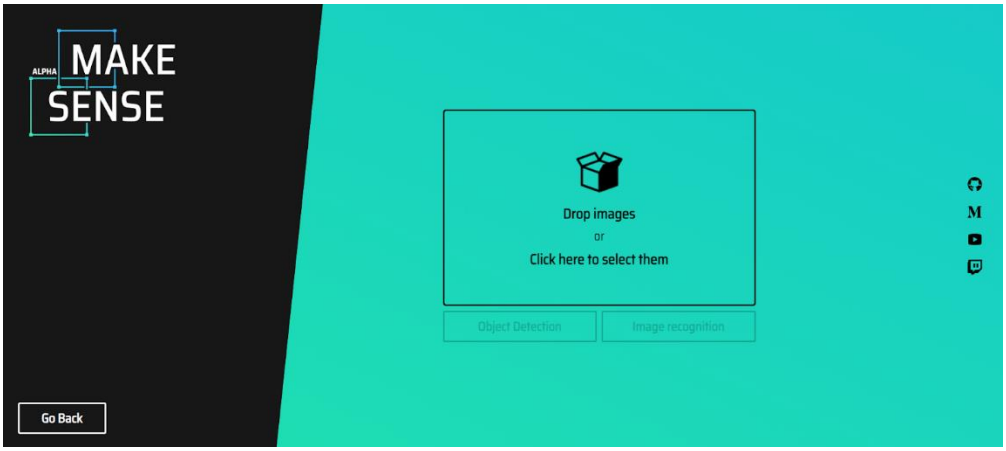


Figura 11. Interfaz makesense para crear las etiquetas

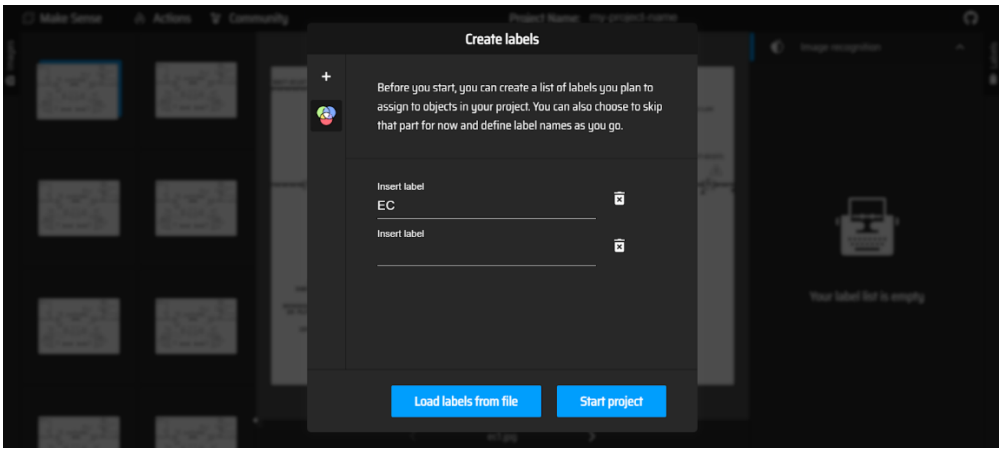


Figura 12. Dibujo rectángulo identificador

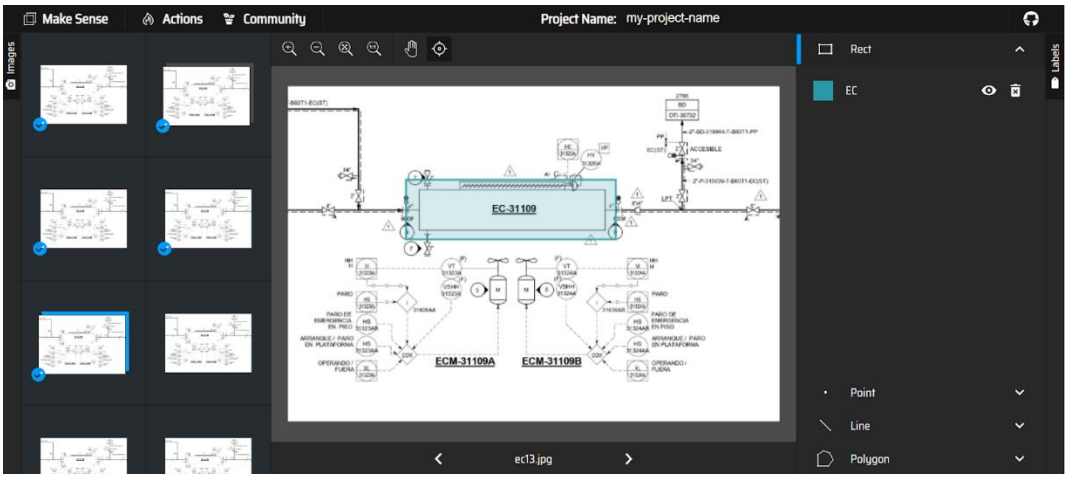
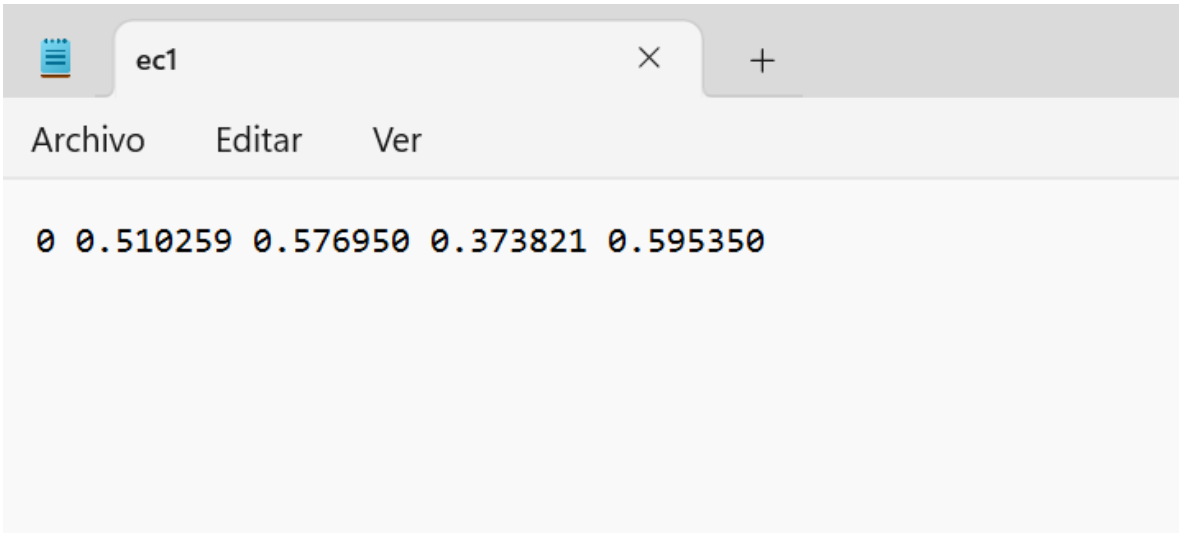


Figura 13. Archivo de texto con coordenadas



En la revisión de la segunda semana, en colaboración con el asesor, se llega a un acuerdo basado en su experiencia. Se reconoce que todos los conjuntos de datos tienen un tamaño muy reducido. A partir de prácticas anteriores en otros proyectos, se sabe que para lograr una identificación con resultados aceptables, es necesario contar con aproximadamente 128 variaciones de imágenes para cada elemento que se desea identificar. Es en este punto donde surge la idea de crear un algoritmo para clonar las imágenes y sus respectivas etiquetas, con el fin de obtener un número suficiente de imágenes para realizar una identificación aceptable.

Es importante destacar que, en un enfoque más amplio del proyecto, sería factible buscar en los P&ID de toda la refinería hasta alcanzar la cantidad de imágenes esperada. Sin embargo, debido a que solo se tenía acceso a una planta específica, se optó por clonar la información ya existente como alternativa.

Semana 3: Clonación de datos

Para abordar el problema de la falta de información en los conjuntos de datos durante la segunda semana, se ha desarrollado un código en Python. Este código tiene la función de clonar una imagen especifica junto con sus respectivas coordenadas en formato .txt. Al ejecutar este

código, se deben proporcionar la ruta de la imagen, el número de veces que se desea clonar y la ruta donde se almacenarán las copias.

Este proceso de clonación es muy rápido. Siguiendo la convención establecida en el proyecto, cada imagen se nombra según su tipo y se le asigna un número consecutivo. Por ejemplo, el intercambiador de calor se denomina "ea" seguido de su número identificador, como "ea01", "ea02" y así sucesivamente. Al crear las copias, además de proporcionar la ruta y el número de copias deseadas, se debe ingresar el último número existente en el conjunto de datos. Esto permitirá comenzar la cuenta a partir del número siguiente a este último.

Figura 14. Algoritmo clonación de datos.

```
from PIL import Image
img= 'ec3'
num = 257
crear = 128
# Abrir la imagen original
imagen_original = Image.open("C:/Users/Camil/Documents/Pasantia/Datos_entrenamiento/data_ec2/images/train/"+img+".jpg")

# Generar copias de la imagen
for i in range(crear):
    imagen_original.save(f"C:/Users/Camil/Documents/Pasantia/Datos_entrenamiento/data_ec2/images/train/ec{i+num}.jpg")

# Abrir el archivo de texto original
with open("C:/Users/Camil/Documents/Pasantia/Datos_entrenamiento/data_ec1/labels/train/"+img+".txt", "r") as archivo_original:
    # Leer el contenido del archivo original
    contenido = archivo_original.read()

# Generar copias del archivo de texto
for i in range(crear):
    # Crear un nuevo archivo con un nombre diferente para cada copia
    num2=i+num
    nombre_archivo = f"ec{i+num}.txt"
    with open("C:/Users/Camil/Documents/Pasantia/Datos_entrenamiento/data_ec2/labels/train/"+nombre_archivo, "w") as archivo_copia:
        # Escribir el contenido del archivo original en el archivo copia
        archivo_copia.write(contenido)
```

En la revisión de esta semana se llega a la conclusión de que se deben poner a prueba los datasets, así que se procede a crear el algoritmo para identificar cada elemento.

Semana 4: Desarrollo y prueba del algoritmo

Debido a contratiempos en la actividad anterior, el desarrollo del código se ha retrasado. Sin embargo, aún se tiene suficiente tiempo de esta semana para realizar el primer acercamiento a la detección de los objetos previamente segmentados y etiquetados.

Con el fin de desarrollar el algoritmo, se ha llevado a cabo una investigación en segundo plano para determinar el modelo que mejor se adapte a las necesidades del proyecto. Como resultado, se han identificado seis candidatos prometedores:

- YOLO (You only look once)
- Faster R-CNN
- SSD (Single Shot MultiBox Detector)
- RetinaNet
- EfficientDet
- Cascade R-CNN

Cada uno de estos modelos posee atributos distintivos que pueden proporcionar mejores resultados según la actividad que se pretenda realizar. A continuación, se presenta una tabla con las ventajas y desventajas de cada modelo:

Tabla 1. Ventajas y desventajas por modelo de detección.

Modelo	Ventajas	Desventajas
YOLOv5	- Alta velocidad y eficiencia. - Implementación ligera y fácil de usar. - Buena precisión general. - Soporte para detección en tiempo real. - Capacidad para trabajar en entornos en línea como Google Colab.	- Puede tener menor precisión en objetos pequeños y detecciones de baja resolución. - Puede requerir ajustes de parámetros para maximizar la precisión en ciertos casos de uso.
Faster R-CNN	- Mayor precisión en objetos pequeños y detecciones precisas. - Buena capacidad de generalización. - Capacidad para detectar objetos en imágenes de alta resolución. - Flexibilidad para ajustar hiper parámetros.	- Mayor complejidad computacional y más lento en tiempo de ejecución. - Requiere recursos computacionales más altos para el entrenamiento. - Mayor complejidad en la implementación y configuración.

SSD	- Alta velocidad y buen rendimiento. - Implementación sencilla. - Eficiente en la utilización de recursos computacionales.	- Puede tener una menor precisión en objetos pequeños y detecciones de baja resolución. - Requiere ajustes de escala para un mejor rendimiento en diferentes tamaños de objetos. - Puede sufrir de problemas de precisión con objetos superpuestos.
RetinaNet	- Buen rendimiento en objetos pequeños y escalas variadas. - Buena detección de objetos difíciles. - Funciona bien con imágenes de baja resolución.	- Mayor complejidad computacional que modelos más simples. - Requiere más tiempo de entrenamiento debido a la arquitectura y complejidad. - Puede tener un rendimiento inferior en la detección de objetos grandes. - Mayor complejidad computacional y más lento en tiempo de ejecución.
EfficientDet	- Logra un equilibrio entre precisión y eficiencia computacional. - Alto rendimiento con menos parámetros. - Eficiente en el uso de recursos computacionales.	- Puede requerir más recursos computacionales para el entrenamiento. - Requiere ajustes de hiper parámetros para obtener un mejor rendimiento. - Puede requerir una implementación más compleja para adaptarse a diferentes configuraciones.
Cascade R-CNN	- Mayor precisión debido a múltiples etapas de detección. - Capacidad para detectar objetos en cascada con diferentes niveles de dificultad. - Mejora la capacidad de suprimir falsos positivos.	- Más lento en tiempo de ejecución debido a las etapas adicionales. - Mayor complejidad en la implementación y configuración. - Requiere más recursos computacionales y tiempo de entrenamiento.

En este punto, es importante definir las necesidades de detección, buscando encontrar un equilibrio entre el tiempo de ejecución, entrenamiento, el consumo eficiente de recursos hardware

y, por supuesto, la precisión en la detección. Con esto en mente, nos adentramos en la etapa de filtrado, donde se genera una matriz de valores.

Tabla 2. Matriz de valor para modelos de detección

Modelo	Tiempos	Recursos computacionales	Precisión	Total
Yolov5	9	10	8	9
Faster R-CNN	7	7	9	7,66
SSD	9	9	6	8
RetinaNet	7	7	9	7,66
EfficientDet	7	8	8	7,66
Cascade R-CNN	7	6	9	7,33

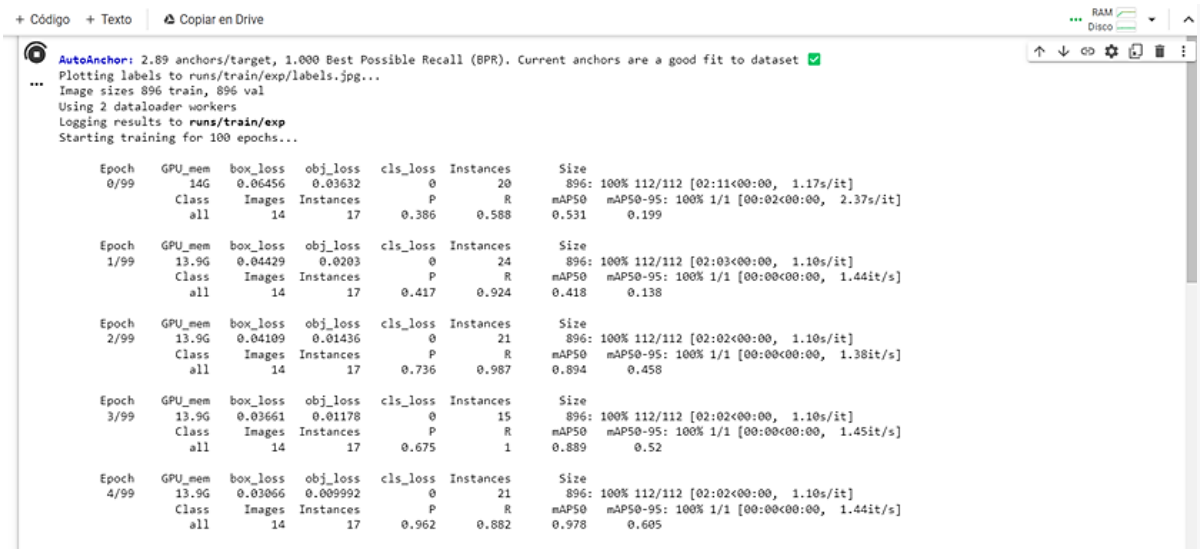
En esencia, ninguno de los modelos mencionados es realmente malo, ya que todos pueden ser utilizados en aplicaciones similares. Sin embargo, al evaluar las condiciones específicas del proyecto, se determina que YOLOv5 es el modelo que mejor se adecúa a dichas condiciones. YOLOv5 presenta excelentes tiempos de ejecución y entrenamiento, no requiere un consumo excesivo de recursos computacionales locales y logra una precisión satisfactoria en la detección de objetos.

Una vez seleccionado el modelo, se procede a realizar el entrenamiento utilizando Google Colab y la implementación de YOLOv5. Esta plataforma nos brinda la capacidad de realizar el entrenamiento en la nube, lo que implica que se nos proporciona el hardware necesario para obtener un rendimiento óptimo.

Para llevar a cabo el proceso de entrenamiento, es necesario subir una carpeta al entorno de Google Colab que contenga tanto las imágenes como las etiquetas. Dentro de esta carpeta, se deben incluir las imágenes destinadas al entrenamiento, así como algunas imágenes

adicionales que servirán para validar este proceso. Así mismo, se deben proporcionar las etiquetas correspondientes a las imágenes de entrenamiento y validación.

Figura 15. Entrenamiento del modelo en Google Colab YoloV5.



Nota: En la imagen adjunta se muestra el proceso de entrenamiento del modelo. En este caso, se trata de un entrenamiento específico para la detección del elemento "intercambiador de calor", utilizando 100 épocas. Es importante destacar que este proceso se repite para cada elemento que se desea detectar. Al finalizar el entrenamiento, se obtiene un archivo que contiene los pesos del modelo. Estos pesos serán posteriormente cargados en el algoritmo desarrollado en Python, permitiendo así realizar la detección de los elementos deseados.

En el desarrollo del algoritmo se utilizaron dos librerías principales:

- Torch: Centrada principalmente en el procesamiento de tensores, lo que la hace especialmente adecuada para tareas de aprendizaje profundo y redes neuronales.
- PIL: Proporciona una amplia gama de funciones y métodos para abrir, manipular, procesar y guardar imágenes en diferentes formatos.

Después de importar las librerías mencionadas anteriormente, se procede a cargar el modelo mediante el uso de la función torch.hub.load(). En este caso, se utiliza el modelo

personalizado de YOLOv5 y se especifica la ruta donde se encuentra el archivo de pesos del modelo (best_ea.pt).

A continuación, se lee la imagen de interés utilizando el método Image.open(). Es necesario proporcionar la ruta de la imagen como argumento en este comando. Una vez que la imagen se ha leído, se procede a redimensionarla. Esto se debe a que el modelo YOLOv5 tiene algunas limitaciones al procesar imágenes de gran tamaño. En este caso, la imagen original tiene una resolución de 4K, pero con el comando "resized" se redimensiona para alcanzar una resolución Full HD (1920x1080).

Una vez que la imagen está en condiciones óptimas, se pasa al modelo y se obtienen los resultados, que se almacenan en una variable llamada "detect" en este caso. Finalmente, se utiliza el método "show()" para visualizar la imagen con las detecciones encontradas.

Figura 16. Algoritmo en Python para detección de imágenes.

```
# Importamos librerias
import torch
from PIL import Image
# Leemos el modelo
model = torch.hub.load('ultralytics/yolov5', 'custom',
                       path_='C:/Users/Camil/Documents/Pasantia/pythonProject/best_ea.pt')
# Leemos imagenes
img_pil = Image.open(r"ilovepdf_merged_page-0169.jpg")
resized = img_pil.resize((1920, 1080))
# Realizamos las detecciones
detect = model(resized)
detect.show()
```

Nota: El proceso de detección es el mismo para todos los elementos, se debe cambiar el archivo de pesos para el elemento que se quiera identificar, es decir, en torch.hub.load() se cargaron archivos como: best_ea.pt, best_cb.pt, best_ga.pt y así para cada elemento.

Cuando el proceso de detección es llevado a cabo se obtiene el resultado presentado en las siguientes figuras:

Figura 17. Detección de entradas y salidas.

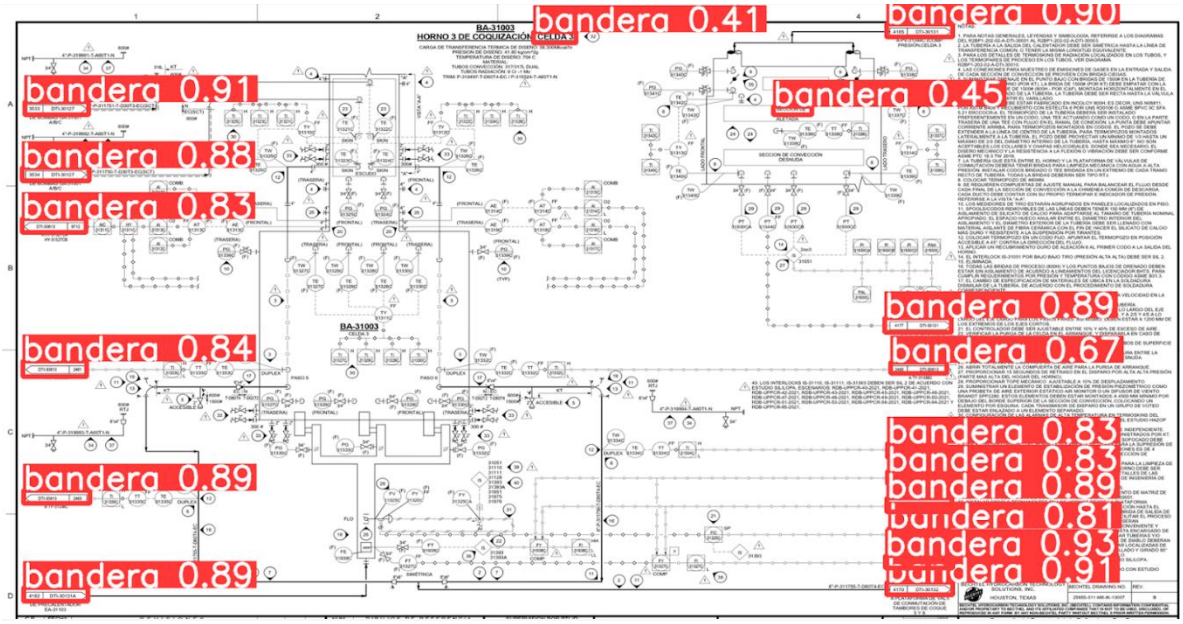


Figura 18. Detección de válvulas.

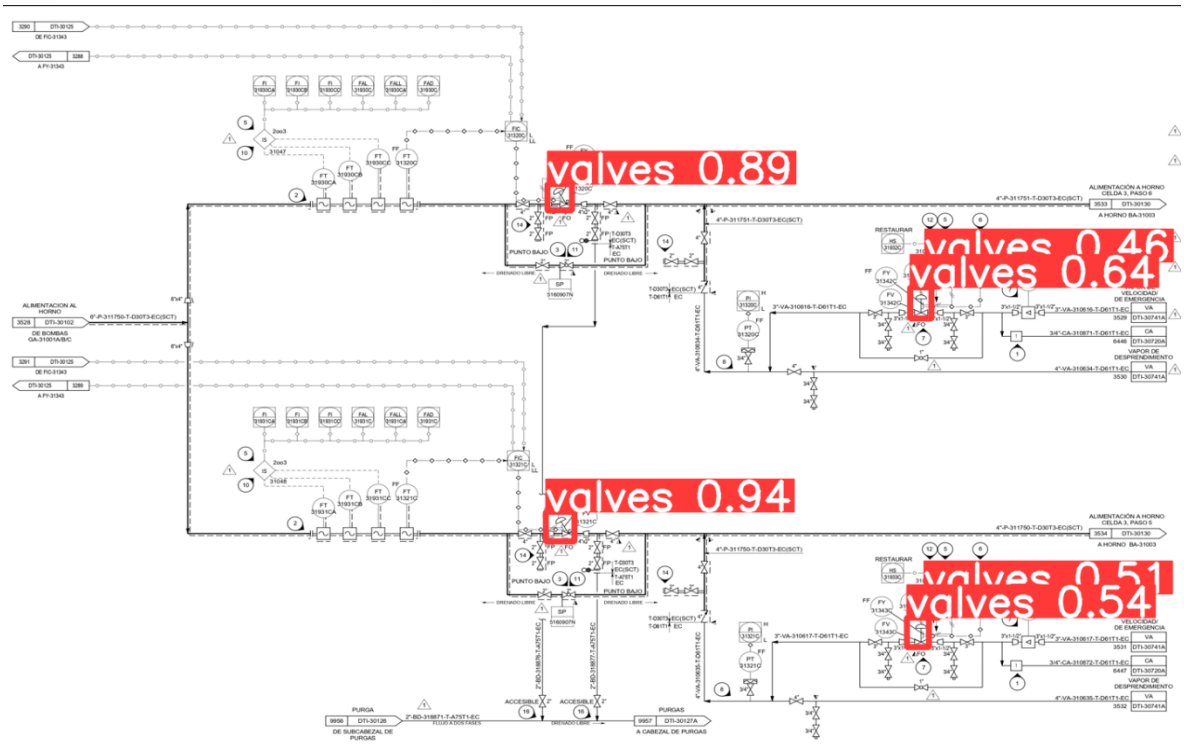


Figura 19. Detección de hornos.

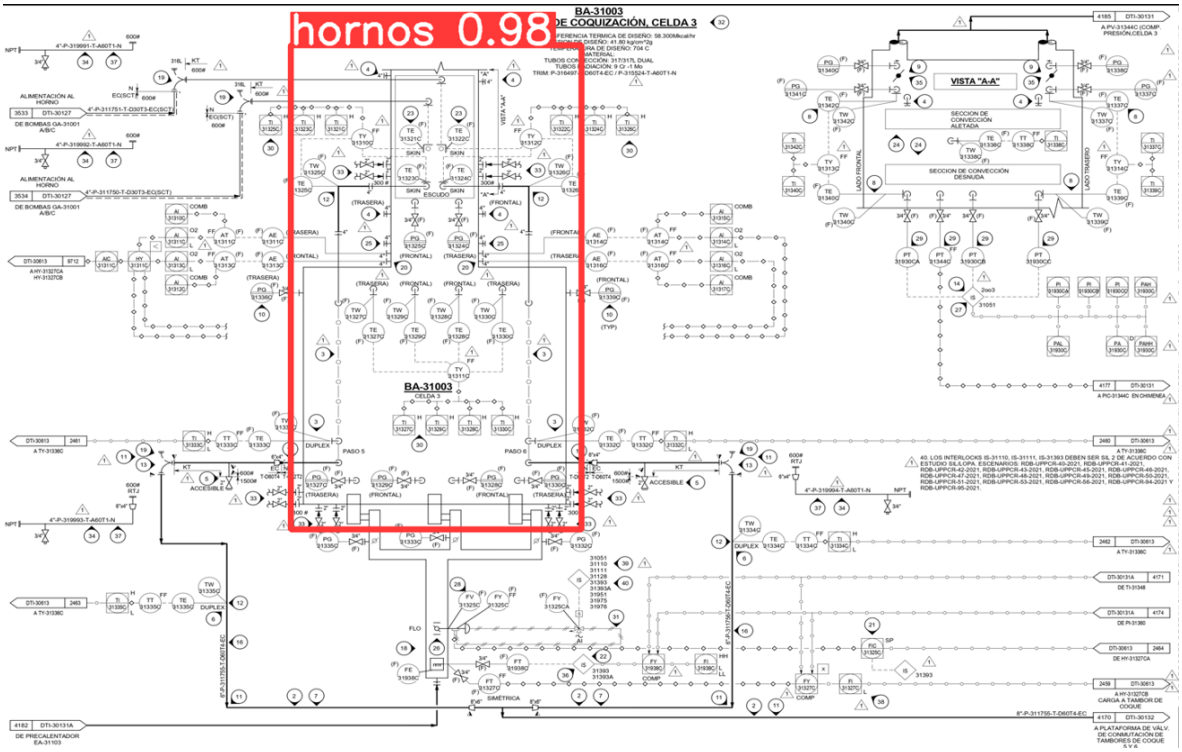


Figura 20. Detección de intercambiadores de calor.

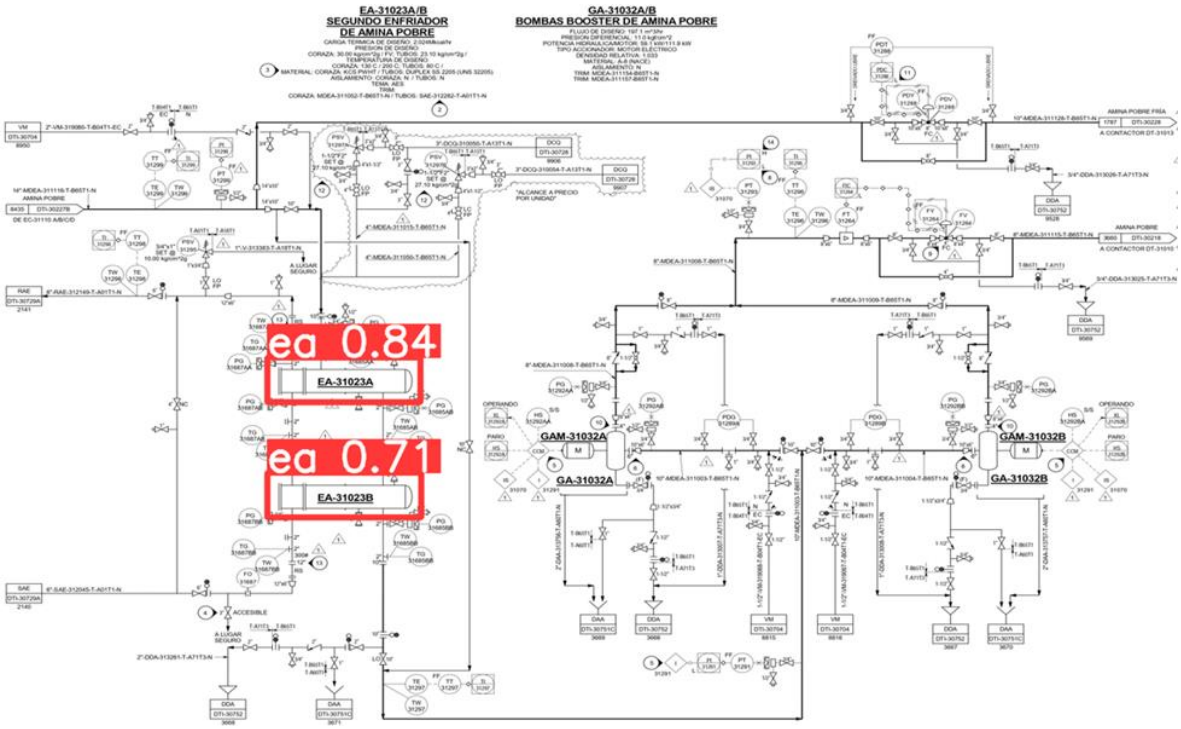


Figura 21. Detección de sistemas de ventilación variante 1.

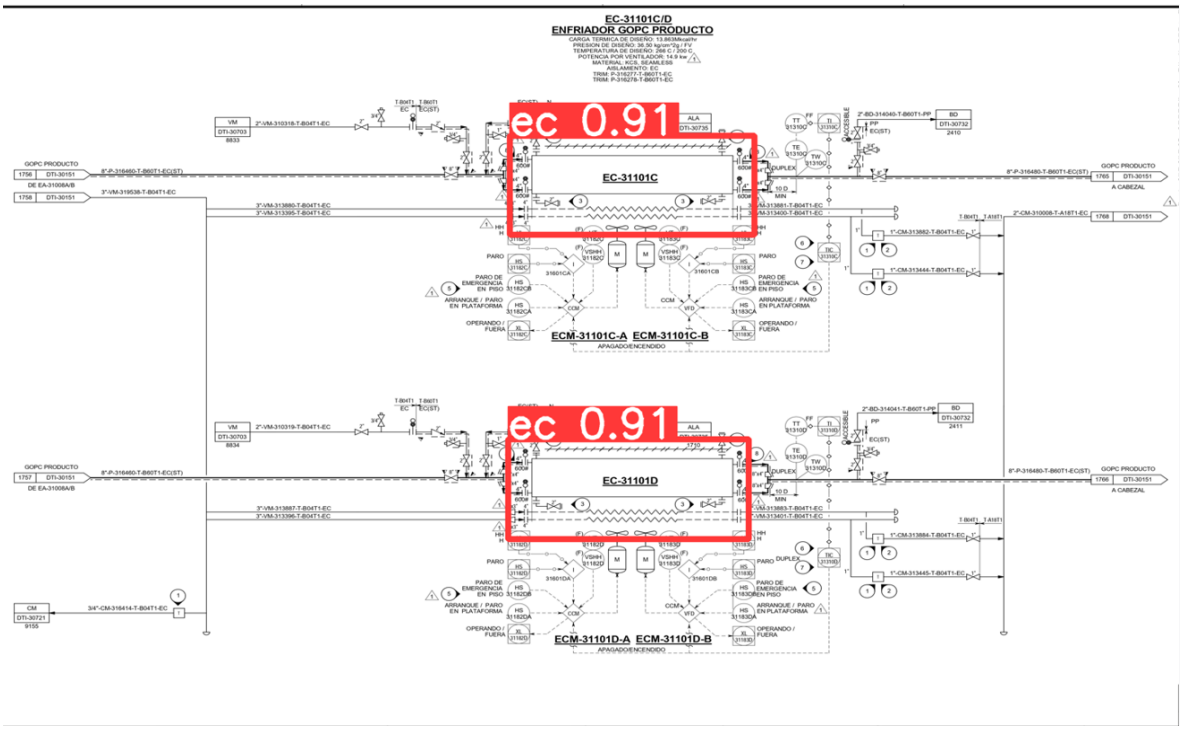


Figura 22. Detección de sistemas de ventilación variante 2.

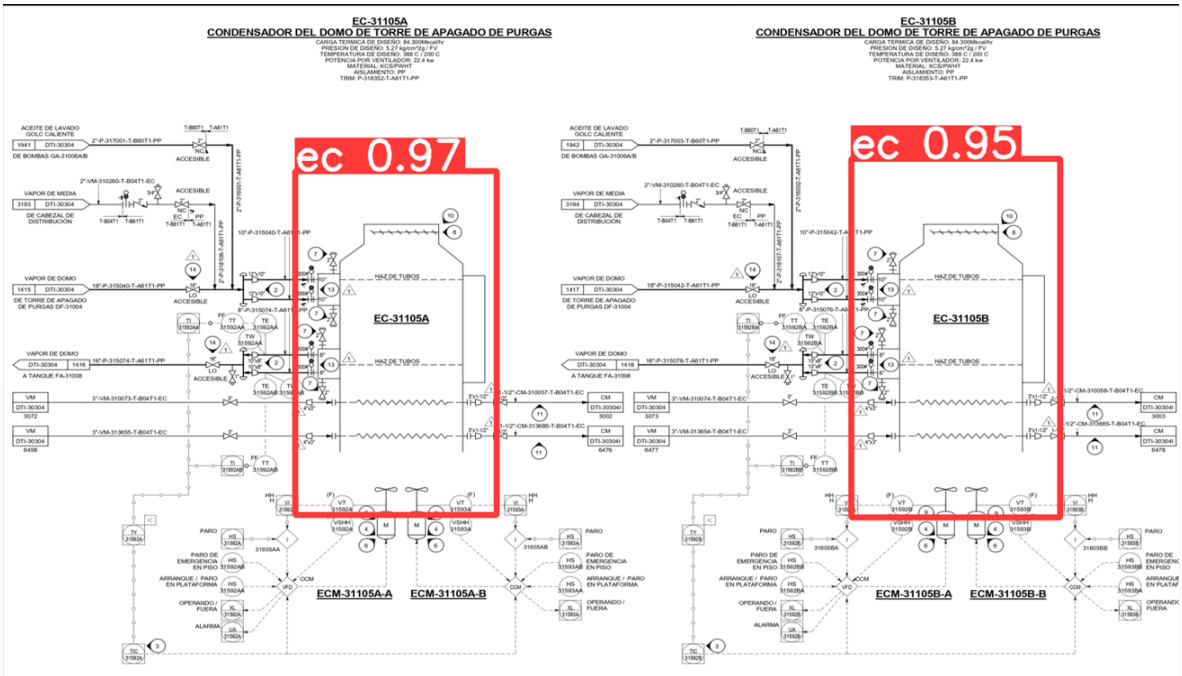


Figura 23. Detección de separadores de crudo variante 1.

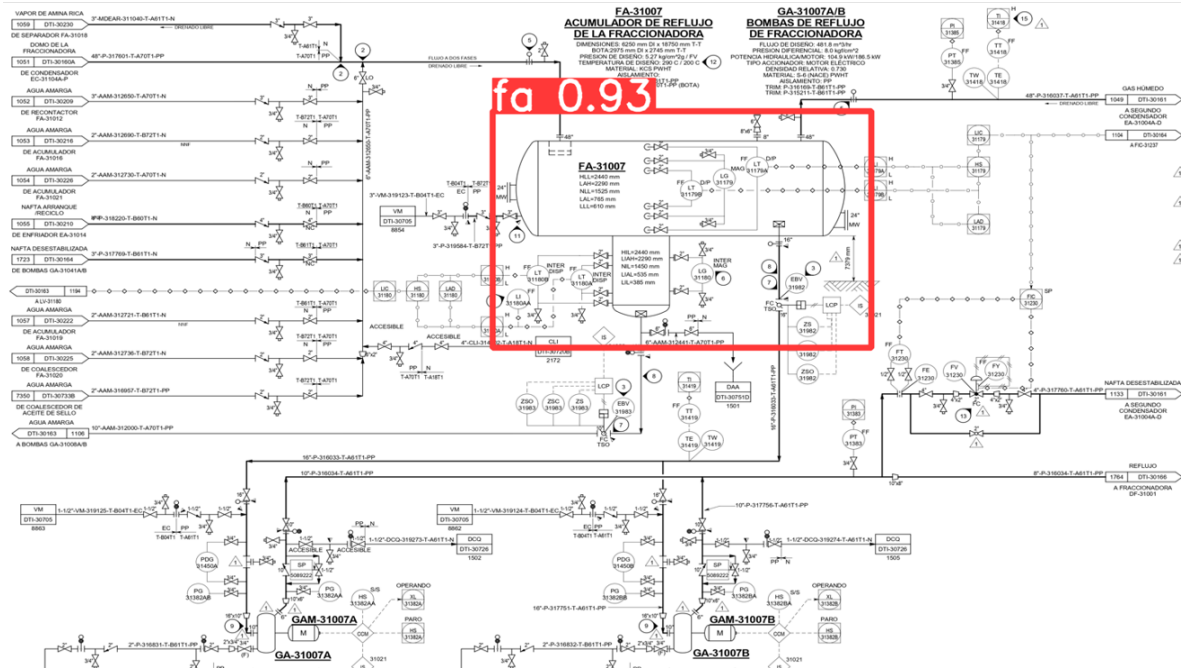


Figura 24. Detección de separadores de crudo variante 2.

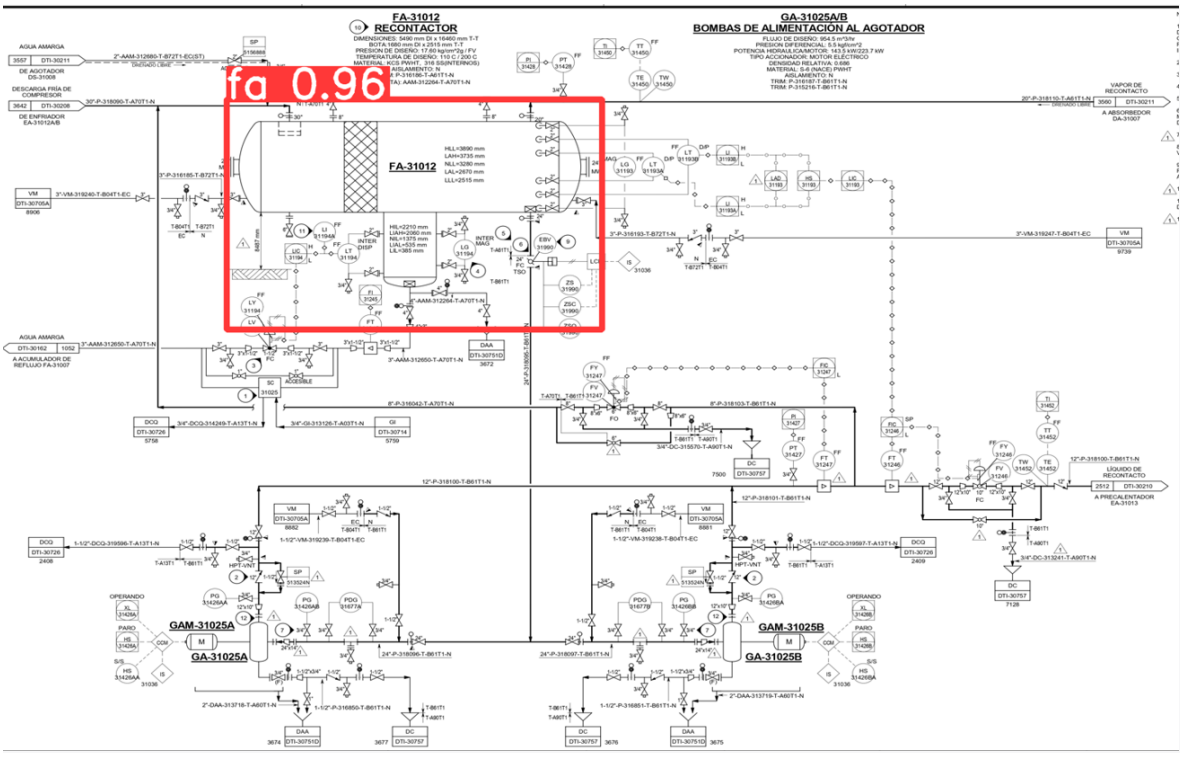


Figura 25. Detección de separadores de crudo variante 3.

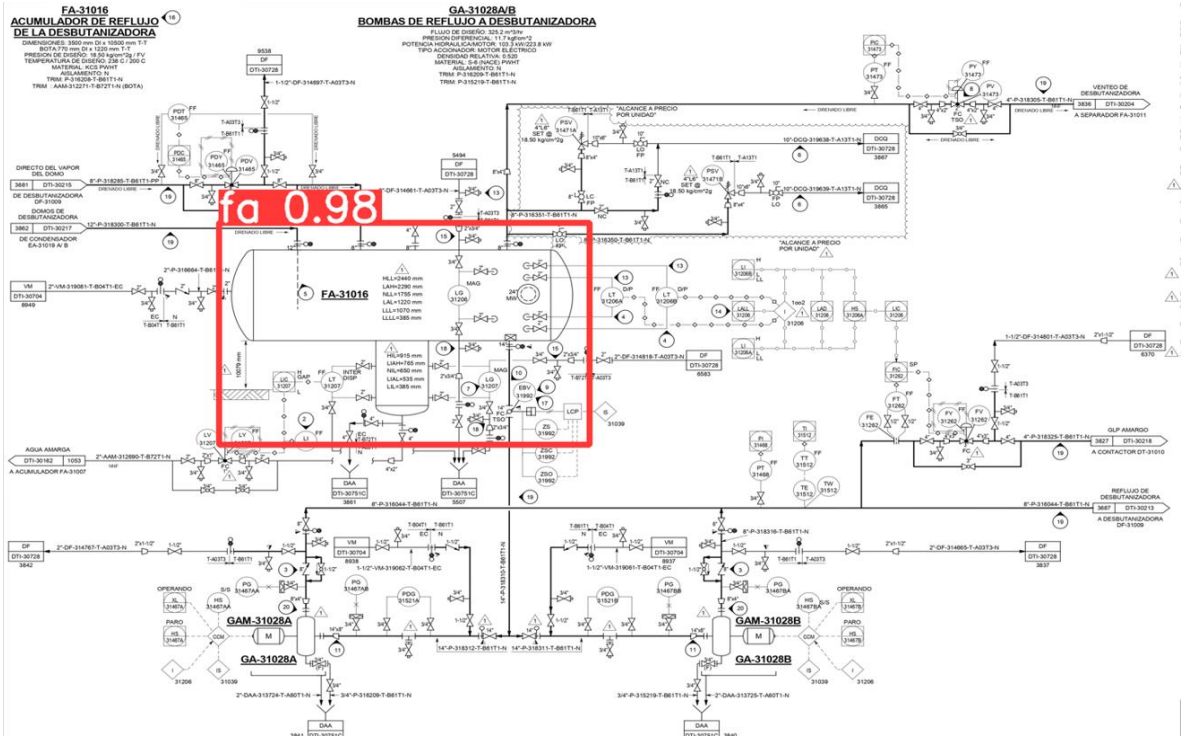


Figura 26. Detección de separadores de crudo variante 4.

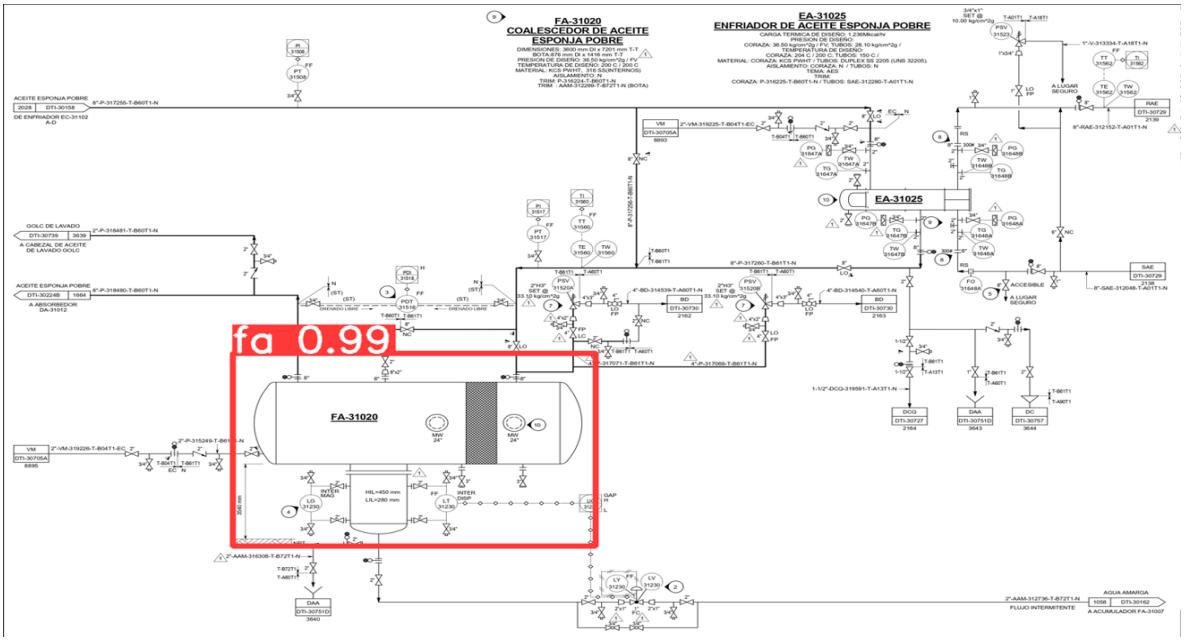


Figura 27. Detección de tanques variante 1.

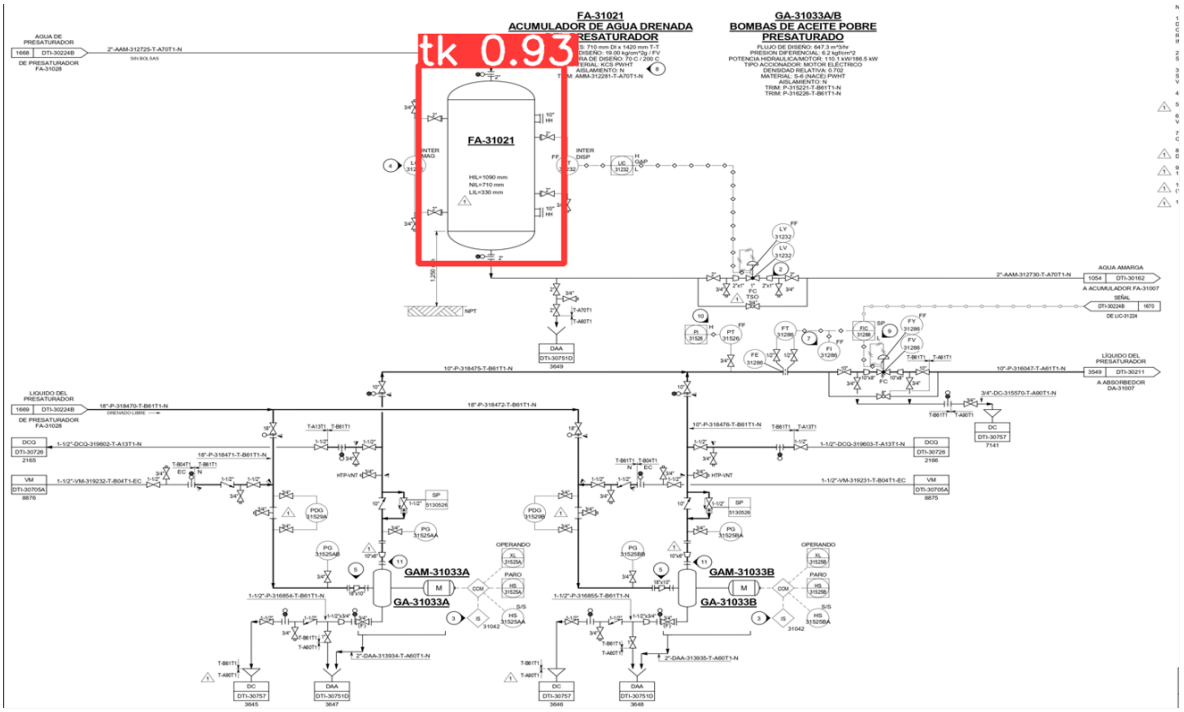


Figura 28. Detección de tanques variante 2.

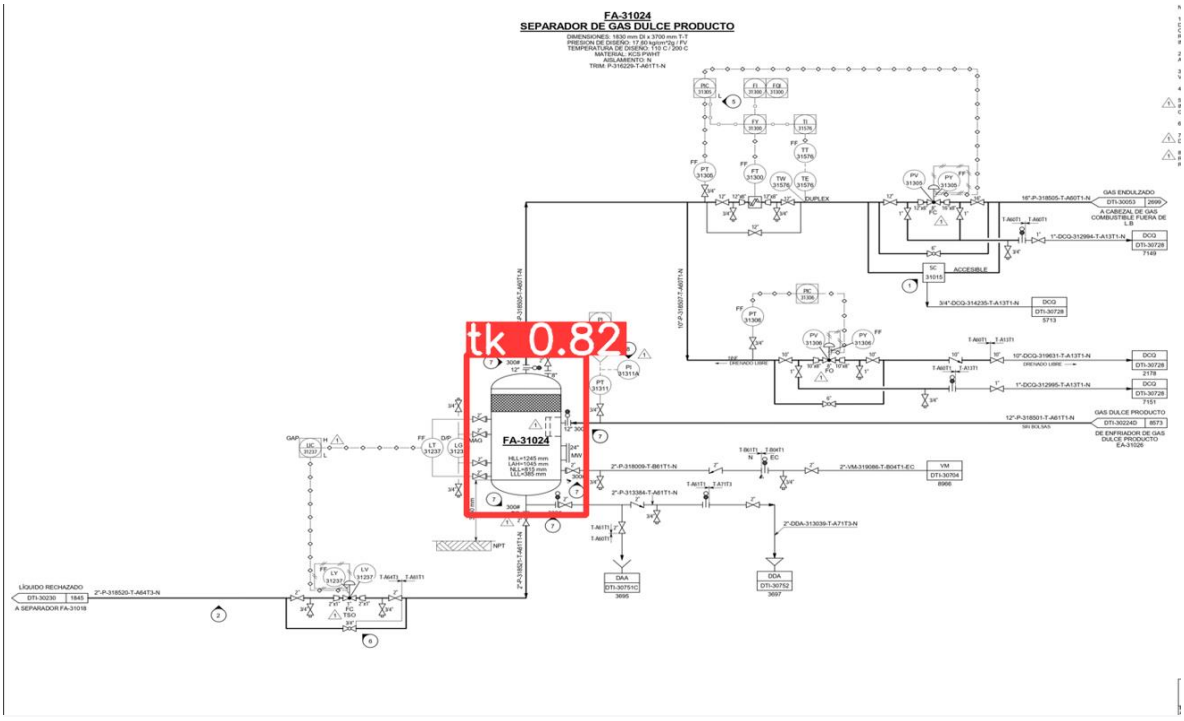


Figura 29. Detección de tanques variante 3.

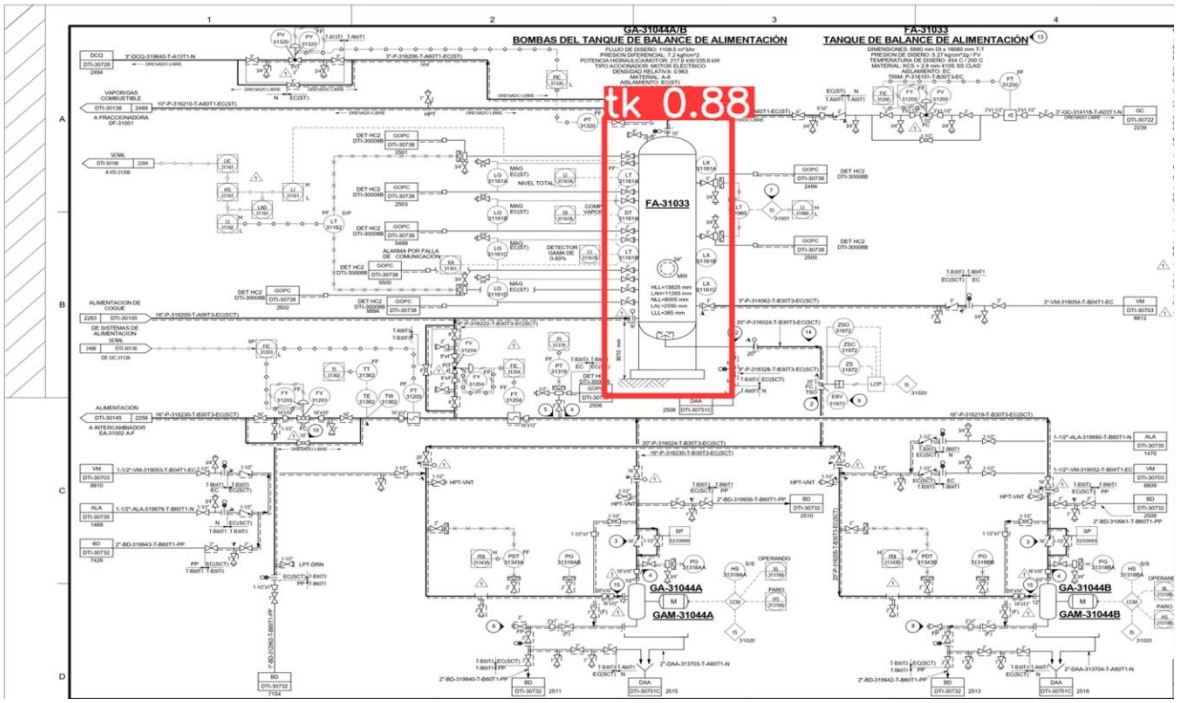


Figura 30. Detección de tanques variante 4.

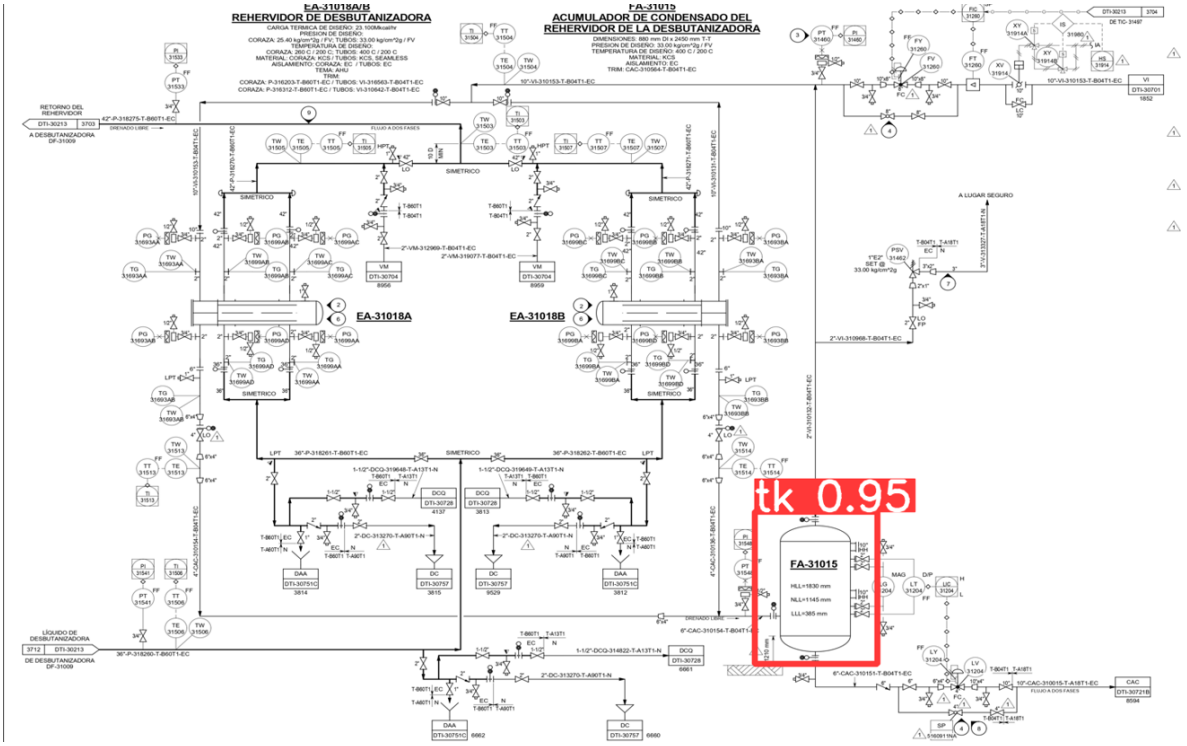


Figura 31. Detección de generadores de vapor.

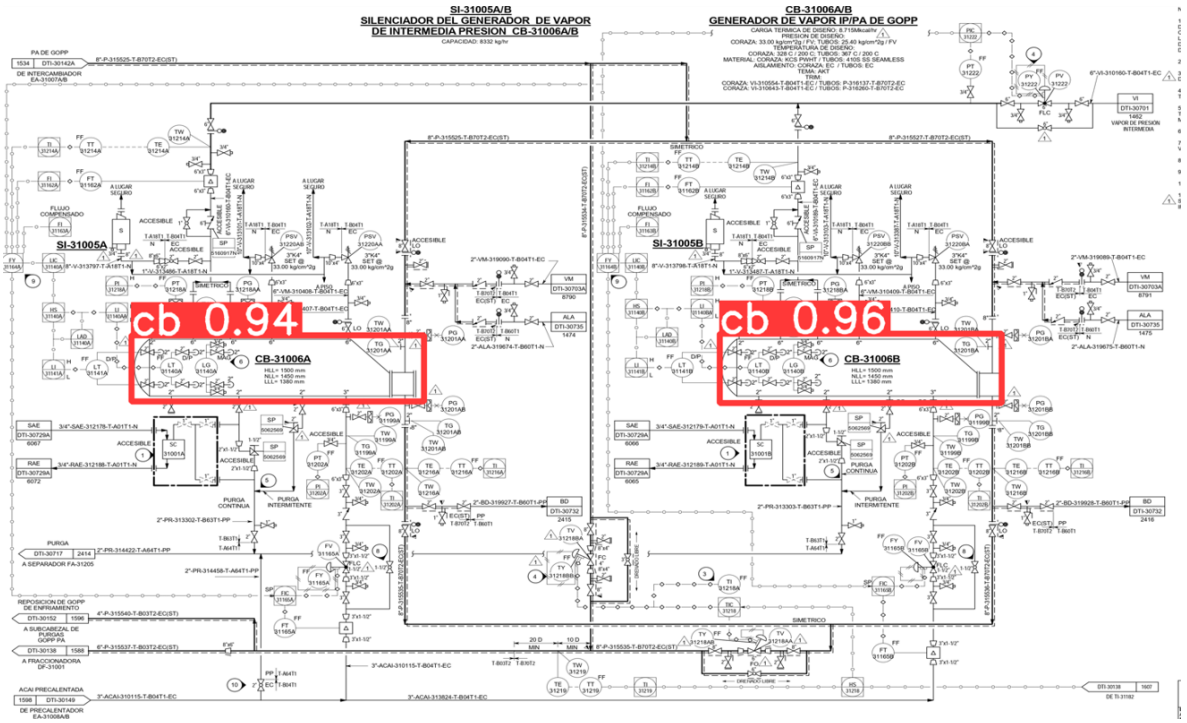
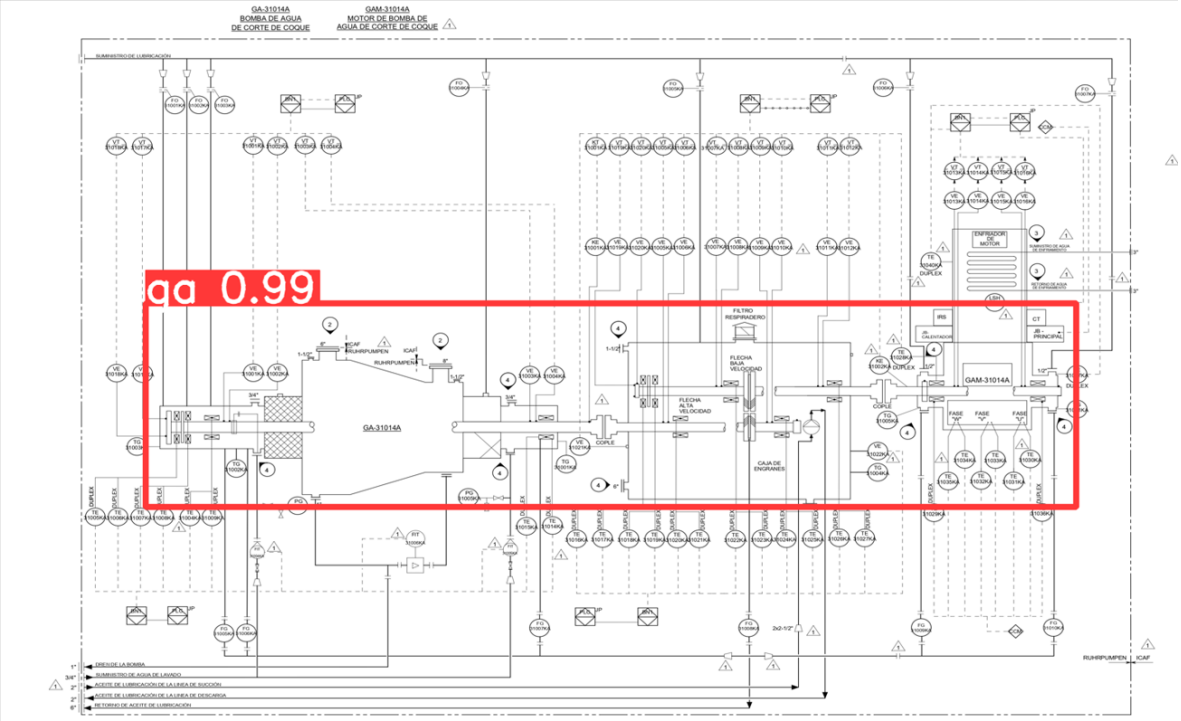


Figura 32. Detección de turbinas.



Estos son ejemplos de detecciones en los que se encuentran exactamente los componentes que se desean encontrar. Se han utilizado imágenes con el modelo para detectar los elementos específicos, y cada detección está asociada con un porcentaje de exactitud. Este porcentaje permite analizar la información proporcionada y realizar acciones adicionales, como el filtrado de información para obtener los mejores resultados posibles.

Sin embargo, también se han llevado a cabo pruebas adicionales para evaluar el comportamiento de los diferentes modelos frente a información con ruido. Esto ayuda a identificar cómo se comportan en situaciones más desafiantes y qué tan robustos son en la detección de elementos en presencia de interferencias.

Como era de esperar, se han encontrado casos en los que se producen fallos en la detección, hallando elementos donde no deberían estar presentes. A continuación, se muestran algunos ejemplos de estos casos para ilustrar dichos fallos:

Figura 33. Fallo en detección de hornos.

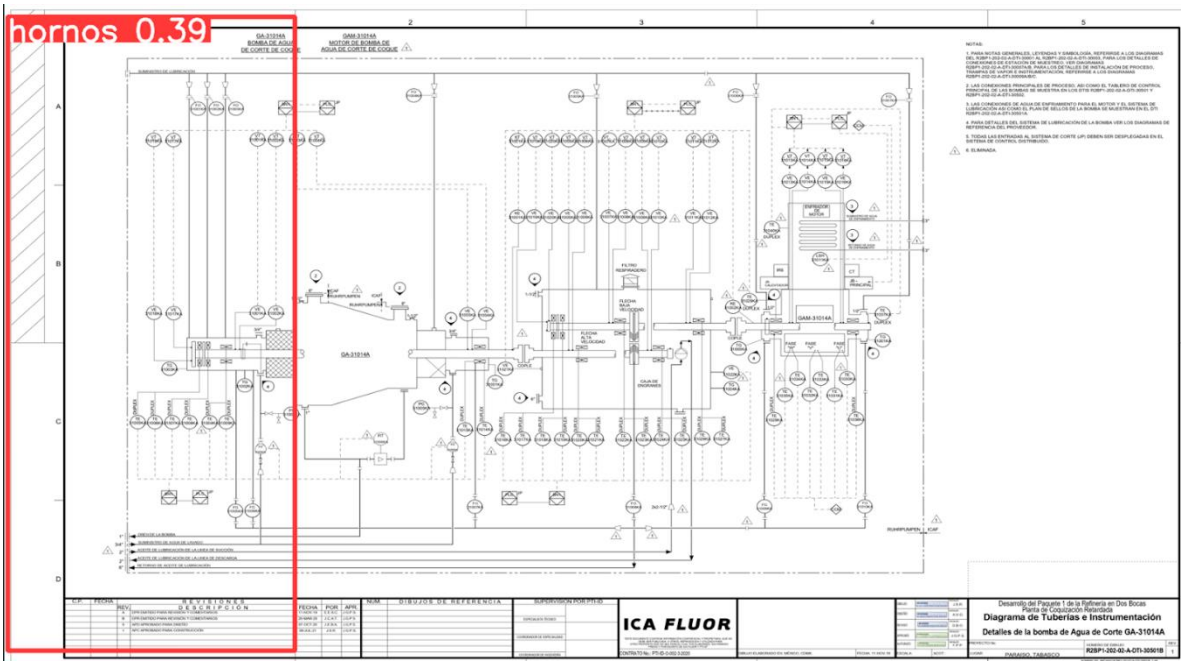


Figura 34. Fallo en detección de hornos 2.

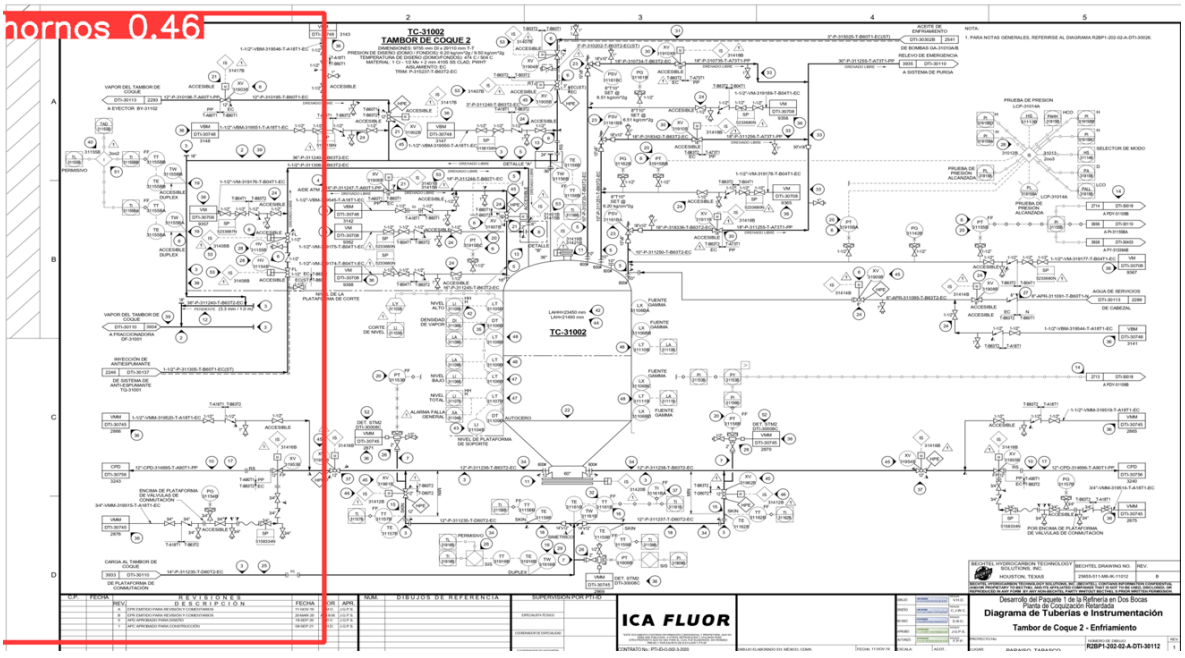


Figura 35. Fallo en detección de generadores de vapor.

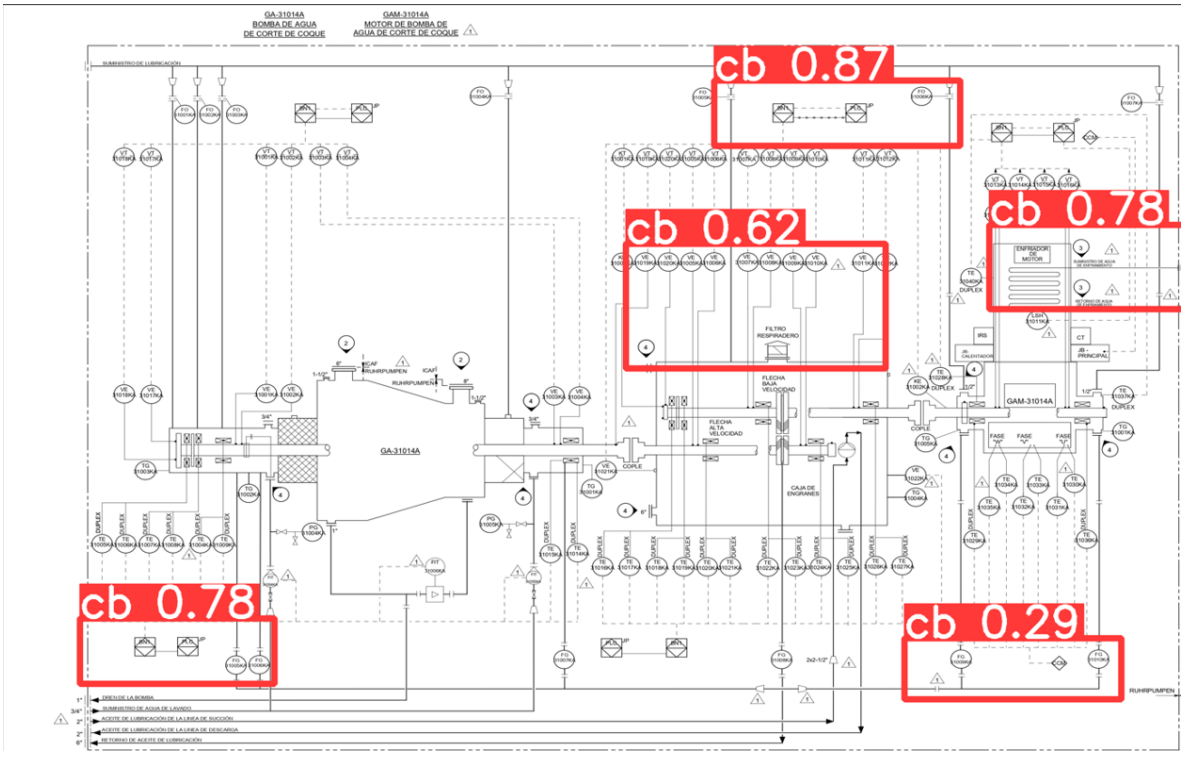
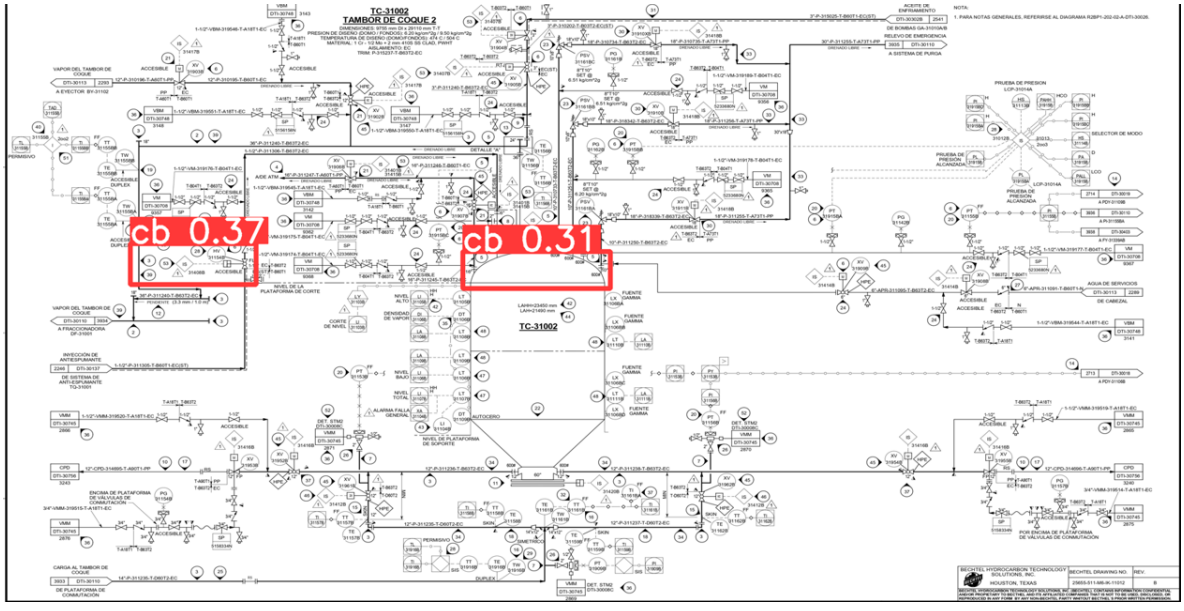


Figura 36. Fallo en detección de generadores de vapor 2.



Nota: Con el fin de evitar una saturación del documento con imágenes de errores, se han documentado un par de ejemplos de los modelos que presentan errores constantes, que son los

generadores de vapor y hornos. Sin embargo, es importante destacar que estos errores tienen un porcentaje de error bajo.

Durante la reunión con el asesor al finalizar esta semana, se llegó a la conclusión que en general, la detección de los elementos funciona de manera adecuada, aunque se identificaron algunas oportunidades de mejora. Sin embargo, se hizo una observación particular en relación con el elemento "línea de conexión". A pesar de ser un elemento estático, se determinó que no todas las líneas de conexión ilustradas en el P&ID deben ser parte del HMI. Por lo tanto, se propuso realizar un dibujo tentativo de las líneas de conexión dentro de la herramienta, siendo este el único elemento que requiere intervención humana para su representación en el HMI. Con esta evaluación positiva, se da paso a la siguiente etapa del proyecto, que consiste en el almacenamiento de la información.

3.2.2. Fase 2: Almacenar información

Semana 5: Creación de la base de datos

Se plantea la creación de una base de datos para almacenar las coordenadas de los elementos identificados. Esta medida busca optimizar la gestión de los elementos, ya que hasta el momento se han manejado de manera local. Con la implementación de esta base de datos, se podrá almacenar tanto las imágenes como las coordenadas, y así se podrán consultar en el momento que sea necesario. Esta mejora permitirá utilizar la herramienta directamente en la máquina virtual de la compañía, evitando así la necesidad de realizar procesos de instalación o configuración por parte de los diseñadores.

En cuanto a la creación de la base de datos, se han evaluado algunos candidatos, los cuales se detallan en la siguiente tabla:

Tabla 3. Cuadro comparativo administradores de bases de datos.

Herramienta	Descripción
phpMyAdmin	Herramienta de administración de bases de datos MySQL con una interfaz web.
Adminer	Alternativa ligera a phpMyAdmin con una interfaz simplificada.
MySQL Workbench	Herramienta oficial de MySQL que ofrece una interfaz gráfica para administrar bases de datos.
Navicat	Herramienta multiplataforma para la administración de bases de datos que admite MySQL, entre otros
HeidiSQL	Herramienta gratuita y de código abierto para administrar bases de datos MySQL, PostgreSQL y SQL Server.

En general, cada uno de estos administradores de bases de datos cumple la misma función. En este caso, se ha decidido utilizar phpMyAdmin debido a su amplia adopción y soporte en línea, además de ser considerada la herramienta más popular en su categoría. Entre sus principales atributos se encuentran:

- Amplia adopción y soporte en línea: phpMyAdmin cuenta con una gran comunidad de usuarios y desarrolladores que ofrecen soporte y recursos en línea.
- Interfaz web intuitiva y fácil de usar: La interfaz de phpMyAdmin es amigable y permite administrar bases de datos de manera sencilla, incluso para usuarios sin experiencia previa.
- Amplias funcionalidades para administrar bases de datos: phpMyAdmin ofrece diversas funcionalidades para administrar y gestionar bases de datos, como la creación de tablas, consultas SQL, importación y exportación de datos, entre otros.
- Gratuito y de código abierto: phpMyAdmin se distribuye de forma gratuita y su código fuente está disponible para que pueda ser modificado y adaptado según las necesidades del proyecto.

Crear bases de datos en phpmyadmin es muy sencillo, en la interfaz de phpMyAdmin **(Fig.37)**, se debe buscar y hacer clic en el enlace "Base de datos" en la parte superior de la pantalla. A continuación, se presentará un campo para ingresar el nombre de la nueva base de datos **(Fig.38)**, es recomendable elegir un nombre descriptivo y significativo que refleje el propósito de la base de datos, de ser necesario, se pueden establecer opciones adicionales para esta, como el conjunto de caracteres y la clasificación. Por lo general, las opciones predeterminadas son suficientes. Por último, se debe hacer clic en el botón de crear.

Figura 37. Interfaz principal phpmyadmin.

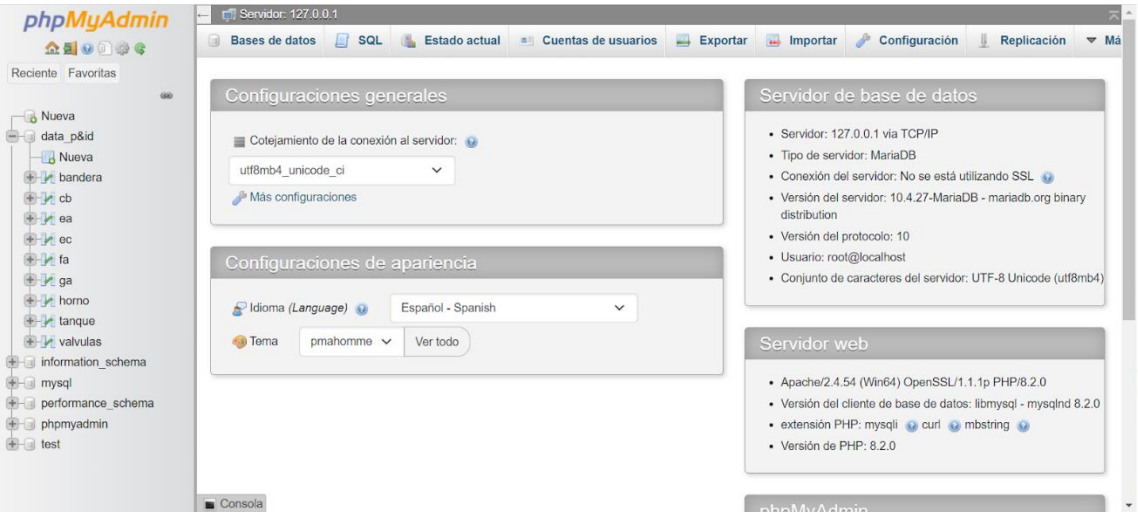
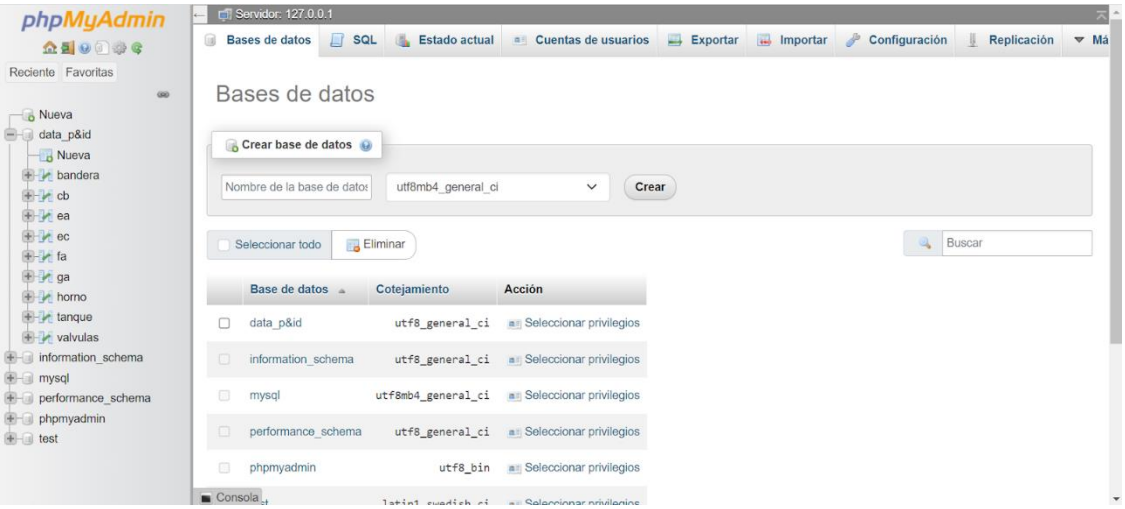


Figura 38. Interfaz phpmyadmin para la creación de nueva base de datos.



Una vez se ha creado la base de datos, es necesario crear las tablas correspondientes en su interior. En este caso, se creará una tabla por cada elemento que se ha identificado. Cada tabla contiene cinco columnas: Nombre, Xmin, Ymin, Xmax y Ymax. Estas columnas se utilizarán para almacenar las coordenadas de cada elemento identificado.

El proceso de creación de estas tablas se lleva a cabo mediante comandos SQL en la consola. Se ingresa la siguiente estructura, la cual varía según el tipo de elemento que se esté creando:

```
CREATE TABLE nombre_de_la_tabla (
    Name,
    Xmin,
    Ymin,
    Xmax,
    Ymax
);
```

En el caso anterior, no se especifican diferentes tipos de datos para cada columna, por defecto, estas se crearán con el tipo de dato VARCHAR. En esta etapa del proyecto, no se están realizando procesos específicos con estos datos, por lo que es apropiado almacenar toda esta información de esta manera.

En la siguiente figura se pueden apreciar todas las tablas que fueron creadas para llevar a cabo el proyecto, como se menciona anteriormente cada elemento tiene que estar asociado a una tabla, en total se crearon nueve tablas.

Figura 39. Tablas creadas en phpMyAdmin.

	Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	bandera	Examinar Estructura Buscar Insertar Vaciar Eliminar	95	InnoDB	utf8_general_ci	16.0 KB	-
☐	cb	Examinar Estructura Buscar Insertar Vaciar Eliminar	10	InnoDB	utf8_general_ci	16.0 KB	-
☐	ea	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_general_ci	16.0 KB	-
☐	ec	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_general_ci	16.0 KB	-
☐	fa	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8_general_ci	16.0 KB	-
☐	ga	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8_general_ci	16.0 KB	-
☐	horno	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8_general_ci	16.0 KB	-
☐	tanque	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8_general_ci	16.0 KB	-
☐	valvulas	Examinar Estructura Buscar Insertar Vaciar Eliminar	87	InnoDB	utf8_general_ci	16.0 KB	-

Al finalizar esta semana se valida la creación de la base de datos por parte del asesor sin complicación alguna.

Semana 6: Almacenar todos los elementos identificados

En primer lugar, el código establece una conexión con la base de datos MySQL, mediante el método mysql.connector. Se especifican los detalles necesarios para conectarse al servidor de la base de datos, como el nombre de usuario, la contraseña y el nombre de la base de datos en la que se trabajarán los datos.

Figura 40. Conexión a la base de datos desde Python.

```
# Conectamos con la base de datos
mydb = mysql.connector.connect(
    host="localhost",
    Suser="root",
    password="",
    database="data_p&id"
)
```

Es en este punto donde se integra el código previamente desarrollado para la identificación de elementos. Al pasar una imagen al modelo entrenado, este genera otra imagen con los elementos detectados y una matriz de datos que incluye las coordenadas de dichos componentes. Esta matriz se convierte en un DataFrame, una estructura de datos tabular similar a una hoja de cálculo, utilizando la biblioteca pandas.

Es crucial realizar un filtrado en los elementos detectados, ya que algunas de estas pueden tener un bajo porcentaje de exactitud y generar ruido. En secciones anteriores se ha observado que un umbral de precisión del 90% es adecuado para la mayoría de los elementos, pero algunos requieren un umbral más alto, como el 95%. Para este fin, se realiza un filtrado en la variable "datos" y se crea un nuevo DataFrame llamado "menos". Posteriormente, se realiza otro proceso de filtrado en "menos" para obtener solo las columnas de interés para este proyecto: "Name", "Xmin", "Ymin", "Xmax" y "Ymax". Este filtrado genera una nueva variable llamada "coordenadas".

Por último, se transforma la variable "coordenadas" en un nuevo DataFrame llamado "df", que se utilizará para el almacenamiento de la información en la base de datos. Es importante destacar que este DataFrame no contiene encabezados ni otras configuraciones adicionales en la tabla o los datos, lo cual es relevante al momento de migrar la información a la DB.

Figura 41. Filtrado de datos obtenidos en la detección de elementos.

```
# saco datos del modelo
datos = detect.pandas().xyxy[0]
# filtrar el dataframe
menos = datos[datos['confidence'] > 0.90]
coordenadas = menos[['name', 'xmin', 'ymin', 'xmax', 'ymax']].copy()
df = pd.DataFrame(coordenadas)
```

Luego, se crea un "motor de base de datos” con el método create_engine, utilizando una biblioteca llamada SQLAlchemy. Este motor actúa como una interfaz que nos permite interactuar con la base de datos de manera más sencilla y eficiente, para su funcionamiento se le debe suministrar la información del nombre de la base de datos a la cual hará conexión. A continuación, se agrega el DataFrame “df” generado en la detección de los elementos a una tabla específica en la base de datos. El método to_sql se utiliza para transferir los datos del DataFrame a la tabla de la base de datos.

Después de completar la operación de escritura, se cierra la conexión con la base de datos para liberar los recursos y asegurarse de que la conexión se cierre correctamente. Esto es importante para evitar problemas en futuras interacciones con la base de datos.

Figura 42. Almacenamiento de la información filtrada a una tabla específica.

```
# Creamos el motor de la base de datos
engine = create_engine('mysql+mysqlconnector://root:@localhost/data_p&id')

# Agregamos los datos a la tabla
df.to_sql(name='tanque', con=engine, if_exists='append', index=False)

# Cerramos la conexión con la base de datos
mydb.close()
```

Nota: Este proceso se repite para cada elemento detectado y su tabla correspondiente. Es decir, se aplica una función específica para cada elemento, donde se realiza la detección, se procesan los datos obtenidos y se almacenan en la base de datos.

Cada elemento tiene su propia función que sigue el mismo flujo: se pasa la imagen del elemento al modelo entrenado, se generan las detecciones y se extraen las coordenadas. Luego se realiza un filtrado de las detecciones para eliminar aquellas con baja precisión y se obtiene un DataFrame con las coordenadas relevantes.

Posteriormente, se transforma ese DataFrame en una estructura adecuada para ser almacenada en la base de datos, siguiendo el formato establecido para cada tabla. Esto implica asegurarse de que la información este correctamente estructurados y se ajusten a los tipos de datos necesarios en cada columna.

Al finalizar la semana se verifica junto con el asesor que los datos carguen de manera adecuada para cada tabla en la base de datos como se puede observar a continuación:

Figura 43. Datos almacenados en la tabla bandera.

name	xmin	ymin	xmax	ymax
bandera	106.6724853515625	492.59930419921875	211.7313995361328	511.13671875
bandera	1421.8980712890625	270.3255615234375	1525.3612060546875	291.2830505371094
bandera	490.0267028808594	898.9354858398438	588.8820190429688	920.4033203125
bandera	870.3887939453125	899.4737548828125	967.1891479492188	920.4523315429688
bandera	108.38709259033203	413.61376953125	204.51255798339844	436.54791259765625
bandera	103.48478698730469	52.19937515258789	207.0579071044922	74.56572723388672
bandera	1421.986083984375	670.4750366210938	1524.4085693359375	693.3348388671875

Figura 44. Datos almacenados en la tabla cb.

name	xmin	ymin	xmax	ymax
cb	978.8770751953125	457.9090881347656	1324.408203125	542.78369140625
cb	256.83380126953125	459.1315002441406	615.64990234375	540.7003784179688
cb	978.8770751953125	457.9090881347656	1324.408203125	542.78369140625
cb	256.83380126953125	459.1315002441406	615.64990234375	540.7003784179688
cb	978.8770751953125	457.9090881347656	1324.408203125	542.78369140625
cb	256.83380126953125	459.1315002441406	615.64990234375	540.7003784179688

Figura 45. Datos almacenados en la tabla ea.

name	xmin	ymin	xmax	ymax
ea	425.121826171875	484.3807067871094	610.9076538085938	538.353759765625
ea	426.17095947265625	622.3262329101562	612.1300048828125	677.8661499023438
ea	649.426513671875	578.552978515625	837.734619140625	630.412841796875
ea	500.0832824707031	439.3915100097656	708.44677734375	492.79266357421875
ea	500.0832824707031	439.3915100097656	708.44677734375	492.79266357421875
ea	809.4255981445312	524.4003295898438	1018.4035034179688	581.748779296875
ea	1163.2835693359375	569.0249633789062	1336.922607421875	619.7649536132812
ea	624.970458984375	570.8785400390625	807.8512573242188	625.004150390625

Figura 46. Datos almacenados en la tabla ec.

name	xmin	ymin	xmax	ymax
ec	718.8433227539062	582.7379760742188	1013.6466674804688	704.1940307617188
ec	720.6090698242188	211.3006134033203	1020.3184814453125	332.5035705566406
ec	551.533447265625	449.6181335449219	817.3665161132812	581.863037109375
ec	537.5211181640625	146.3538360595703	829.462158203125	265.9581298828125
ec	551.533447265625	449.6181335449219	817.3665161132812	581.863037109375
ec	537.5211181640625	146.3538360595703	829.462158203125	265.9581298828125
ec	763.8880004882812	613.57568359375	1031.823974609375	738.6039428710938

Figura 47. Datos almacenados en la tabla fa.

name	xmin	ymin	xmax	ymax
fa	697.142822265625	172.8973846435547	1160.3487548828125	460.622802734375
fa	373.1791687011719	150.02586364746094	832.917724609375	431.1761474609375
fa	362.5933837890625	310.86834716796875	816.7728881835938	581.21533203125
fa	378.2257080078125	531.3949584960938	823.3939819335938	803.2628173828125
fa	391.137939453125	552.3938598632812	818.5534057617188	699.2652587890625
fa	469.7227783203125	657.0199584960938	1254.081787109375	1041.073974609375
fa	484.5075988769531	680.5728149414062	1221.807373046875	881.110107421875

Figura 48. Datos almacenados en la tabla ga.

name	xmin	ymin	xmax	ymax
ga	274.626708984375	407.8306579589844	1403.4012451171875	654.4667358398438
ga	202.2803192138672	444.8403015136719	1366.014892578125	688.3902587890625
ga	274.626708984375	407.8306579589844	1403.4012451171875	654.4667358398438

Figura 49. Datos almacenados en la tabla horno.

name	xmin	ymin	xmax	ymax
hornos	453.5921630859375	87.70511627197266	809.7903442382812	677.0758666992188
hornos	451.7110595703125	86.1723861694336	807.9635009765625	675.355224609375
hornos	453.3350830078125	86.64646911621094	804.3270263671875	676.26025390625
hornos	453.2021484375	88.91905975341797	809.2099609375	675.9397583007812
hornos	453.5263366699219	87.46886444091797	811.5789794921875	674.23779296875
hornos	453.89990234375	88.08517456054688	807.9248657226562	675.3673095703125

Figura 50. Datos almacenados en la tabla tanque.

name	xmin	ymin	xmax	ymax
tk	1014.548095703125	569.861572265625	1166.272705078125	836.46337890625
tk	607.41064453125	121.94263458251953	786.6167602539062	375.5318298339844
tk	640.2698974609375	54.14128875732422	869.6229858398438	599.3906860351562
tk	660.5225830078125	81.0963134765625	878.1192626953125	927.2825927734375
tk	796.1649169921875	173.88909912109375	963.3812255859375	537.3826904296875
tk	498.07891845703125	75.58186340332031	689.5775756835938	926.176025390625
tk	640.2698974609375	54.14128875732422	869.6229858398438	599.3906860351562

Figura 51. Datos almacenados en la tabla válvulas.

name	xmin	ymin	xmax	ymax
valves	762.366455078125	656.7382202148438	797.6387939453125	692.4596557617188
valves	764.3021850585938	256.3220520019531	794.7994995117188	288.1086730957031
valves	1208.5079345703125	382.89691162109375	1234.881591796875	420.54998779296875
valves	1204.6864013671875	785.4715576171875	1231.300048828125	821.6046752929688
valves	1361.8602294921875	149.2708740234375	1383.3663330078125	178.3659210205078
valves	1361.8602294921875	149.2708740234375	1383.3663330078125	178.3659210205078
valves	762.366455078125	656.7382202148438	797.6387939453125	692.4596557617188

3.2.3. Fase 3: Dibujo vectorial

Inicialmente, se había dividido el trabajo en tres actividades principales, que incluían la conexión con la base de datos para realizar el dibujo de los elementos detectados. Sin embargo, al comenzar el desarrollo del algoritmo, se observó que sería más conveniente realizar el dibujo de manera local, es decir, al identificar un elemento, dibujarlo y luego subir las coordenadas correspondientes. Esto se debe a que resultaría ineficiente detectar, procesar y subir los datos a la base de datos para luego tener que consultarlos minutos después, cuando ya se habían obtenido localmente.

Por lo tanto, se propone una actividad única, que consiste en implementar un algoritmo para dibujar los elementos identificados y adaptarlos a los procesos previos. Se asignan cuatro semanas para llevar a cabo esta implementación, con el objetivo de optimizar el proceso y evitar redundancias en la transferencia de información entre la detección y el dibujo de los elementos. Esta nueva estrategia permitirá ahorrar recursos y simplificar el flujo de trabajo, al realizar el dibujo de los elementos de manera inmediata al momento de la detección. De esta manera, se reducirán los tiempos de respuesta y se optimizará el uso de la herramienta desarrollada.

Semana 7: Dibujos vectoriales en formato SVG

El primer paso antes de desarrollar el algoritmo es definir los dibujos vectoriales de los elementos. Estos dibujos serán creados de manera manual y posteriormente el algoritmo deberá ubicarlos en la posición y tamaño correspondientes dentro de la plantilla HMI. Es importante destacar que el dibujo de cada elemento está estrechamente vinculado con el dibujo original en el P&ID, ya que así lo requiere el proyecto.

En este caso, se ha optado por utilizar la herramienta Figma para realizar los dibujos vectoriales. Esta elección se debe a que la empresa utiliza esta herramienta para sus mockups y demos, y es ideal utilizar al máximo las herramientas comunes en los procesos de la compañía. Con Figma, se pueden crear dibujos precisos y de alta calidad, que luego serán utilizados por el

algoritmo para ubicar los elementos en la plantilla HMI. Este paso es fundamental para garantizar una representación visual adecuada de los elementos dentro del HMI, y asegurarse de que estén alineados con los diseños originales en el P&ID. A continuación, se presentan los elementos diseñados en Figma junto a sus representaciones en el P&ID:

Figura 52. Representación SVG vs P&ID de las entradas y salidas (Banderas).

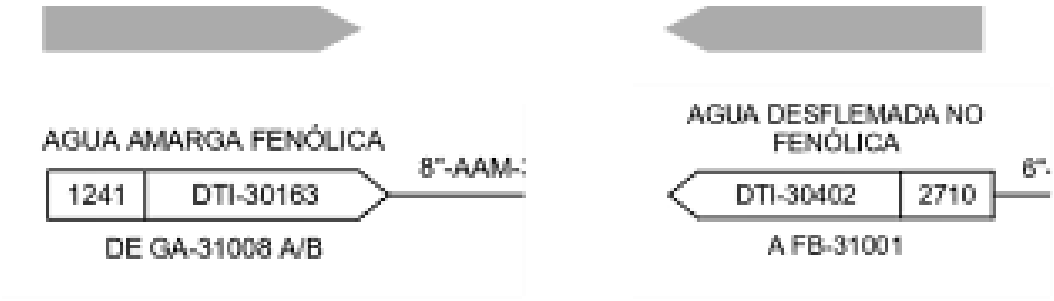


Figura 53. Representación SVG vs P&ID de generadores de vapor.

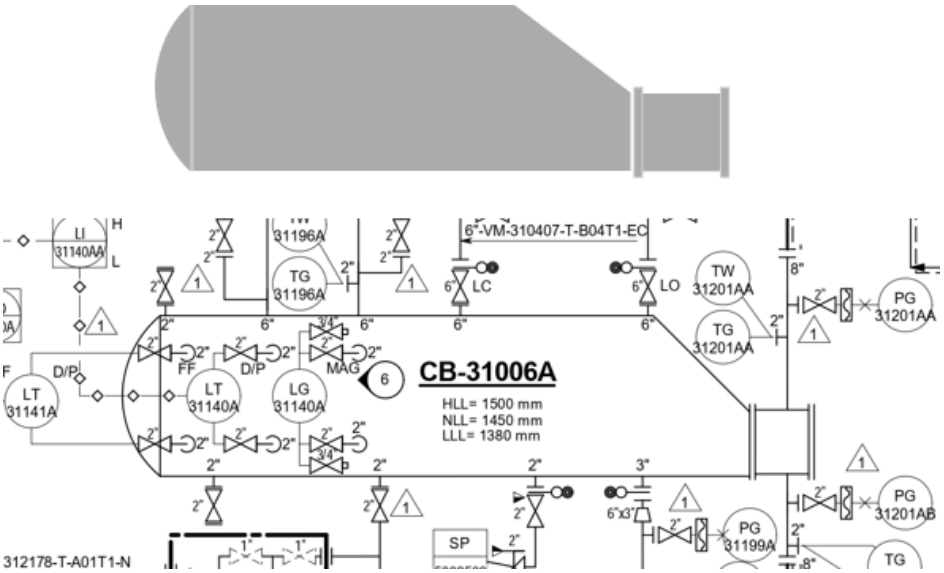


Figura 54. Representación SVG vs P&ID del horno.

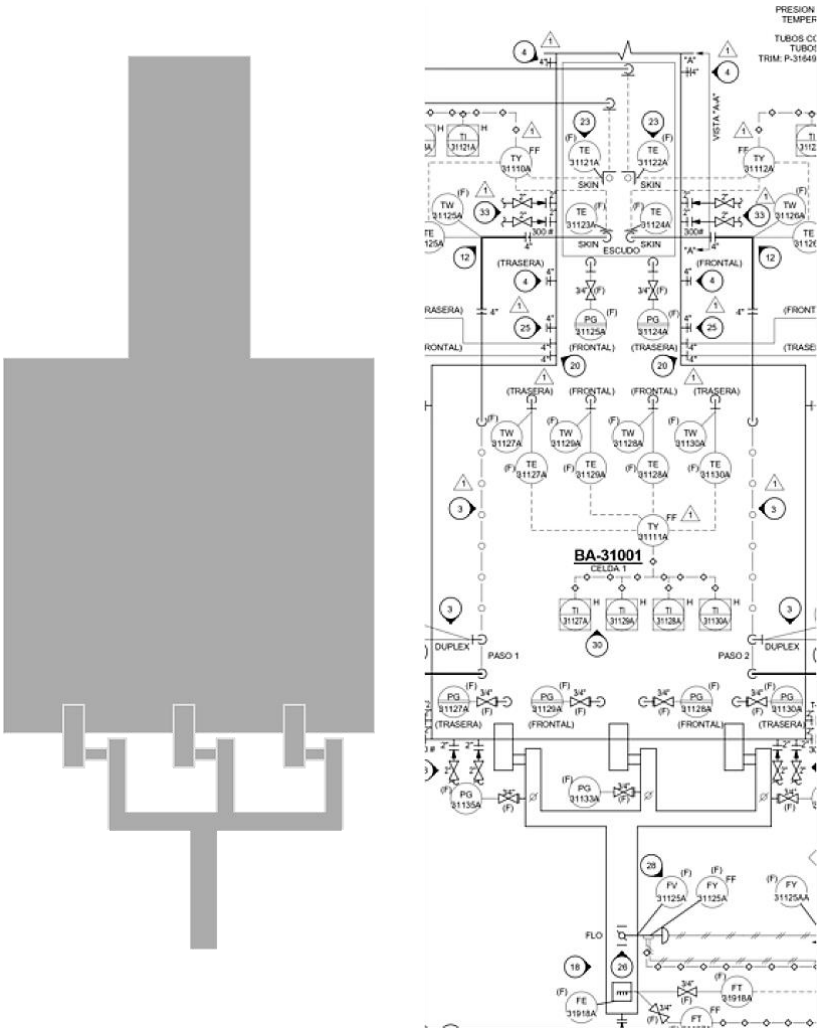
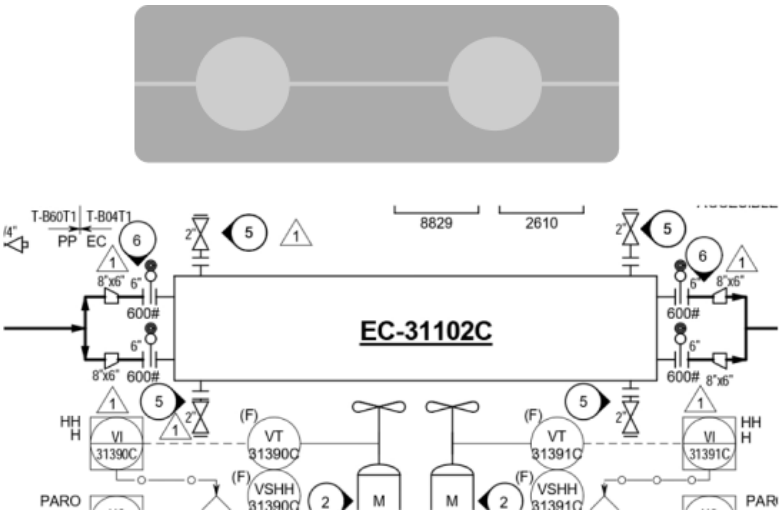


Figura 55. Representación SVG vs P&ID de sistema de ventilación.



Nota: Es importante señalar que la representación del sistema de ventilación en la interfaz gráfica es diferente a la que se muestra en el diagrama P&ID. Esta variación en la representación visual ha sido solicitada específicamente por el cliente del proyecto. Todas las variantes del sistema de ventilación se representarán de acuerdo con esta nueva representación visual acordada con el cliente.

Figura 56. Representación SVG vs P&ID de turbinas.

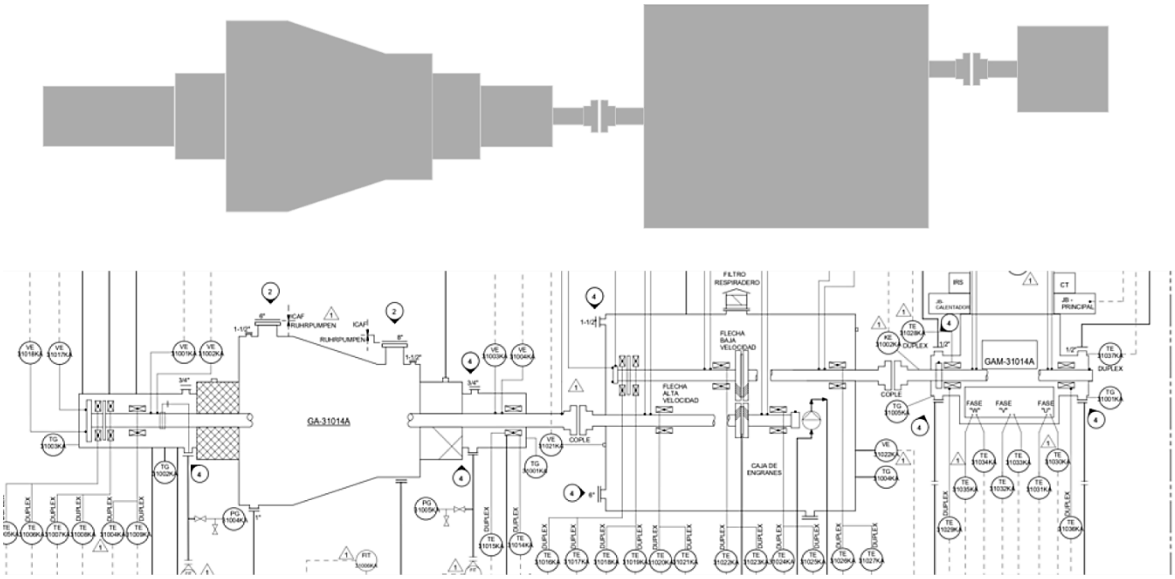


Figura 57. Representación SVG vs P&ID de intercambiador de calor.

Figura 58. Representación SVG vs P&ID de separador de crudo variante 1.

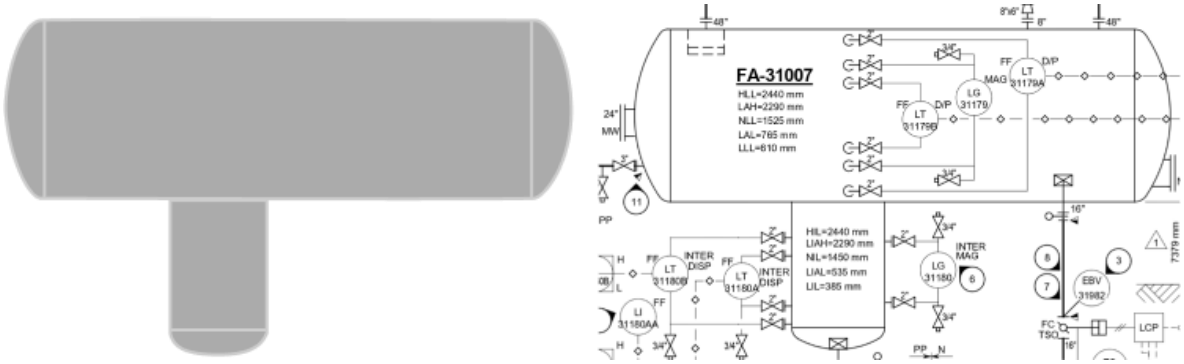


Figura 59. Representación SVG vs P&ID de separador de crudo variante 2.

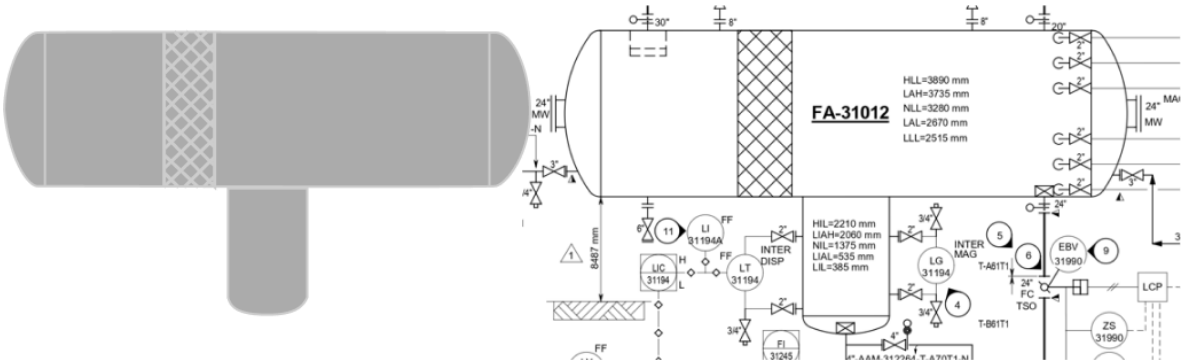


Figura 60. Representación SVG vs P&ID de separador de crudo variante 3.

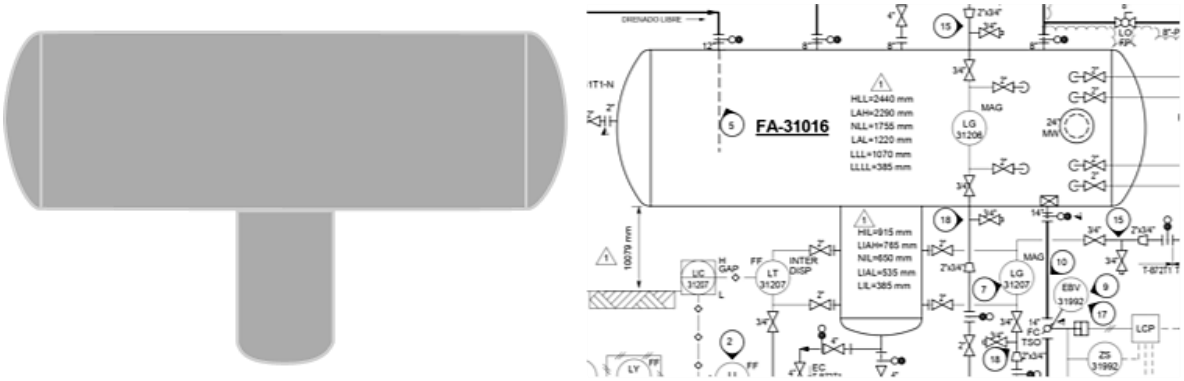


Figura 61. Representación SVG vs P&ID de separador de crudo variante 4.

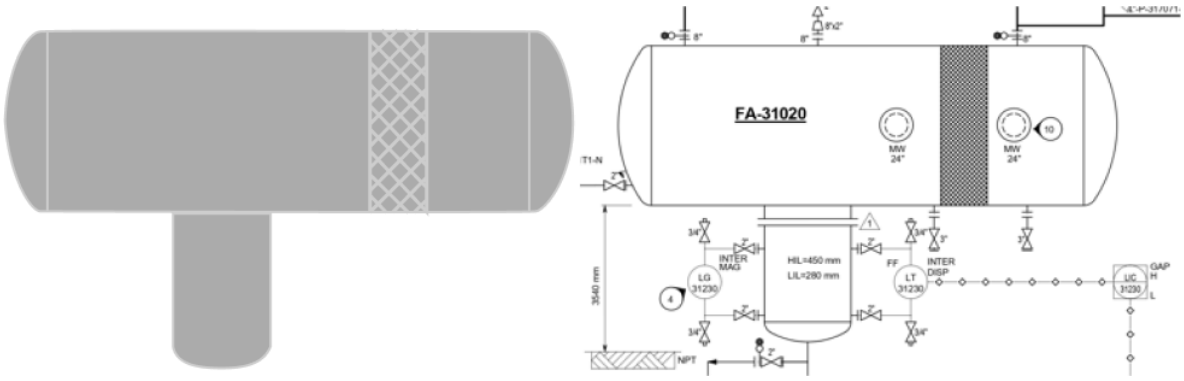


Figura 62. Representación SVG vs P&ID de tanque variante 1.

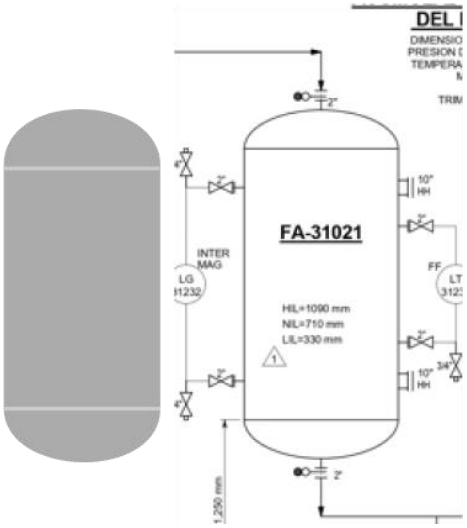


Figura 63. Representación SVG vs P&ID de tanque variante 2.

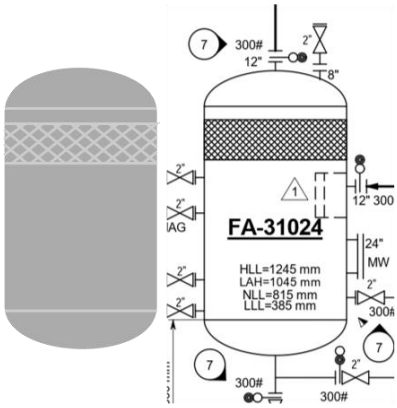


Figura 64. Representación SVG vs P&ID de tanque variante 3.

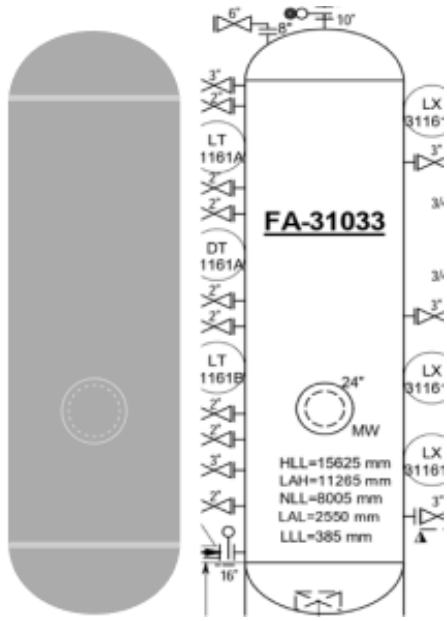


Figura 65. Representación SVG vs P&ID de tanque variante 4.

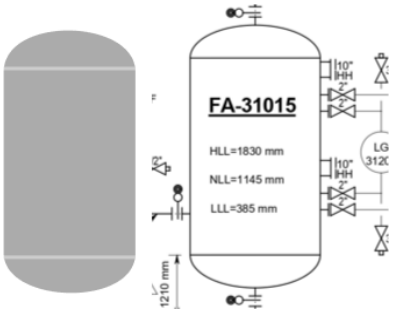
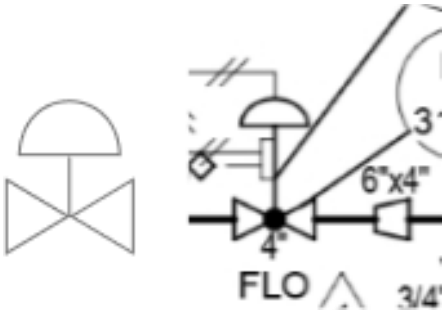


Figura 66. Representación SVG vs P&ID de válvulas.



Nota: Es importante recalcar que no se incluyen los nombres de los elementos detectados. Esto se debe a que la identificación y superposición de texto se realiza en otra parte del algoritmo, después de que los elementos hayan sido identificados y dibujados en la interfaz gráfica. Una vez que se haya completado el proceso de dibujo, se añadirá el texto correspondiente a cada elemento para proporcionar una representación completa y legible de los elementos en la plantilla HMI.

Una vez que los diseños han sido completados, se procede a integrarlos en el algoritmo en desarrollo. Los diseños se implementan como objetos de tipo "path" en el formato SVG. El formato SVG representa las imágenes como código, almacenando propiedades como forma, color y tamaño. Esto permite agregar los diseños al algoritmo sin perder calidad y facilita la generación de imágenes editables posteriormente.

En particular, se inicia el proceso de integración con las entradas y salidas, que, aunque parecen elementos simples, presentan desafíos. La dificultad radica en posicionar correctamente estos elementos y lograr una interpretación precisa. Esto se debe a que las entradas y salidas no tienen una diferencia gráfica clara, su única distinción es la orientación. Una entrada puede estar en la parte derecha o izquierda de la página, y lo mismo aplica para las salidas.

El código de detección desarrollado no distingue los cambios de orientación en las banderas. Se intentó entrenar el modelo para reconocer estos cambios, pero resultó innecesario. El comportamiento de una bandera está definido por el texto que la acompaña, no por su dirección. Por ejemplo, una bandera con el texto "A DF-12345" siempre se identificará como una salida, sin importar su orientación. De manera similar, una bandera con el texto "DE EA-1234" siempre se interpretará como una entrada, independientemente de su orientación. A continuación, se muestran ejemplos gráficos para ilustrar estos casos:

Figura 67. Bandera de entrada variante 1.

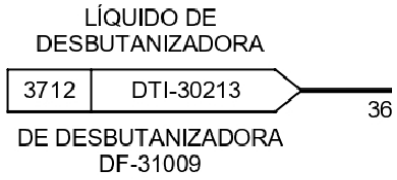


Figura 68. Bandera de entrada variante 2.

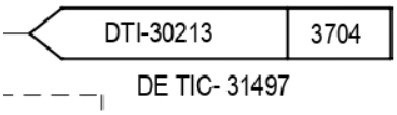


Figura 69. Bandera de salida variante 1.

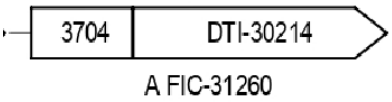
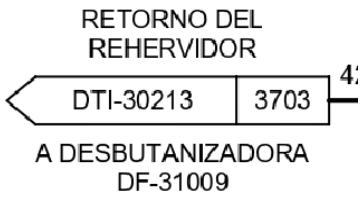


Figura 70. Bandera de salida variante 2.



El algoritmo desarrollado hace un procesamiento de las coordenadas de los elementos detectados y realiza el dibujo correspondiente utilizando la biblioteca svgwrite en Python, además con la ayuda de la biblioteca pytesseract se identifica el texto de los elementos.

Inicia definiendo la variable ruta que almacena la ruta de un archivo de imagen en formato JPG, se definen las dimensiones del lienzo del archivo SVG con las variables width y height, estableciendo el ancho en 1920 y la altura en 1080 píxeles. Se crea un objeto Drawing de

svgwrite llamado dwg, que representa el archivo SVG. Luego, se utiliza el método rect para dibujar un cuadrado en el lienzo. Se especifican las coordenadas de la esquina superior izquierda y la esquina inferior derecha del cuadrado utilizando las coordenadas (0, 0) para la esquina superior izquierda y (width, height) para la esquina inferior derecha. También se establece el color de relleno del cuadrado en "#CCCCCC" este color es un gris claro requisito del cliente y que cumple con las recomendaciones de la norma ANSI ISA 101. Finalmente, se agrega el cuadrado al objeto dwg utilizando el método add. Esto inserta el cuadrado en el lienzo del archivo SVG, el archivo SVG resultante se guarda con el nombre "imagen_f.svg" **(Fig.71)**, posteriormente se realiza el proceso de detección mencionado a lo largo de documento y una vez generado los datos del modelo se procede a filtrar las coordenadas de los elementos con una confianza superior al 50% y se almacenan en el DataFrame 'menos'. Después, se realiza una copia de las columnas de coordenadas ('xmin', 'ymin', 'xmax', 'ymax') y se convierten a tipo entero.

Luego, se crea un array de numpy 'final' a partir de las coordenadas y se convierte a una lista 'final2'. A continuación, se obtienen las dimensiones de filas y columnas de 'final' para su iteración posterior. **(Fig.72)**

Posteriormente, se itera sobre la lista de coordenadas en 'final2'. Para cada posición, se extraen las coordenadas 'x', 'y', 'x1' y 'y1'. Si la diferencia entre 'x1' y 'x' es mayor a 85, se procede a dibujar un rectángulo con las coordenadas específicas. Luego, se recorta una imagen 'recorte' utilizando las coordenadas del rectángulo y se utiliza el módulo de reconocimiento óptico de caracteres Tesseract ('tess') para extraer el texto de la imagen recortada. **(Fig.73)**

A continuación, se realiza una serie de condiciones basadas en el valor del texto extraído, tomando el primer carácter y evaluando su valor, si es “A” es definido como salida y si es “D” es considerada una entrada. Dependiendo del texto, se selecciona una ruta de dibujo 'path_data' que varía para cada condición según la orientación y posición del texto en el p&id, luego se crea un objeto 'path' con dicha ruta y un color de relleno determinado. Posteriormente, se lleva a cabo

una reubicación y se agrega el objeto de ruta al dibujo. Finalmente, se agrega un objeto de texto 'texto' con variables previamente definidas para tipo, tamaño y color de fuente, con el texto extraído en la posición correspondiente al dibujo. **(Fig.74)**

Figura 71. Creación del lienzo de trabajo SVG.

```
ruta = r"C:/Users/Camil/Documents/Pasantia" \
       r"/pythonProject/train/ilovepdf_merged_page-0087.jpg"
# Defino las dimensiones del lienzo
width, height = 1920, 1080
# Crear el archivo SVG y el grupo principal
dwg = svgwrite.Drawing('imagen_f.svg', size=(width, height))
# Dibujar el cuadrado
square = dwg.rect((0, 0), (width, height), fill='#CCCCCC')
dwg.add(square)
```

Figura 72. Adecuación de los datos suministrados por el DataFrame.

```
#filtrar el dataframe
menos = datos[datos['confidence'] > 0.50]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
```

Figura 73. Detección de texto Pytesseract.

```
for pos in final2:
    x, y, x1,y1= pos[:4]
    if x1-x > 85:
        # dibujar un rectángulo con coordenadas específicas
        x0, y0, xf, yf = x*2, (y*2)+34, x1*2, (y1*2)+40
        recorte = img_pil.crop((x0, y0, xf, yf))
        txt = tess.image_to_string(recorte)
```

Figura 74. Dibujo de las banderas de acuerdo con los condicionales.

```
        if txt:
            txt = txt
        else:
            txt = ' '
        if txt[0] == 'A' and x > 800:
            path_data = 'M0.569427 0.919189H82.5854L95.5694 7.73047L82.5854 14.5417H0.569427V0.919189Z'
            path = dwg.path(d=path_data, fill='#ABABAB')
            path.translate(1790, y)
            dwg.add(path)
            texto = dwg.text(txt[1:], insert=(1790, y), fill=fill_i, font_size=font_size_i, font_family=font_i)
            dwg.add(texto)
        if txt[0] == 'A' and x < 800:
            path_data = 'M95 13.6226H12.9841L-1.90657e-08 6.81128L12.9841 0H95V13.6226Z'
            path = dwg.path(d=path_data, fill='#ABABAB')
            path.translate(30, y)
            dwg.add(path)
            texto = dwg.text(txt[2:], insert=(30+15, y), fill=fill_i, font_size=font_size_i, font_family=font_i)
            dwg.add(texto)
        if txt[0] == 'D' and x > 800:
            path_data = 'M95 13.6226H12.9841L-1.90657e-08 6.81128L12.9841 0H95V13.6226Z'
            path = dwg.path(d=path_data, fill='#ABABAB')
            path.translate(1790, y)
            dwg.add(path)
            texto = dwg.text(txt[2:], insert=(1790+15, y), fill=fill_i, font_size=font_size_i, font_family=font_i)
            dwg.add(texto)
        if txt[0] == 'D' and x < 800:
            path_data = 'M0.569427 0.919189H82.5854L95.5694 7.73047L82.5854 14.5417H0.569427V0.919189Z'
            path = dwg.path(d=path_data, fill='#ABABAB')
            path.translate(30, y)
            dwg.add(path)
            texto = dwg.text(txt[2:], insert=(30, y), fill=fill_i, font_size=font_size_i, font_family=font_i)
            dwg.add(texto)
```

El resultado de este código genera una imagen SVG con todas la banderas identificadas **(Fig.77)** y acatando la peticiones del cliente tanto como las recomendaciones de la norma ISA 101 para diseño de High Performance HMI, donde se sugiere en cuanto a colores: utilizar un color de fondo insaturado o neutro, como un gris claro, esto se debe a que los colores intensos pueden generar distorsiones cromáticas y dificultar la legibilidad de la información mostrada, además, es importante asegurarse que exista suficiente contraste entre los colores del primer plano (como el texto o los elementos gráficos) y el fondo. Esto garantiza que la información sea claramente visible y comprensible. En cuanto a la tipografía, la legibilidad del texto es un aspecto clave en el diseño de pantallas de HMI. Para garantizar la legibilidad, se siguieron las siguientes pautas:

- Utilizar fuentes claras y fáciles de leer: fuentes que sean legibles, evitando estilos extravagantes o con adornos excesivos. Fuentes como Arial, Helvetica, Verdana o Roboto son ejemplos de opciones comunes.

- Tamaño de letra adecuado: Asegurarse de que el tamaño de la fuente sea lo suficientemente grande para que pueda ser leído cómodamente por los operadores. Considerar el entorno en el que se utilizará el HMI y el tamaño de la pantalla para determinar el tamaño adecuado de la fuente.

En el caso actual se utilizó como fuente Arial, tamaño 10 y de color gris oscuro.

Figura 75. Configuración de la fuente.

```
#font
fill_i = 'grey'
font_size_i = '10'
font_i = 'Arial'
```

Figura 76. Diagrama P&ID básico para detección de banderas.

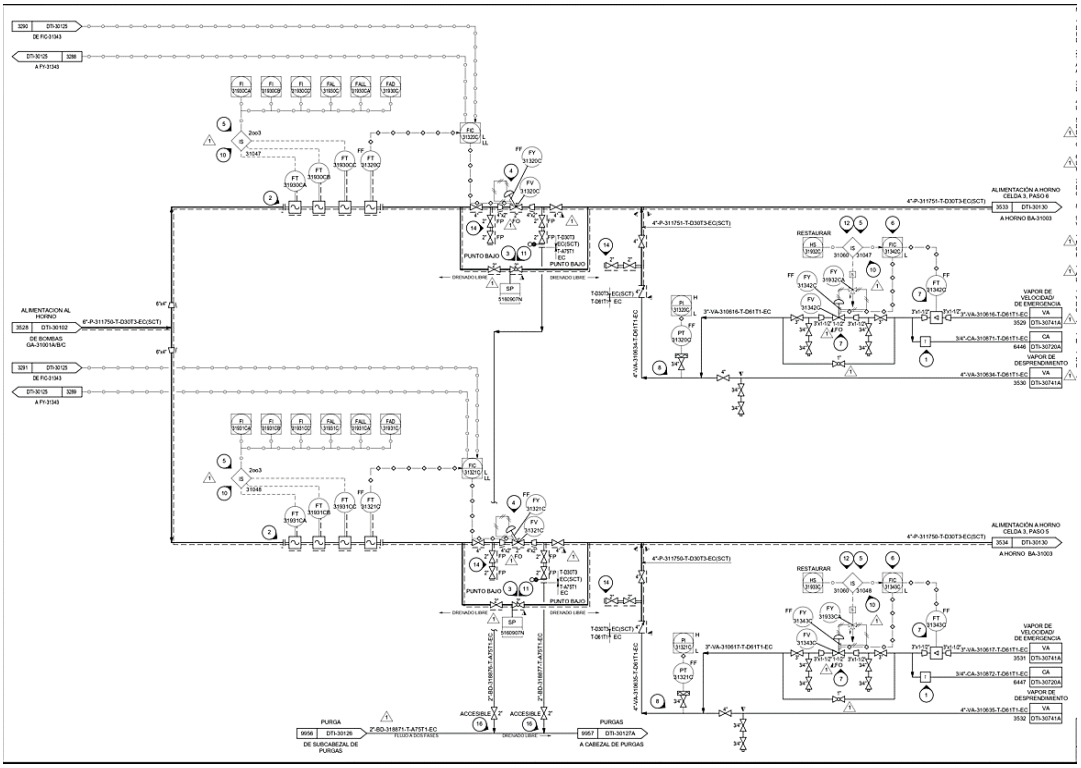


Figura 77. Plantilla HMI basada en la identificación de banderas.

El resultado se ha sometido a revisión con el asesor, quien ha otorgado su aprobación al diseño de la bandera. Aunque aún se deben realizar pruebas de calidad para obtener un veredicto definitivo, esta primera muestra ha dejado satisfecho al asesor. Como parte del plan de trabajo, se tiene previsto avanzar en diferentes elementos cada semana y, finalmente, unificar los algoritmos para generar la imagen de la plantilla definitiva.

Semana 8: Dibujos vectoriales en formato svg etapa 2

Esta semana, se enfoca en desarrollar los algoritmos para cuatro elementos clave: generadores de vapor, sistema de ventilación, intercambiadores y separadores de crudo. Comenzaremos por los generadores de vapor, ya que son elementos frecuentes en esta planta. Normalmente, no suelen estar acompañados por otros tipos de elementos, a excepción de banderas, líneas de conexión y válvulas. Sin embargo, es común que haya más de un generador de vapor en el diagrama P&ID. El proceso es muy similar al que se llevó a cabo para dibujar las banderas, exceptuando la condición de rotación, está solo aplica en esta versión a la bandera, finalmente, procedemos a desarrollar el algoritmo.

En primer lugar, el algoritmo filtra los datos del DataFrame “datos” para seleccionar aquellos registros donde el valor de la columna "confidence" sea mayor a 0.90. Los datos filtrados se almacenan en la variable “menos”. A continuación, se realizan copias de las columnas 'xmin', 'ymin', 'xmax' y 'ymax' del DataFrame “menos” y se guardan en la variable “coo”.

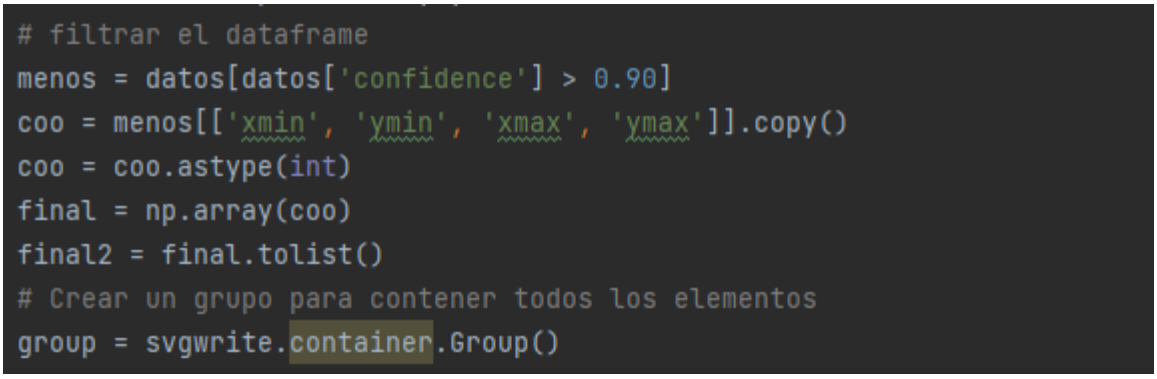
Posteriormente, se convierten los valores de la variable “coo” en números enteros utilizando el método astype(int). Después de eso, los valores de “coo” se convierten en un arreglo NumPy y se transforman en una lista, que se guarda en la variable “final2”.

Luego, se procede a crear un objeto Group utilizando la biblioteca svgwrite para almacenar elementos gráficos **(Fig.78)**. A continuación, se itera sobre los valores de final2 y se realizan la siguiente serie de acciones **(Fig.79-81)**:

- Realiza un recorte de una imagen (img_pil) utilizando las coordenadas proporcionadas.
- Aplica reconocimiento óptico de caracteres (OCR) a la imagen recortada utilizando la biblioteca Tesseract (tess). Se busca un patrón específico en el texto resultante utilizando una expresión regular.
- Si se encuentra un patrón correspondiente a "CB-" en el texto, se crea un nuevo grupo en el objeto dwg (que representa el gráfico SVG).
- Si se encuentra un patrón correspondiente a "FA-", se omite la generación de elementos gráficos, esto se debe a que normalmente los elementos “generador de vapor” y “separador de crudo” suelen presentar confusiones al sistema en su detección.

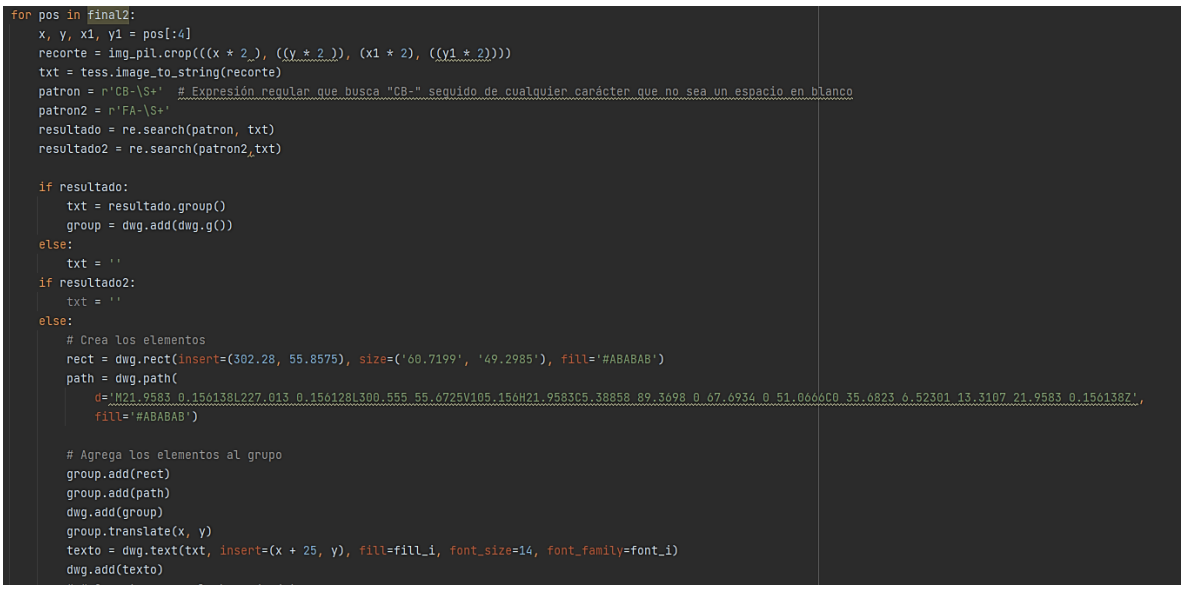
- En caso contrario, se genera el elemento gráfico, y se agrega al grupo dwg. Además, se realiza una traslación de posición para el grupo y se agrega un texto correspondiente utilizando la función texto ().

Figura 78. Adaptación de los datos suministrados por el DataFrame.



```
# filtrar el dataframe
menos = datos[datos['confidence'] > 0.90]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
```

Figura 79. Algoritmo de dibujo para separadores de vapor.



```
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2 ), ((y * 2 )), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'CB-\S+'  # Expresión regular que busca "CB-" seguido de cualquier carácter que no sea un espacio en blanco
    patron2 = r'FA-\S+'
    resultado = re.search(patron, txt)
    resultado2 = re.search(patron2,txt)

    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
    if resultado2:
        txt = ''
    else:
        # Crea los elementos
        rect = dwg.rect(insert=(302.28, 55.8575), size=('60.7199', '49.2985'), fill='#ABABAB')
        path = dwg.path(
            d='M21.9583 0.156138L227.013 0.156128L300.555 55.6725V105.156H21.9583C5.38858 89.3698 0 67.6934 0 51.0666C0 35.6823 6.52301 13.3107 21.9583 0.156138Z',
            fill='#ABABAB')

        # Agrega los elementos al grupo
        group.add(rect)
        group.add(path)
        dwg.add(group)
        group.translate(x, y)
        texto = dwg.text(txt, insert=(x + 25, y), fill=fill_i, font_size=14, font_family=font_i)
        dwg.add(texto)
```

Figura 80. Diagrama P&ID básico para detección de generadores de vapor.

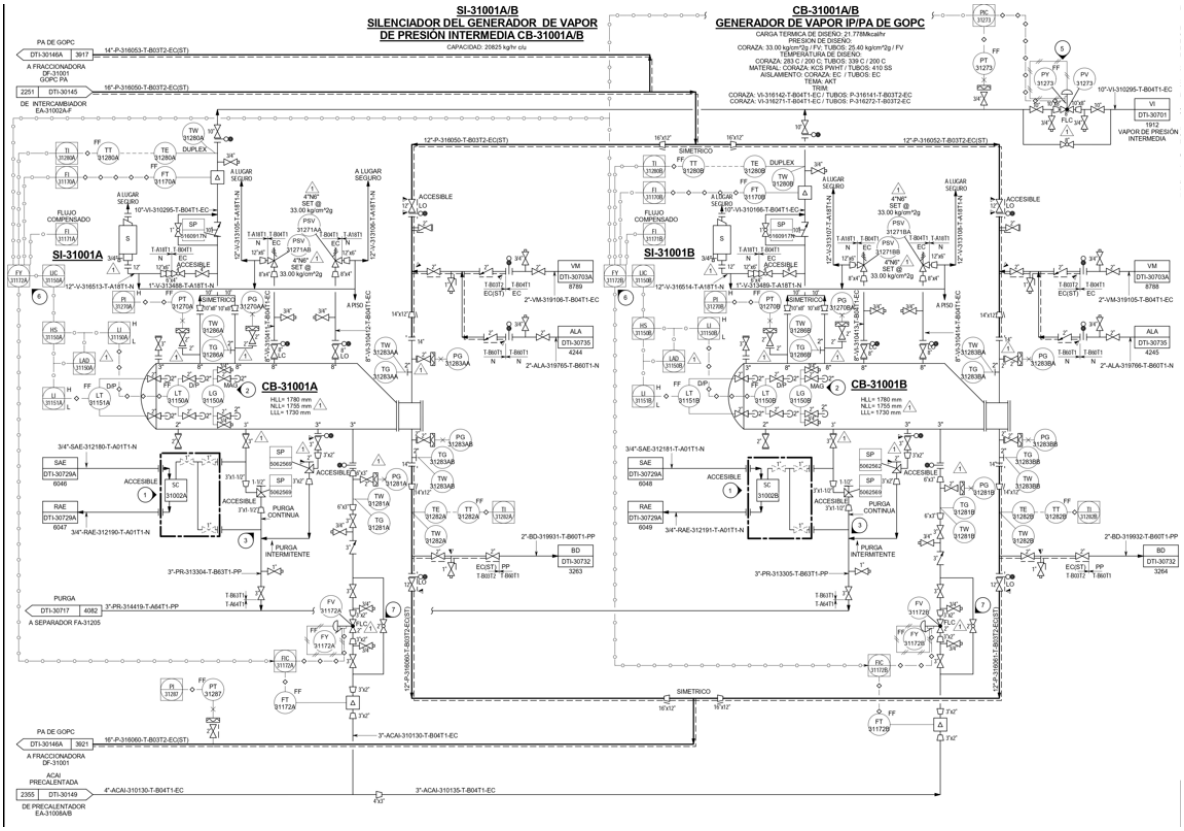
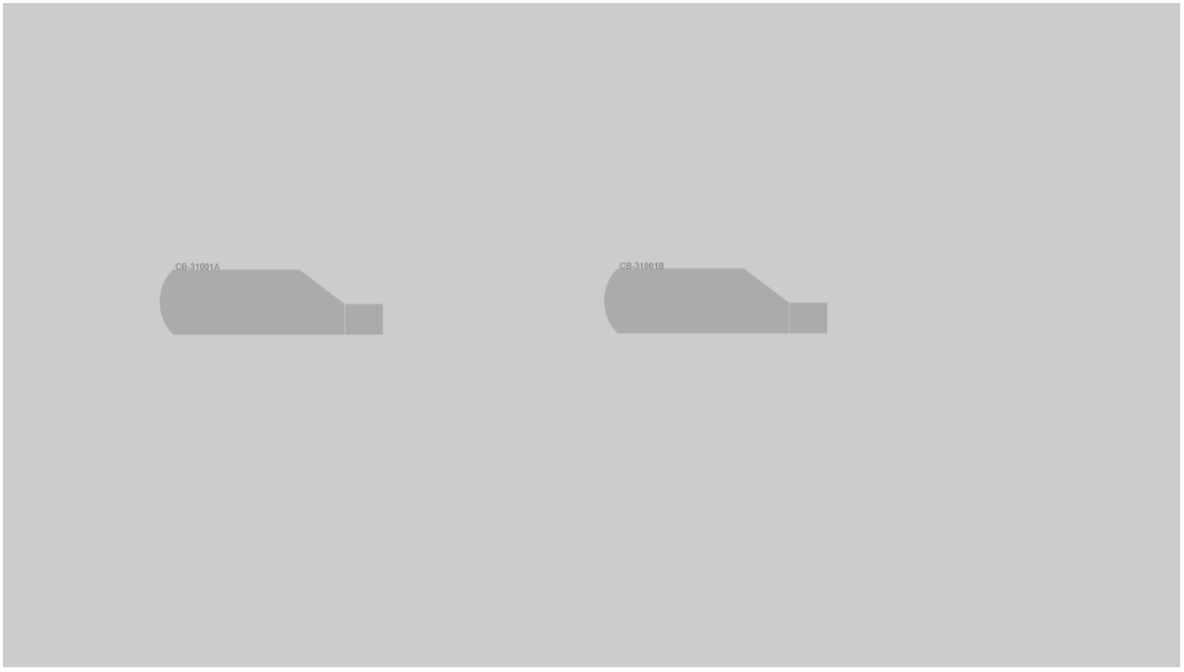


Figura 81. Plantilla HMI basada en la identificación de generadores de vapor.



Se ha desarrollado un algoritmo fundamental que se utiliza para dibujar todos los componentes, adaptando la lista de acciones según el elemento identificado. En resumen, este algoritmo realiza una serie de pasos para procesar los datos: primero filtra los datos, luego realiza recortes de imágenes, aplica OCR para extraer el texto y busca patrones específicos (las primeras dos letras del tag) para asegurar que el elemento detectado coincida con lo esperado, lo que reduce el margen de error. Una vez completado este proceso, se generan elementos gráficos SVG basados en los resultados obtenidos. Cabe destacar que los valores y objetos utilizados en cada paso del código pueden variar según el contexto y los requisitos del proyecto.

Siguiendo este orden de ideas y para no repetir la misma información se procede a realizar una lista de los parámetros que varían por cada algoritmo: porcentaje de exactitud, texto detectado por el OCR y Grupo SVG implementando. Teniendo esto claro se procede a realizar el dibujo de los ventiladores o el sistema de ventilación este elemento tiene los siguientes parámetros **(Fig.82-86)**:

- Porcentaje de exactitud: 53%
- Texto detectado por el OCR: "EC-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "EC-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene diferentes figuras, incluyendo un rectángulo (rect), dos círculos (circle1 y circle2) y un path (líneas de recorrido).

Figura 82. Algoritmo de dibujo para sistema de ventilación.

```
# filtrar el dataframe
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'EC-\S+'

    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())

        # Crea los elementos

        rect = dwg.rect(insert=(0.489502, 0.724243), size=('339.07', '110.496'), rx='10', fill='#ABABAB')
        path = dwg.path(d='M0.49881 55.9719H339.61', stroke='#CDCDCD', stroke_width='3')
        circle1 = dwg.circle(center=(76.3179, 55.9721), r='31.3675', fill='#CDCDCD', stroke='#CDCDCD',
                             stroke_width='3')
        circle2 = dwg.circle(center=(252.791, 55.9721), r='31.3675', fill='#CDCDCD', stroke='#CDCDCD',
                             stroke_width='3')

        # Agrega los elementos al grupo
        group.add(rect)
        group.add(path)
        group.add(circle1)
        group.add(circle2)
        dwg.add(group)
        group.translate(x, y)
        texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
        dwg.add(texto)
```

Figura 83. Diagrama P&ID básico para detección de sistema de ventilación versión 1.

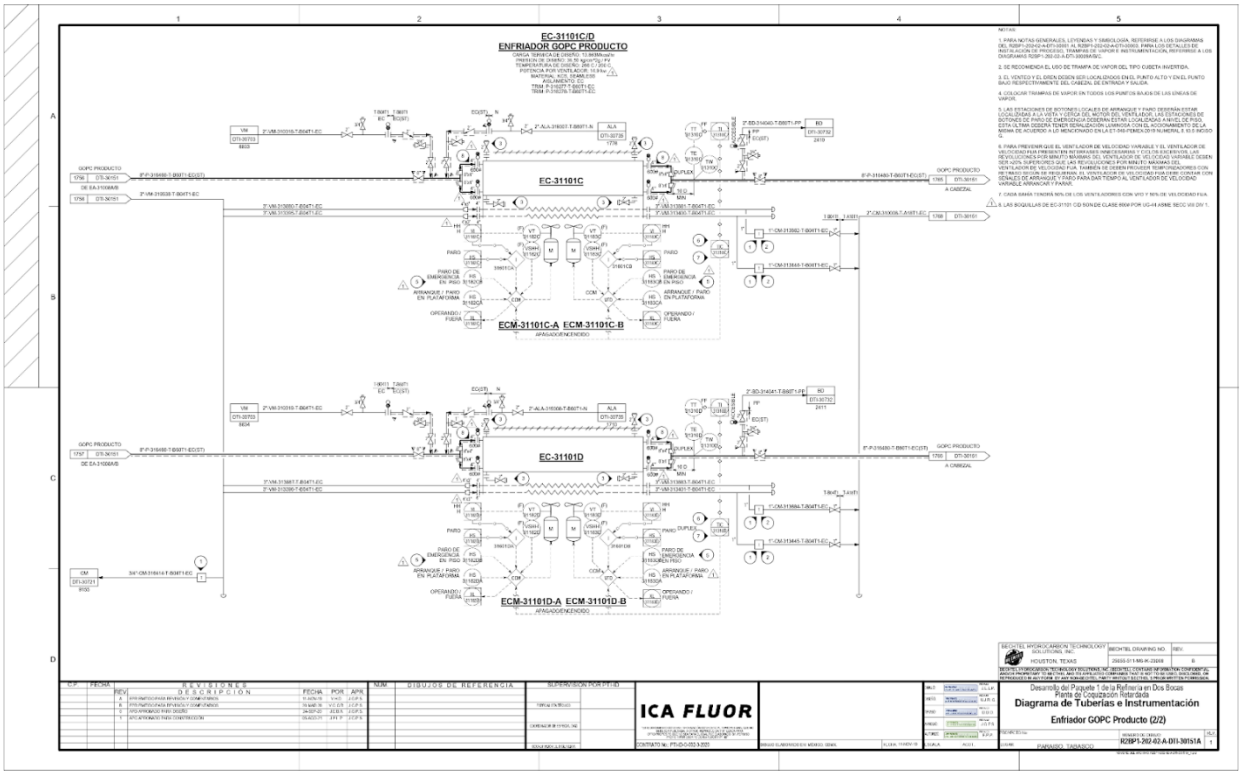


Figura 84. Plantilla HMI basada en la identificación de sistema de ventilación V1.

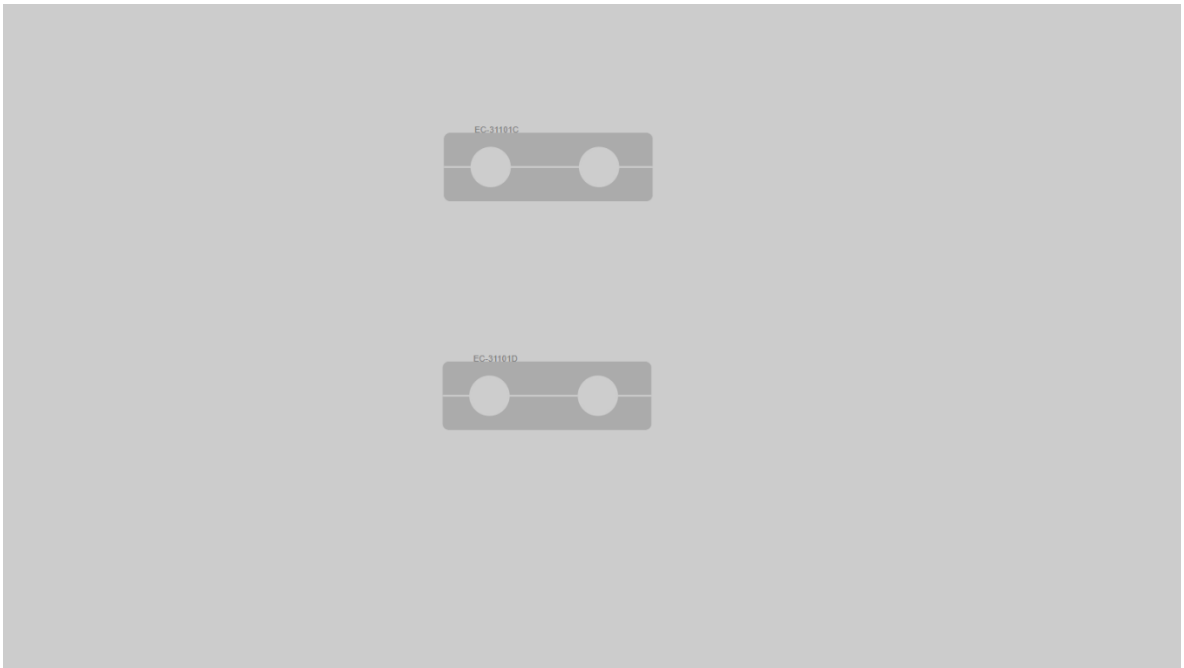


Figura 85. Diagrama P&ID básico para detección de sistema de ventilación versión 2.

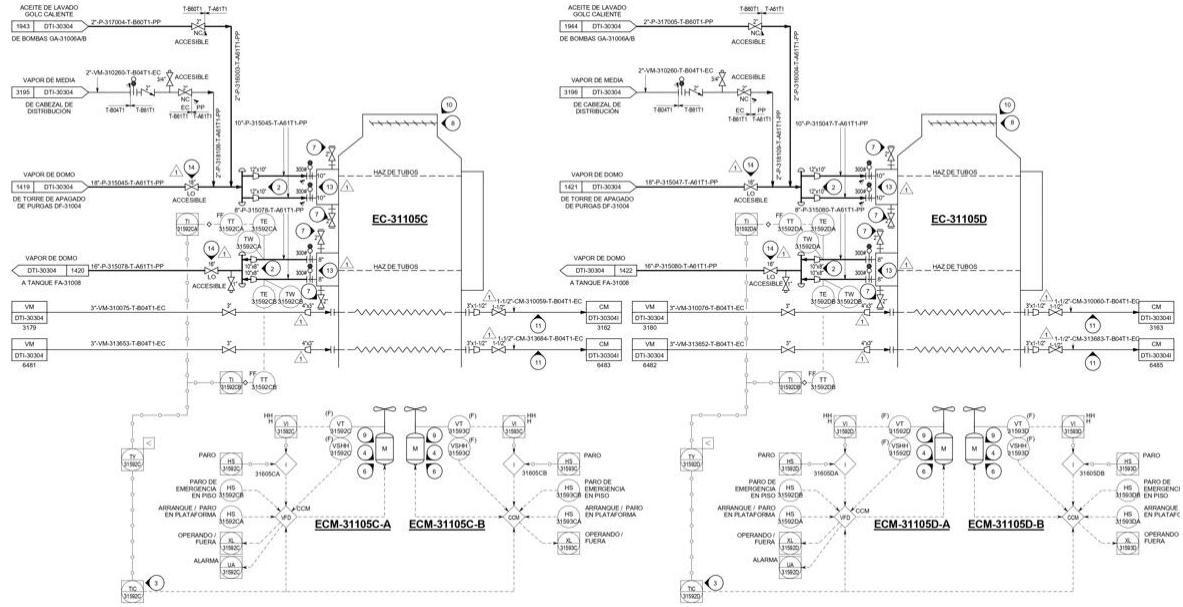
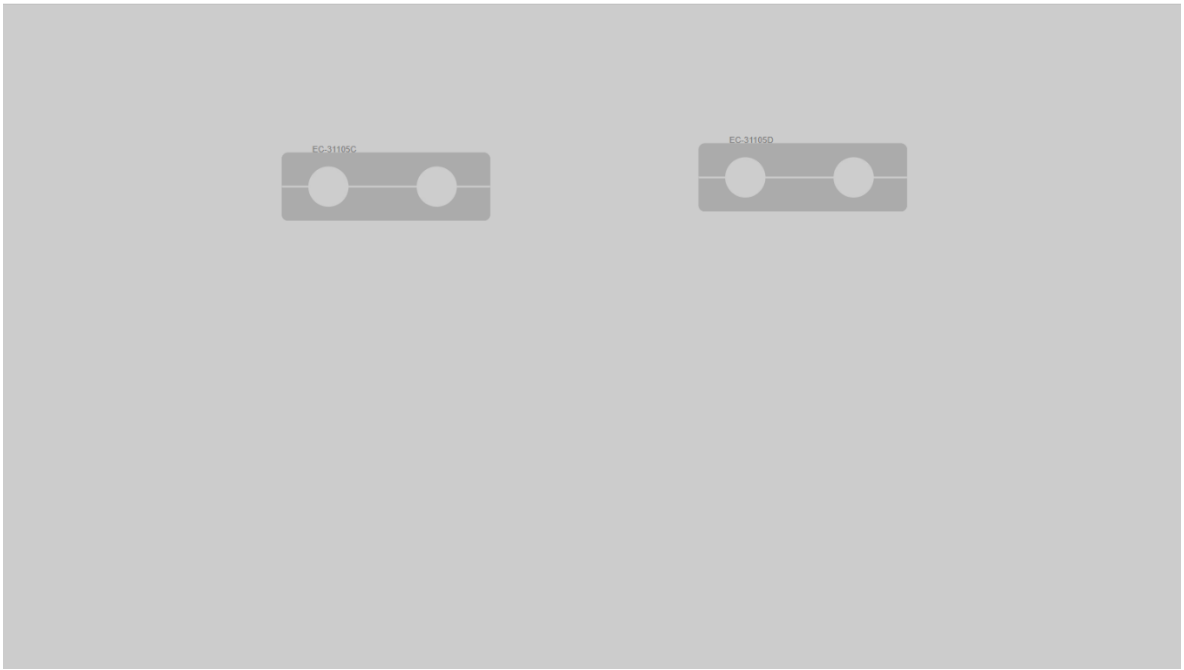


Figura 86. Plantilla HMI basada en la identificación de sistema de ventilación V1.



Dibujo de los intercambiadores de calor, este elemento tiene los siguientes parámetros **(Fig.87-89)**:

- Porcentaje de exactitud: 53%
- Texto detectado por el OCR: "EA-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "EA-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene diferentes figuras dos rectángulos (rect1 y rect2) y un path (líneas de recorrido).

Figura 87. Algoritmo de dibujo para intercambiadores de calor.

```
# filtrar el dataframe
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coordenadas = menos[['name', 'xmin', 'ymin', 'xmax', 'ymax']].copy()
df = pd.DataFrame(coordenadas)
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'EA-\S+'  # Expresión regular que busca "ea-" seguido de cualquier carácter que no sea un espacio en blanco

    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())

        # Crea los elementos

        rect1 = dwg.rect(insert=(62.6928, 1.2793), size=('6.16827', '84.468'), fill='#ABABAB', stroke='#CDCDCD')
        rect2 = dwg.rect(insert=(0.56427, 1.2793), size=('6.16827', '84.468'), fill='#ABABAB', stroke='#CDCDCD')
        path = dwg.path(
            d='M7.14621 9.87976H326.316C352.232 16.0292 350.578 73.3755 326.316 77.1469H7.14621V9.87976Z',
            fill='#ABABAB')

        # Agrega los elementos al grupo
        group.add(rect1)
        group.add(rect2)
        group.add(path)
        dwg.add(group)
        group.translate(x, y)
        texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
        dwg.add(texto)
```

Figura 88. Diagrama P&ID básico detección de sistema de intercambiadores de calor.

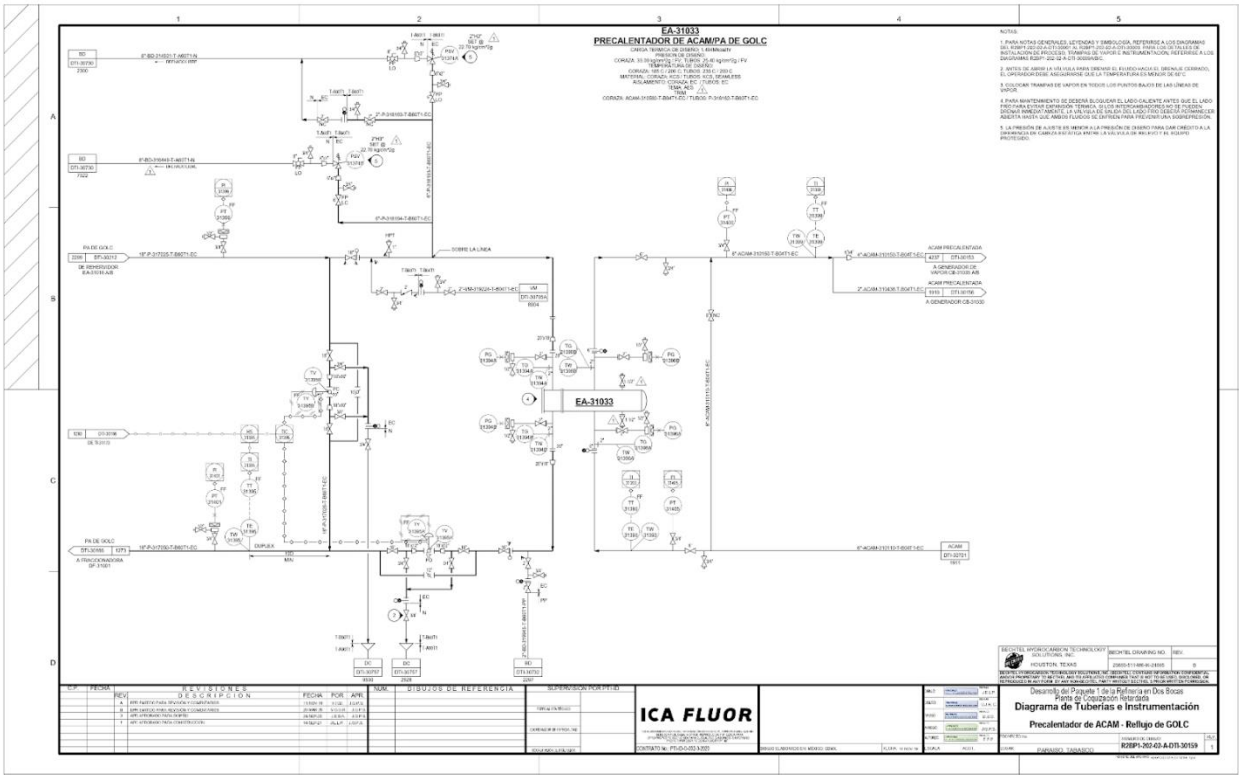
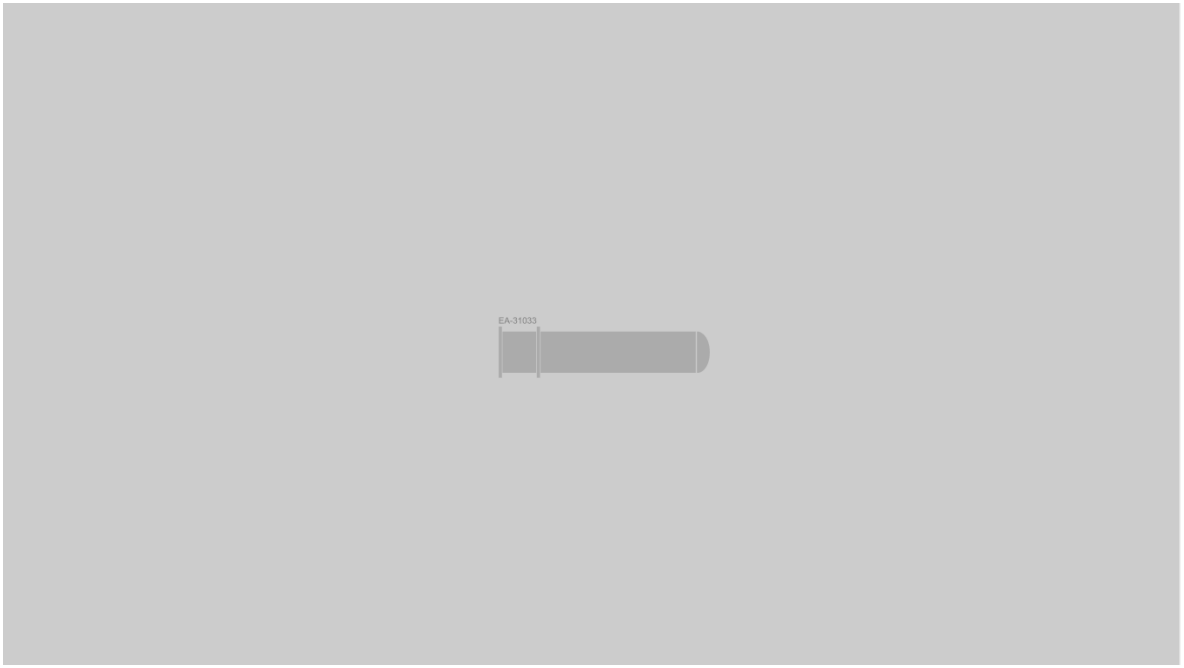


Figura 89. Plantilla HMI basada en la identificación de intercambiadores de calor.



Dibujo de los separadores de crudo, este elemento tiene los siguientes parámetros **(Fig.90-101)**:

- Porcentaje de exactitud: 53%, todas las variaciones tienen este mismo porcentaje.
- Texto detectado por el OCR: "FA-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "FA-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene diferentes figuras tipo path para cada versión del elemento.

Figura 90. Algoritmo de dibujo para separadores de crudo versión 1.

```
# filtrar el dataframe
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coordenadas = menos[['name', 'xmin', 'ymin', 'xmax', 'ymax']].copy()
df = pd.DataFrame(coordenadas)
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'EA-\S+'  # Expresión regular que busca "ea-" seguido de cualquier carácter que no sea un espacio en blanco

    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())

        # Crea los elementos

        rect1 = dwg.rect(insert=(62.6928, 1.2793), size=('6.16827', '84.468'), fill='#ABABAB', stroke='#CDCDCD')
        rect2 = dwg.rect(insert=(0.56427, 1.2793), size=('6.16827', '84.468'), fill='#ABABAB', stroke='#CDCDCD')
        path = dwg.path(
            d='M7.14621 9.87976H326.316C352.232 16.0292 350.578 73.3755 326.316 77.1469H7.14621V9.87976Z',
            fill='#ABABAB')

        # Agrega los elementos al grupo
        group.add(rect1)
        group.add(rect2)
        group.add(path)
        dwg.add(group)
        group.translate(x, y)
        texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
        dwg.add(texto)
```

Figura 91. Diagrama P&ID básico para detección de separadores de crudo versión 1.

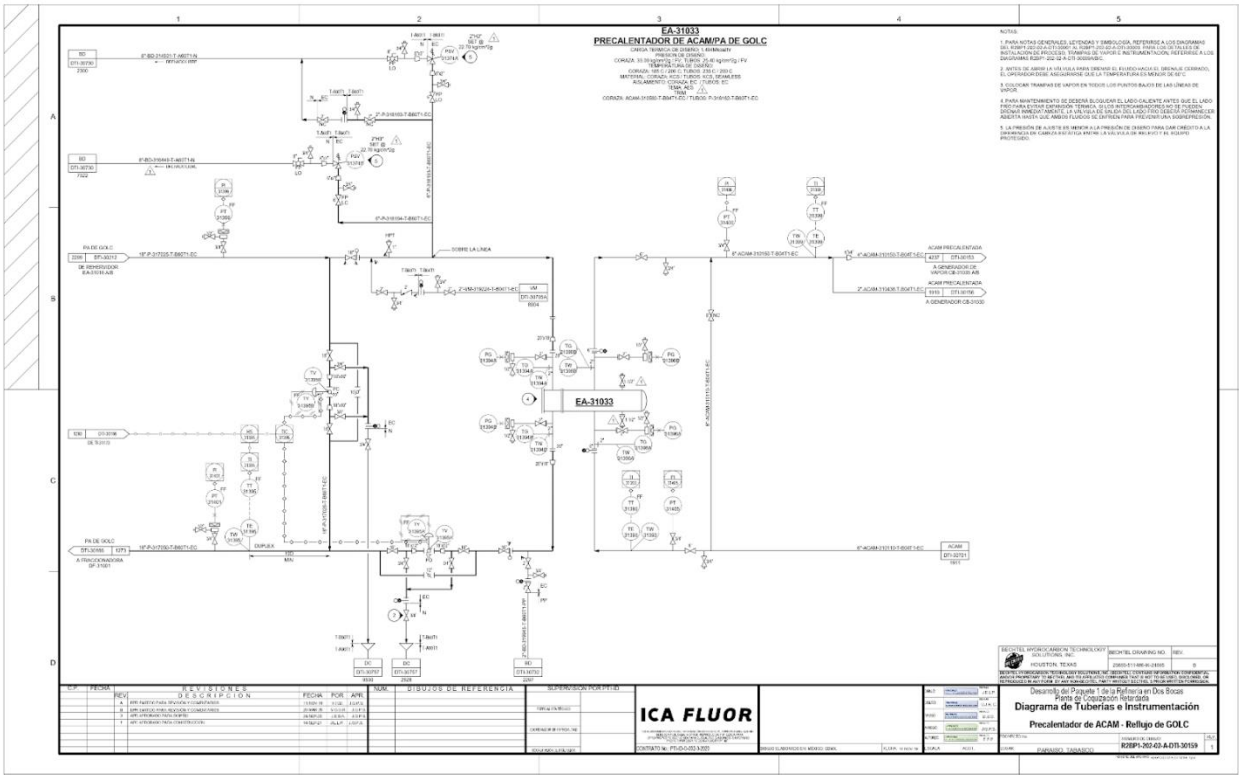


Figura 92. Plantilla HMI basada en la identificación de separadores de crudo V1.

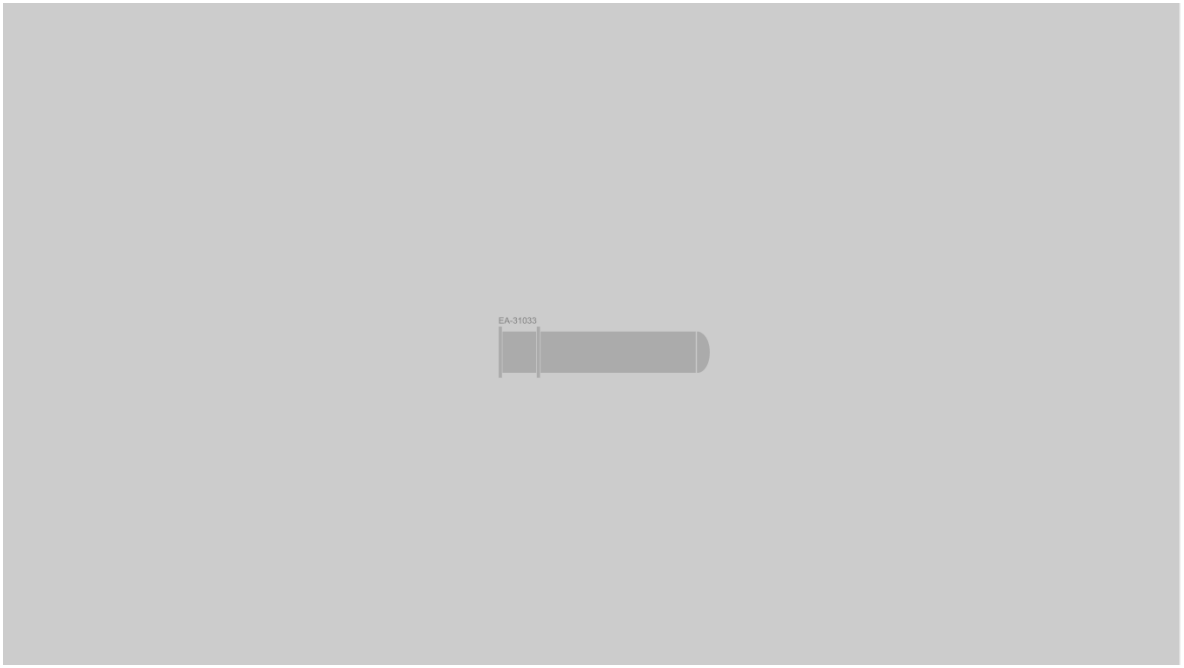
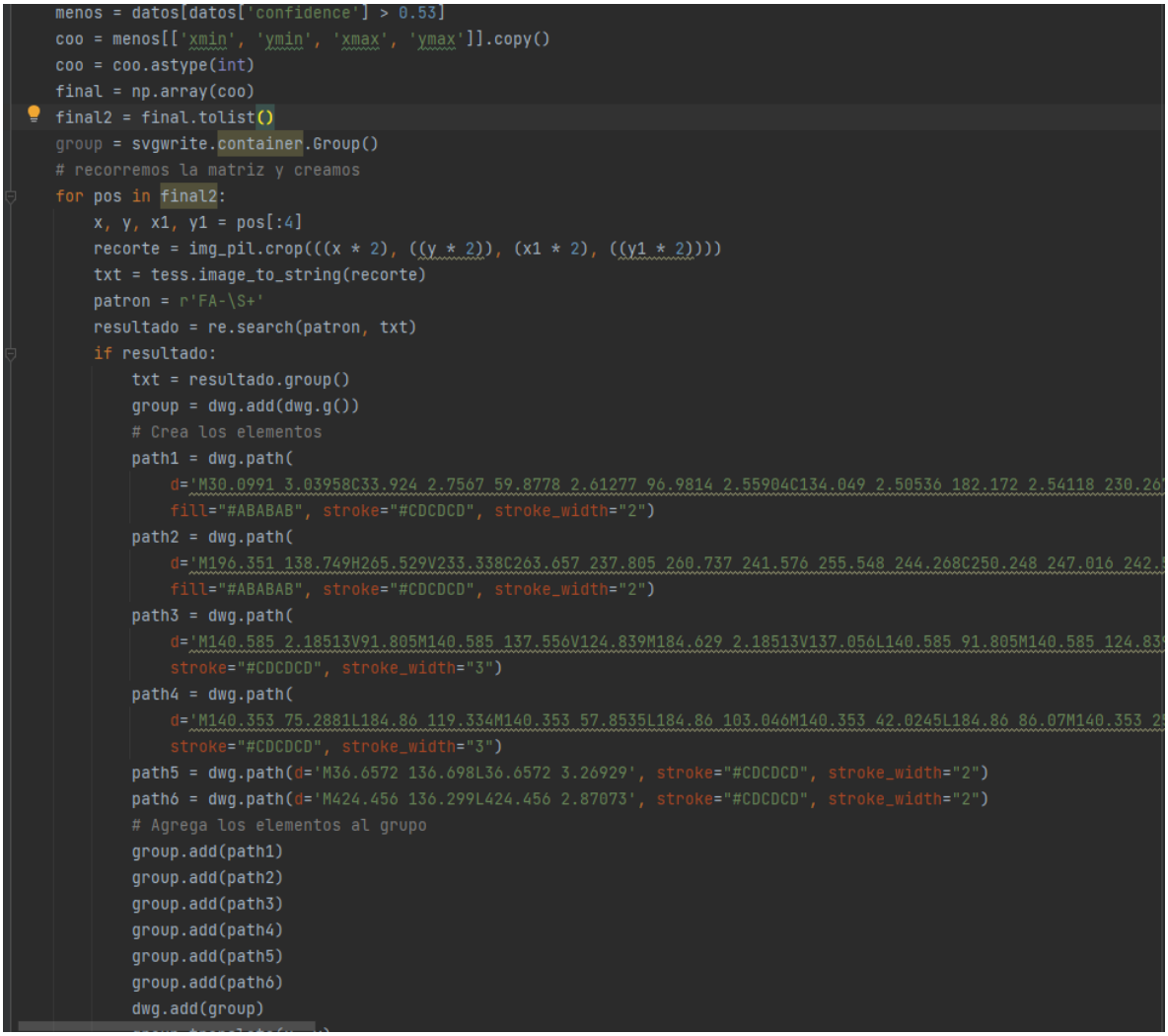


Figura 93. Algoritmo de dibujo para separadores de crudo versión 2.



```
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'FA-\S+'
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
        # Crea los elementos
        path1 = dwg.path(
            d='M30.0991 3.03958C33.924 2.7567 59.8778 2.61277 96.9814 2.55904C134.049 2.50536 182.172 2.54118 230.26
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path2 = dwg.path(
            d='M196.351 138.749H265.529V233.338C263.657 237.805 260.737 241.576 255.548 244.268C250.248 247.016 242.5
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path3 = dwg.path(
            d='M140.585 2.18513V91.805M140.585 137.556V124.839M184.629 2.18513V137.056L140.585 91.805M140.585 124.83
            stroke="#CDCDCD", stroke_width="3")
        path4 = dwg.path(
            d='M140.353 75.2881L184.86 119.334M140.353 57.8535L184.86 103.046M140.353 42.0245L184.86 86.07M140.353 2
            stroke="#CDCDCD", stroke_width="3")
        path5 = dwg.path(d='M36.6572 136.698L36.6572 3.26929', stroke="#CDCDCD", stroke_width="2")
        path6 = dwg.path(d='M424.456 136.299L424.456 2.87073', stroke="#CDCDCD", stroke_width="2")
        # Agrega los elementos al grupo
        group.add(path1)
        group.add(path2)
        group.add(path3)
        group.add(path4)
        group.add(path5)
        group.add(path6)
        dwg.add(group)
```

Figura 94. Diagrama P&ID básico para detección de separadores de crudo versión 2.

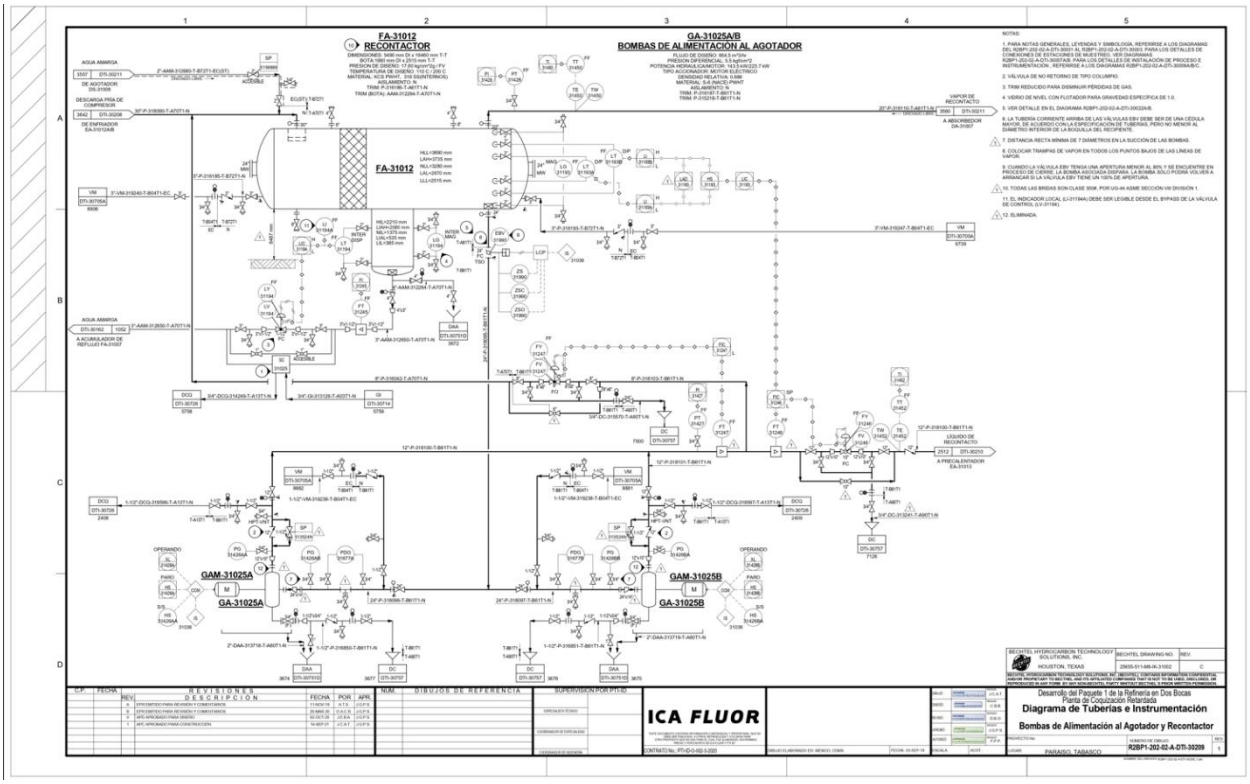


Figura 95. Plantilla HMI basada en la identificación de separadores de crudo V2.

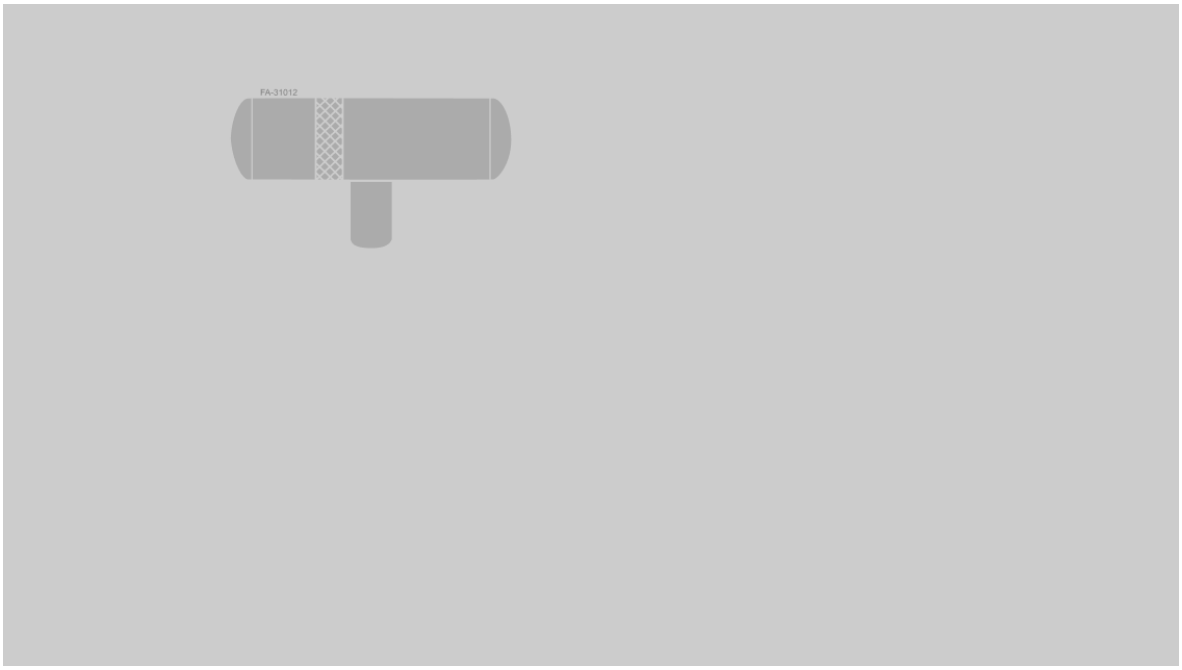


Figura 96. Algoritmo de dibujo para separadores de crudo versión 3.

```
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'FA-\S+'
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
        # Crea los elementos
        path1 = dwg.path(
            d='M23.5368 1.42607C26.5268 1.18867 46.8907 1.06709 76.0373 1.02178C105.145 0.976526 142.935 1.00672 180
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path2 = dwg.path(
            d='M150.226 115.216H211.575V199.167C209.917 203.109 207.335 206.432 202.751 208.807C198.056 211.239 191.
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path3 = dwg.path(d='M335.307 112.899L335.307 1.87329', stroke="#CDCDCD", stroke_width="2")
        path4 = dwg.path(d='M24.4072 112.899L24.4072 1.87329', stroke="#CDCDCD", stroke_width="2")
        # Agrega los elementos al grupo
        group.add(path1)
        group.add(path2)
        group.add(path3)
        group.add(path4)
        dwg.add(group)
        group.translate(x, y)
        texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
        dwg.add(texto)
```

Figura 97. Diagrama P&ID básico para detección de separadores de crudo versión 3.

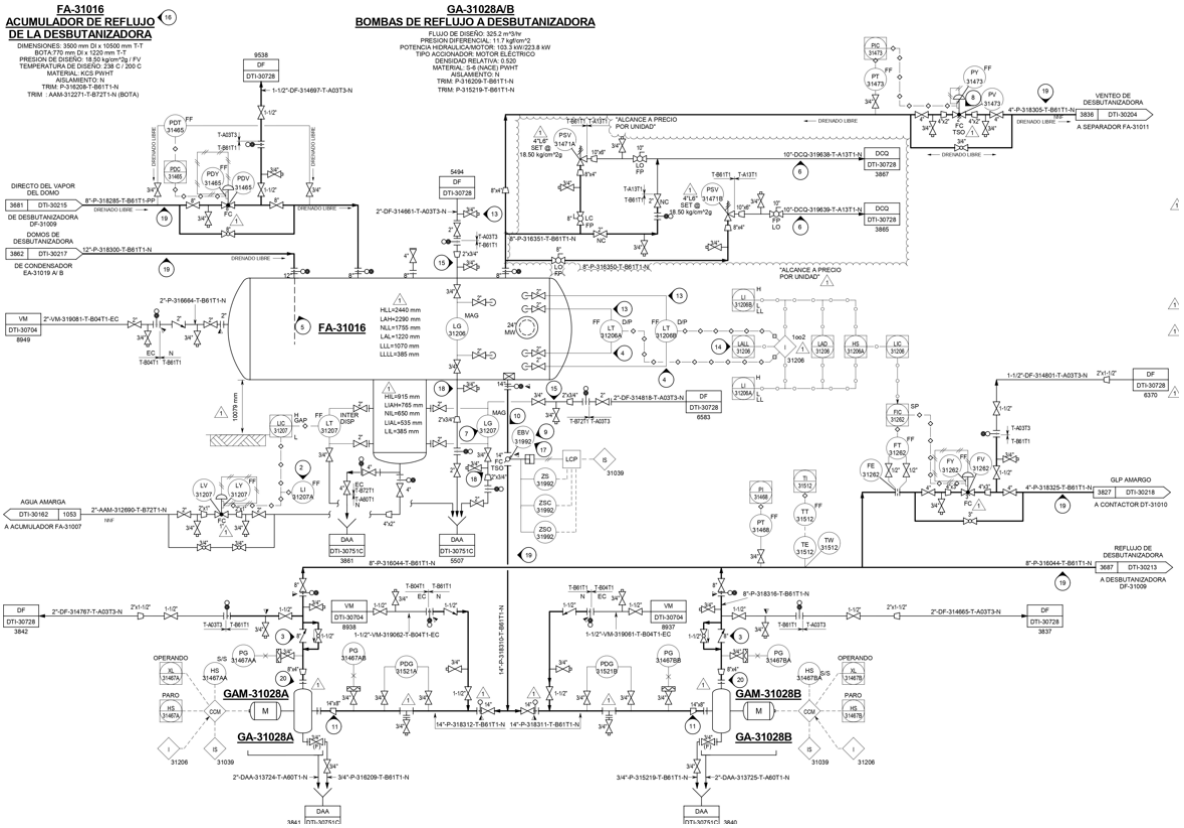


Figura 98. Plantilla HMI basada en la identificación de separadores de crudo V3.

Figura 99. Algoritmo de dibujo para separadores de crudo versión 4.

```
menos = datos[datos['confidence'] > 0.53]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'FA-\S+'
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
        # Crea los elementos
        path1 = dwg.path(
            d='M31.5161 2.20196C35.4441 1.88693 62.1122 1.72649 100.244 1.66661C138.336 1.60679 187.789 1.64671 237.
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path2 = dwg.path(
            d='M141.204 153.292H222.121V263.501C219.928 268.739 216.501 273.168 210.405 276.324C204.198 279.539 195.
            fill="#ABABAB", stroke="#CDCDCD", stroke_width="2")
        path3 = dwg.path(
            d='M304.719 1.34376V101.297M304.719 152.324V138.14M351.782 1.34376V151.766L304.719 101.297M304.719 138.1
            stroke="#CDCDCD", stroke_width="3")
        path4 = dwg.path(
            d='M304.472 82.876L352.03 132M304.472 63.431L352.03 113.834M304.472 45.7769L352.03 94.9011M304.472 27.35
            stroke="#CDCDCD", stroke_width="3")
        path5 = dwg.path(d='M36.8511 151.318L36.8511 1.97538', stroke="#CDCDCD", stroke_width="2")
        path6 = dwg.path(d='M436.938 151.318L436.938 1.97538', stroke="#CDCDCD", stroke_width="2")
        # Agrega los elementos al grupo
        group.add(path1)
        group.add(path2)
        group.add(path3)
        group.add(path4)
        group.add(path5)
        group.add(path6)
```

Figura 100. Diagrama P&ID básico para detección de separadores de crudo versión 4.

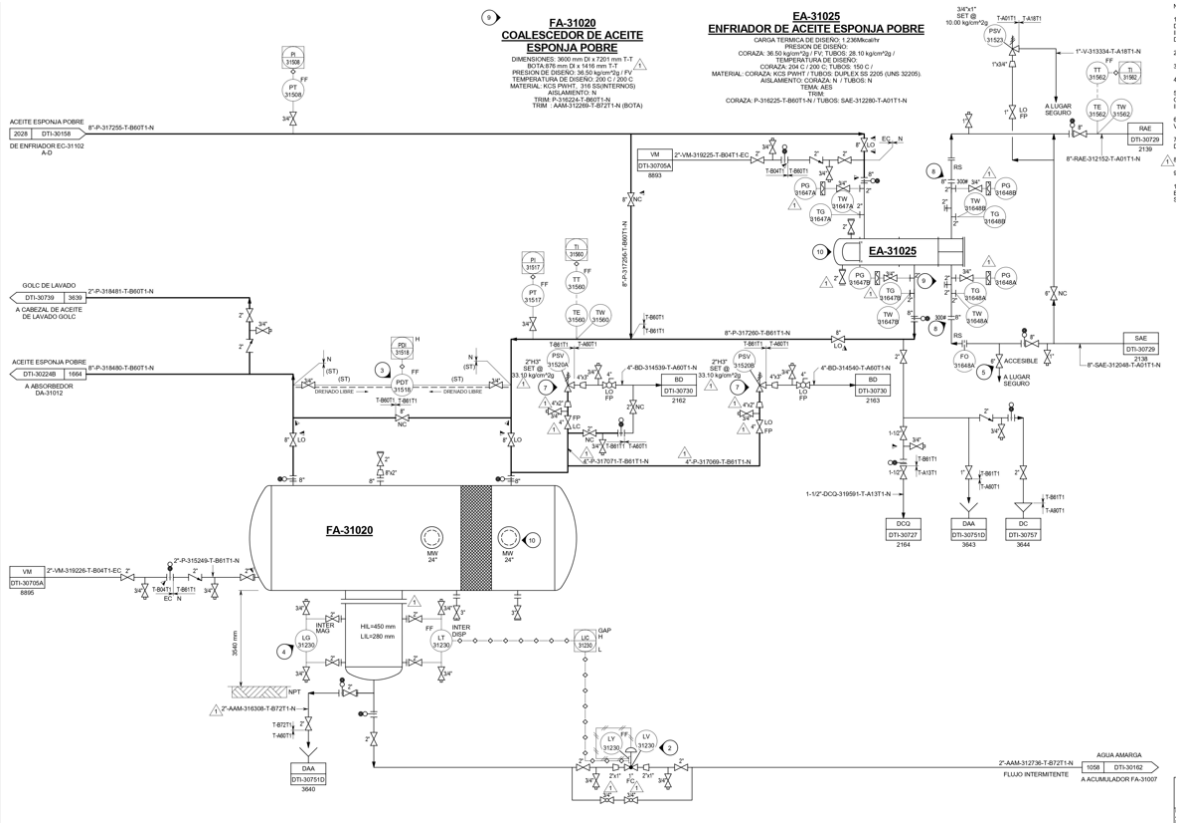
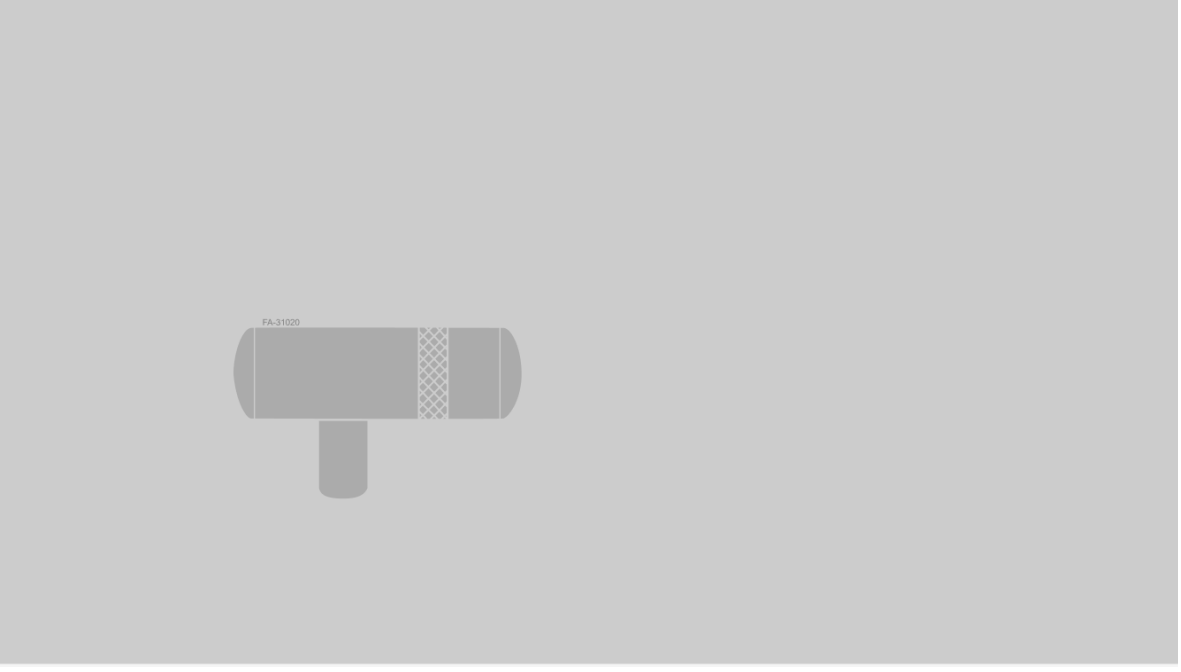


Figura 101. Plantilla HMI basada en la identificación de separadores de crudo V4.



Nota: Como se puede observar, el porcentaje de exactitud en todos los algoritmos es cercano al 50%. Esto se debe a que se busca abarcar la mayor cantidad de variantes que aún conserven similitud con el objeto deseado, y luego se utiliza un segundo filtro de texto para asegurar la identificación precisa del objeto. Si en el futuro el dataset de imágenes para cada elemento aumenta significativamente, es posible que el porcentaje de exactitud de muchos elementos se eleve, reduciendo así la posibilidad de errores en la detección.

Se ha llegado a un acuerdo con el asesor de la empresa para realizar la revisión una vez se haya completado todo el proceso de dibujo, lo cual está programado para la próxima semana. Esto permitirá una evaluación exhaustiva y garantizará que todos los elementos cumplan con los requisitos y estándares establecidos.

Semana 9: Dibujos vectoriales en formato SVG etapa 3

Durante esta semana, se ha planeado finalizar el desarrollo de los algoritmos de dibujo para los elementos restantes, que incluyen tanques, hornos, turbinas y válvulas. Al igual que en los algoritmos anteriores, se han tenido en cuenta los parámetros específicos de cada elemento, como el porcentaje de exactitud, el texto detectado por el OCR y el grupo SVG implementado.

Estos algoritmos se han diseñado siguiendo la misma metodología utilizada anteriormente, con el objetivo de asegurar la detección y representación precisa de cada elemento en el dibujo. Cada componente ha sido analizado individualmente, considerando sus características únicas y los patrones identificables en su etiquetado.

Una vez aplicados los algoritmos a los datos, se ha generado el grupo SVG correspondiente a cada elemento, el cual contiene las formas necesarias para representarlo de manera adecuada en el dibujo final.

Dibujo de los tanque o acumuladores, este elemento tiene los siguientes parámetros **(Fig.102-113)**:

- Porcentaje de exactitud: 80%, todas las variaciones tienen este mismo porcentaje.
- Texto detectado por el OCR: "FA-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "FA-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene diferentes figuras tipo path para cada versión del elemento.

En el caso de los elementos separadores de crudo y tanques, se ha establecido que comparten el mismo nombre de identificador, que es "FA". Esto puede generar ambigüedad y dificultad en la distinción entre ambos elementos. Para abordar esta situación, se ha aumentado el porcentaje de exactitud en la detección de los tanques.

Figura 102. Algoritmo de dibujo para tanques versión 1.

```
menos = datos[datos['confidence'] > 0.80]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2 ), ((y * 2 )), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    print(txt)
    patron = r'FA-\S+'  # Expresión regular que busca "FA" seguido de cualquier carácter que no sea un espacio en blanco
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
    # Crea los elementos
    path1 = dwg.path(
        d='M0.580811 45.7477C0.580811 -16.1733 91.9213 -12.6602 91.9213 45.7477V241.603C91.9213 301.322 0.580811 297.37 0.580811
        fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path2 = dwg.path(
        d='M0.678589 41.793L91.9208 41.793',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path3 = dwg.path(
        d='M0.678589 246.248L91.9208 246.248',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    group.add(path2)
    group.add(path3)
    dwg.add(group)
```

Figura 103. Diagrama P&ID básico para detección de tanques versión 1.

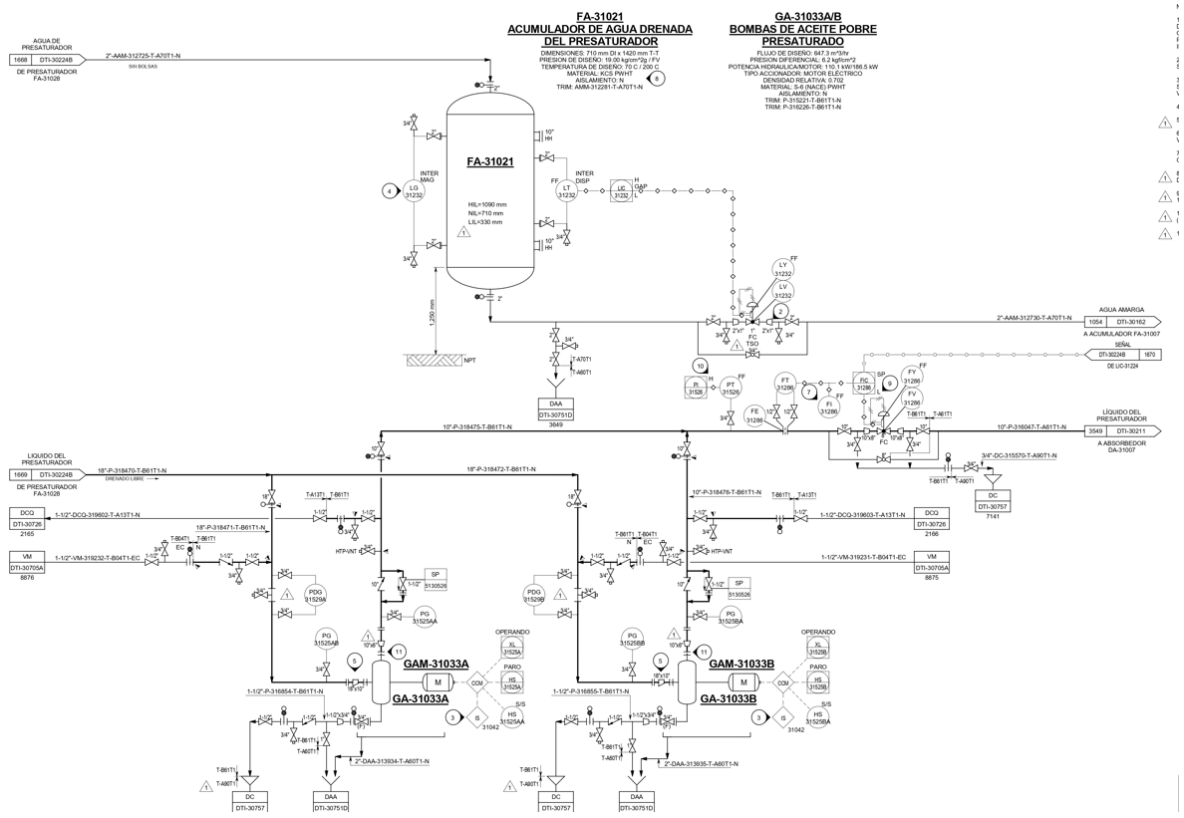


Figura 104. Plantilla HMI basada en la identificación de tanques V1.

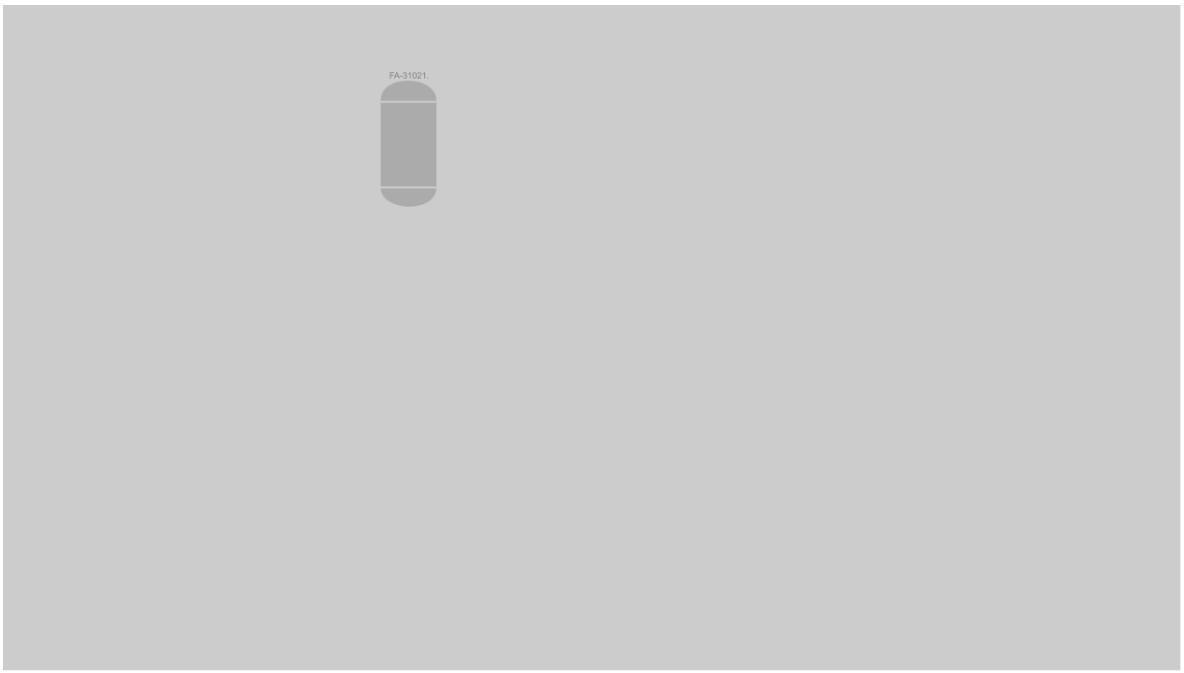


Figura 105. Algoritmo de dibujo para tanques versión 2.

```
menos = datos[datos['confidence'] > 0.80]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2 ), ((y * 2 )), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    print(txt)
    patron = r'FA-\S+'  # Expresión regular que busca "FA" seguido de cualquier carácter que no sea un espacio en bla
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
    # Crea los elementos
    path1 = dwg.path(
        d='M1.77698 49.8853C1.77698 -18.4258 171.565 -14.5502 171.565 49.8853V265.952C171.565 331.835 1.77698 327.475
        fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path2 = dwg.path(
        d='M1.9588 45.5225L171.564 45.5225',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path3 = dwg.path(
        d='M1.9588 271.077L171.564 271.077',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    group.add(path2)
    group.add(path3)
    dwg.add(group)
```

Figura 106. Diagrama P&ID básico para detección de tanques versión 2.

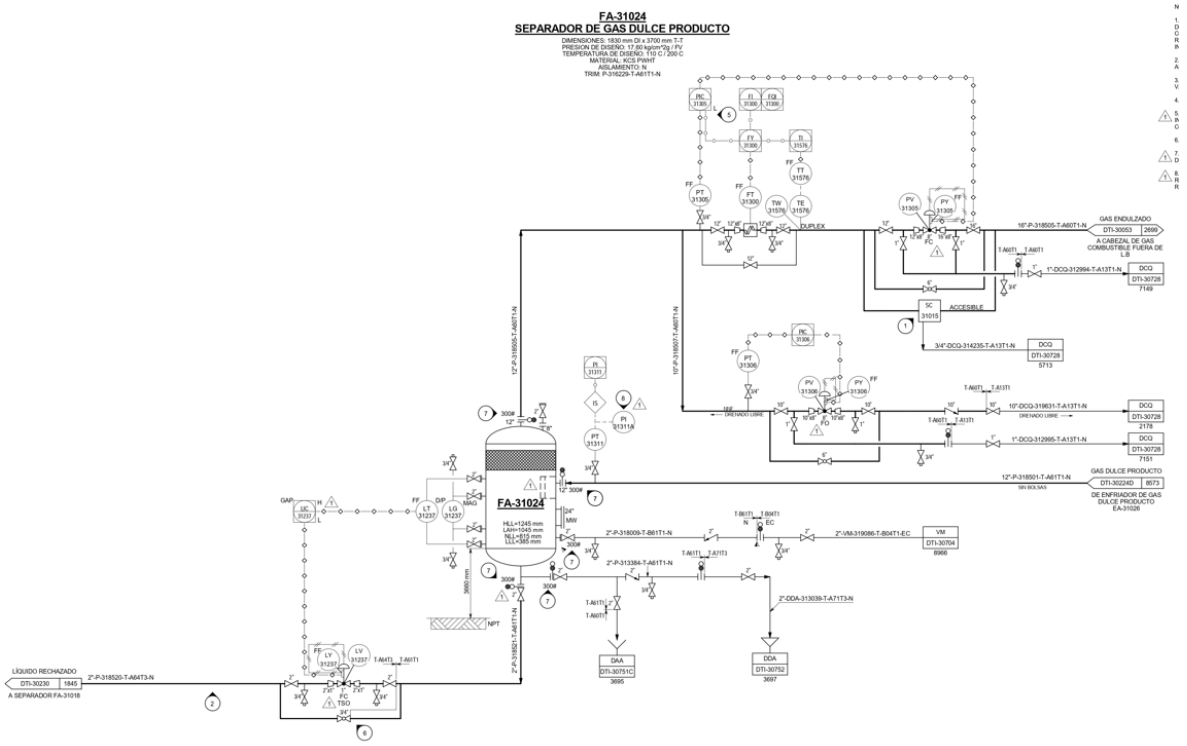


Figura 107. Plantilla HMI basada en la identificación de tanques V2.

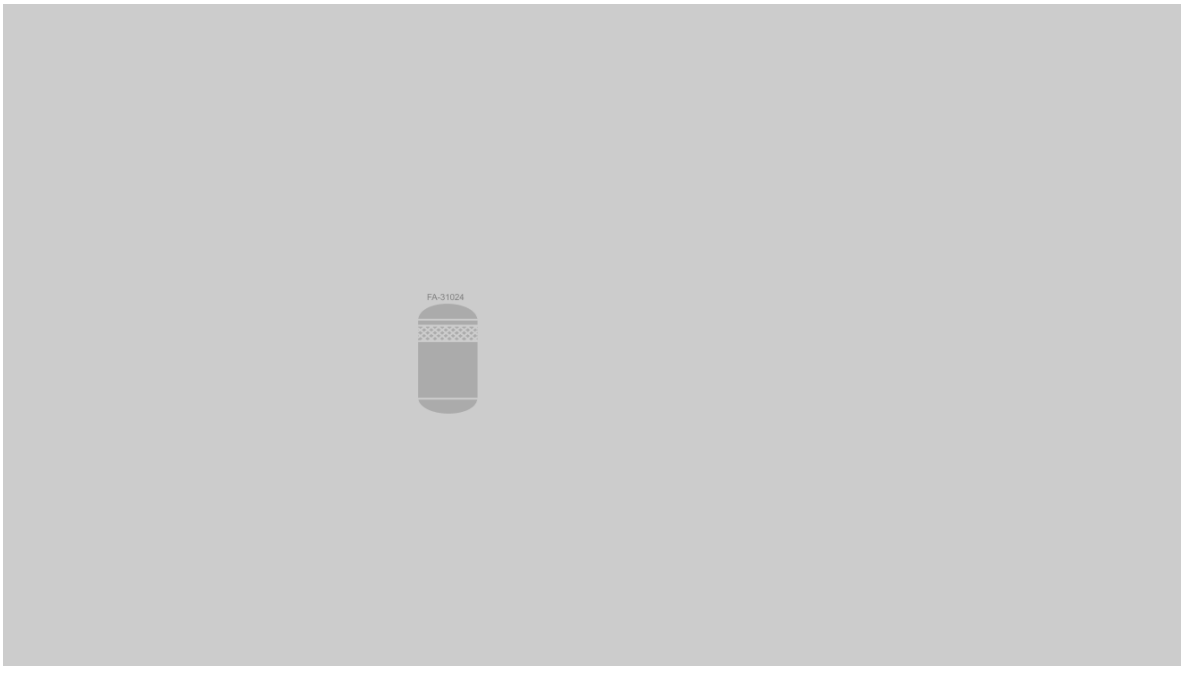


Figura 108. Algoritmo de dibujo para tanques versión 3.

```
menos = datos[datos['confidence'] > 0.80]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2 ), ((y * 2 )), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    print(txt)
    patron = r'FA-\S+'  # Expresión regular que busca "FA" seguido de cualquier carácter que no sea un espacio en blanco
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
    # Crea los elementos
    path1 = dwg.path(
        d='M1.47717 106.606H114.216M171.769 106.606H155.771M1.47717 62.5628L171.139',
        fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path2 = dwg.path(
        d='M93.4379 106.838L148.846 62.3311M71.5058 106.838L128.356',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path3 = dwg.path(
        d='M1.2253 80.024L25.0441 62.3311M1.2253 85.3307L26.7756',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    group.add(path2)
    group.add(path3)
```

Figura 109. Diagrama P&ID básico para detección de tanques versión 3.

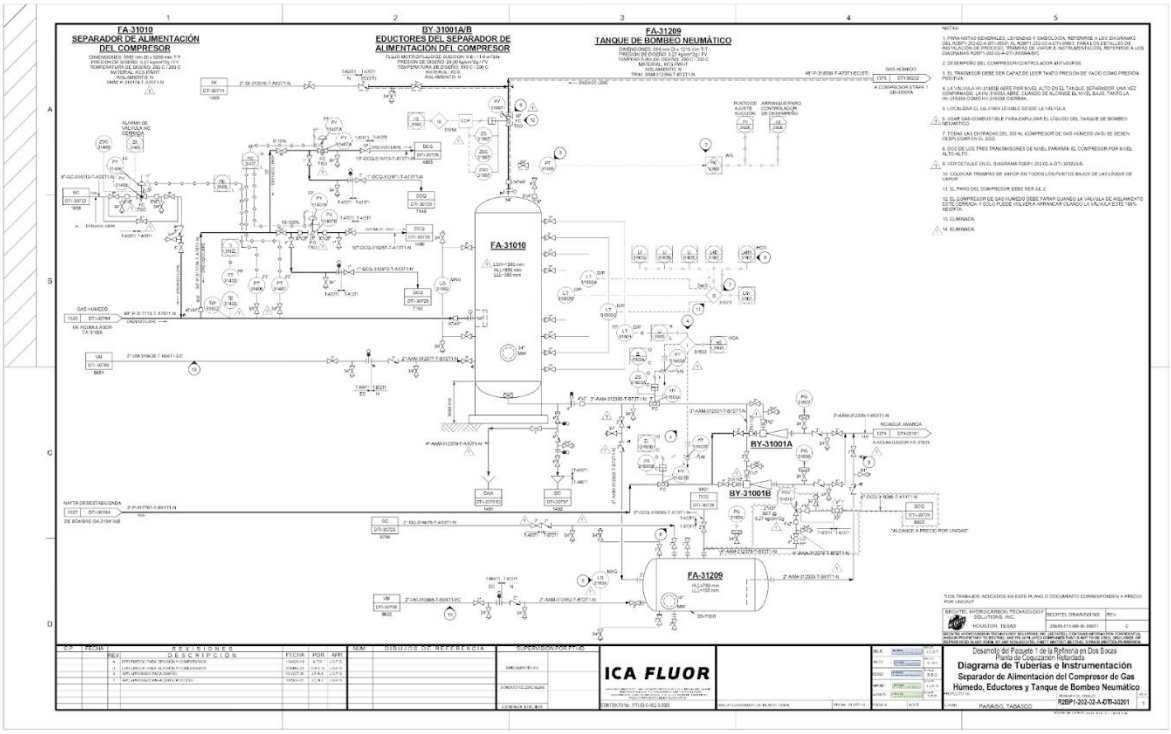


Figura 110. Plantilla HMI basada en la identificación de tanques V3.

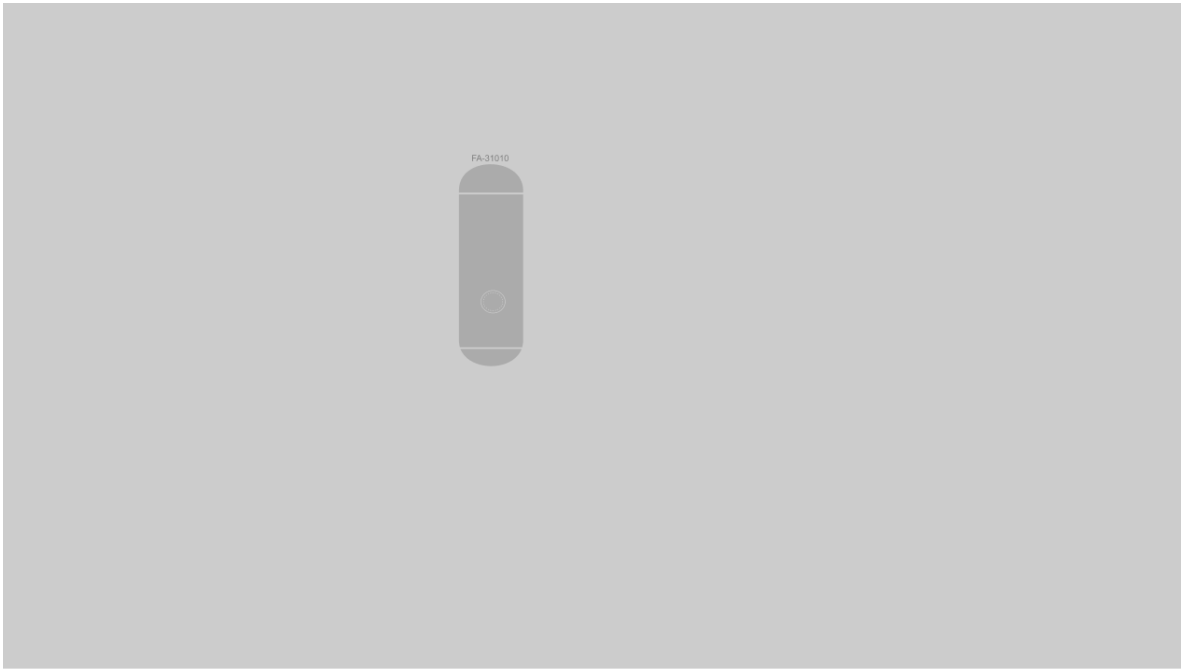


Figura 111. Algoritmo de dibujo para tanques versión 4.

```
menos = datos[datos['confidence'] > 0.80]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2 ), ((y * 2 )), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    print(txt)
    patron = r'FA-\S+'  # Expresión regular que busca "FA" seguido de cualquier carácter que no sea un espacio en blanco
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
    # Crea los elementos
    path1 = dwg.path(
        d='M14.0781 62.3311L69.197 106.838M34.2787 62.3311L89.3977 106.838M55.6337',
        fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path2 = dwg.path(
        d='M75.5458 62.3311L130.376 106.838M97.1894 62.3311L151.731 106.838M117.967',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    path3 = dwg.path(
        d='M138.456 62.3311L171.643 88.9891M159.523 62.3311L171.643 72.2989 ',fill='#ABABAB', stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    group.add(path2)
    group.add(path3)
    dwg.add(group)
```

Figura 112. Diagrama P&ID básico para detección de tanques versión 4.

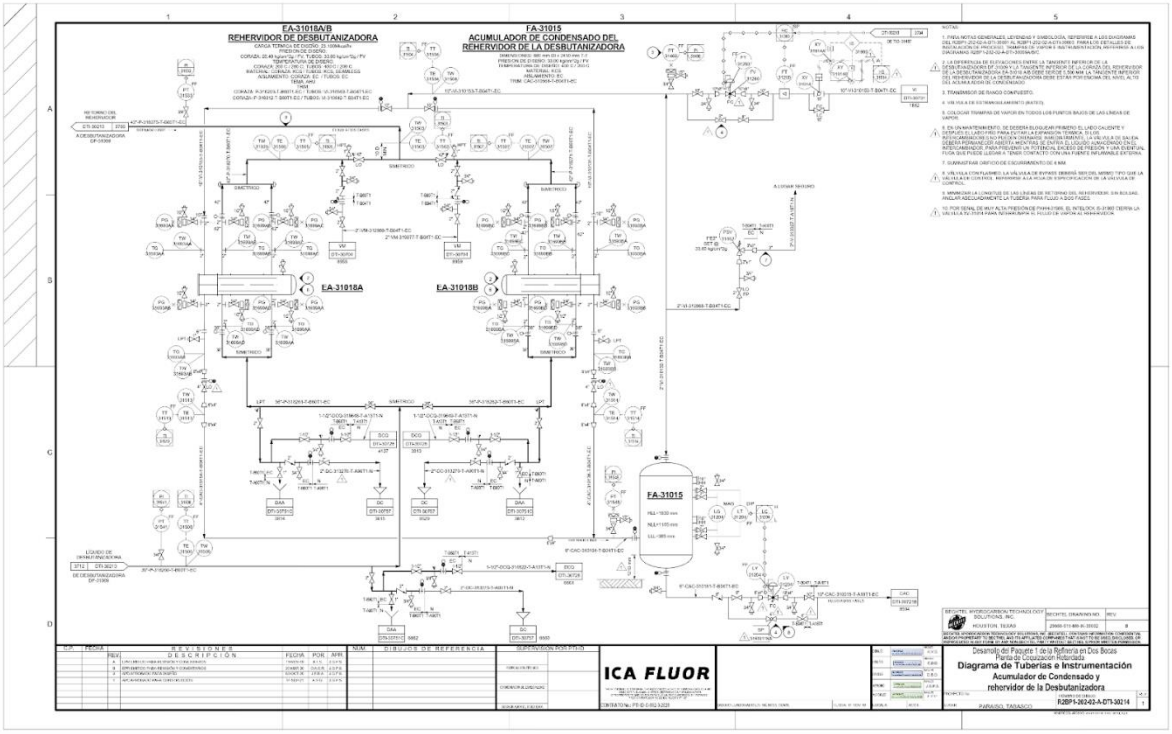
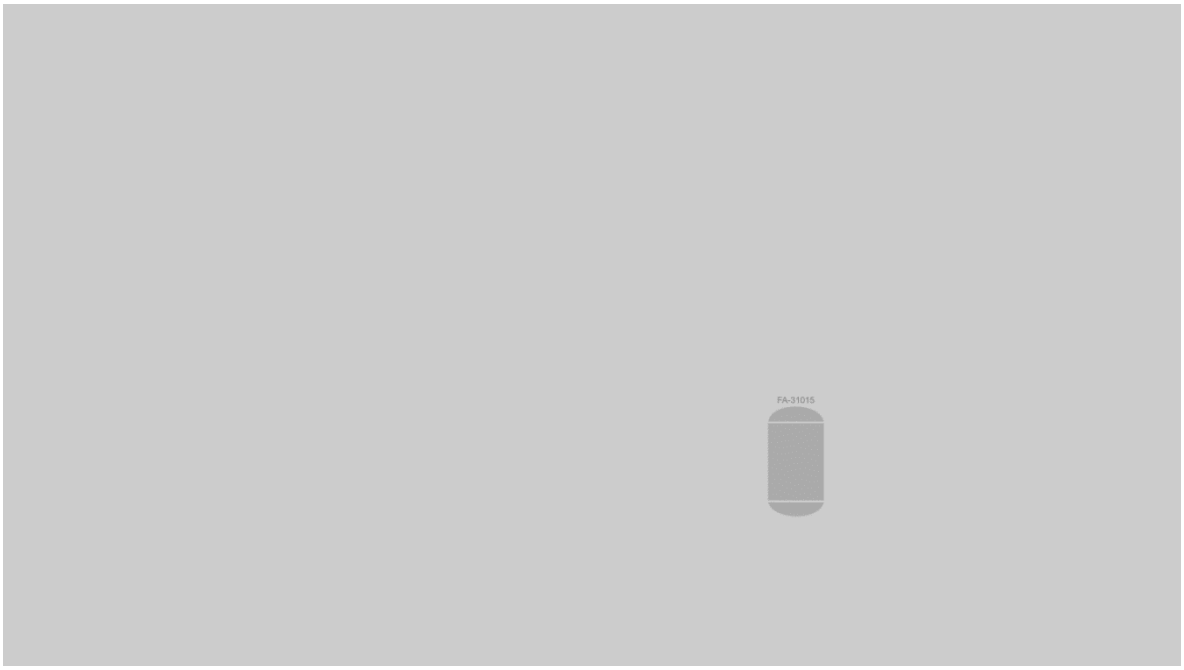


Figura 113. Plantilla HMI basada en la identificación de tanques V4.



Dibujo de la turbina, este elemento tiene los siguientes parámetros **(Fig.114-116)**:

- Porcentaje de exactitud: 90%, todas las variaciones tienen este mismo porcentaje.
- Texto detectado por el OCR: "GA-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "GA-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene diferentes figuras como rectángulos (rect) y path (líneas de recorrido)

Figura 114. Algoritmo de dibujo para turbinas

```
menos = datos[datos['confidence'] > 0.90]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'GA-\S+'
    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
    else:
        txt = ''
    grupo = dwg.add(dwg.g())
    # Crea los elementos
    grupo.add(dwg.rect(insert=(0.907234, 108.387), size=(171.62, 78.7716), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(174.195, 91.5), size=(65.1083, 112.544), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(508.842, 91.5), size=(65.1083, 112.544), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(574.641, 107.76), size=(94.0988, 80.0247), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(788.747, 1.39453), size=(373.269, 292.756), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(669.357, 136.731), size=(118.805, 22.0822), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(1162.43, 136.731), size=(52.5161, 22.0822), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(1235, 136.731), size=(53.4569, 22.0822), fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.path(
        d='M449.92 65.7971L449.998 65.823H450.079H511.163V230.871H450.079H450L449.924 230.895L321.298 272.814H240.844V22.731H321.296L449.92 65.7971Z',
        fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.path(
        d='M1207.23 126.829L1216.7 126.635V133.962V161.027V168.921H1207.66V161.027V160.527H1207.16H1197.62V134.462H1206.73H1207.23V133.962V126.829Z',
        fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.path(
        d='M1229.91 126.829L1220.45 126.635V133.962V161.027V168.921H1229.49V161.027V160.527H1229.99H1239.53V134.462H1230.41H1229.91V133.962V126.829Z',
        fill='#ABABAB', stroke='#CDCDCD'))
    grupo.add(dwg.rect(insert=(1278.67, 91.5), size=(119.263, 112.544), fill='#ABABAB', stroke='#CDCDCD'))
```

Figura 115. Diagrama P&ID básico para detección de turbinas

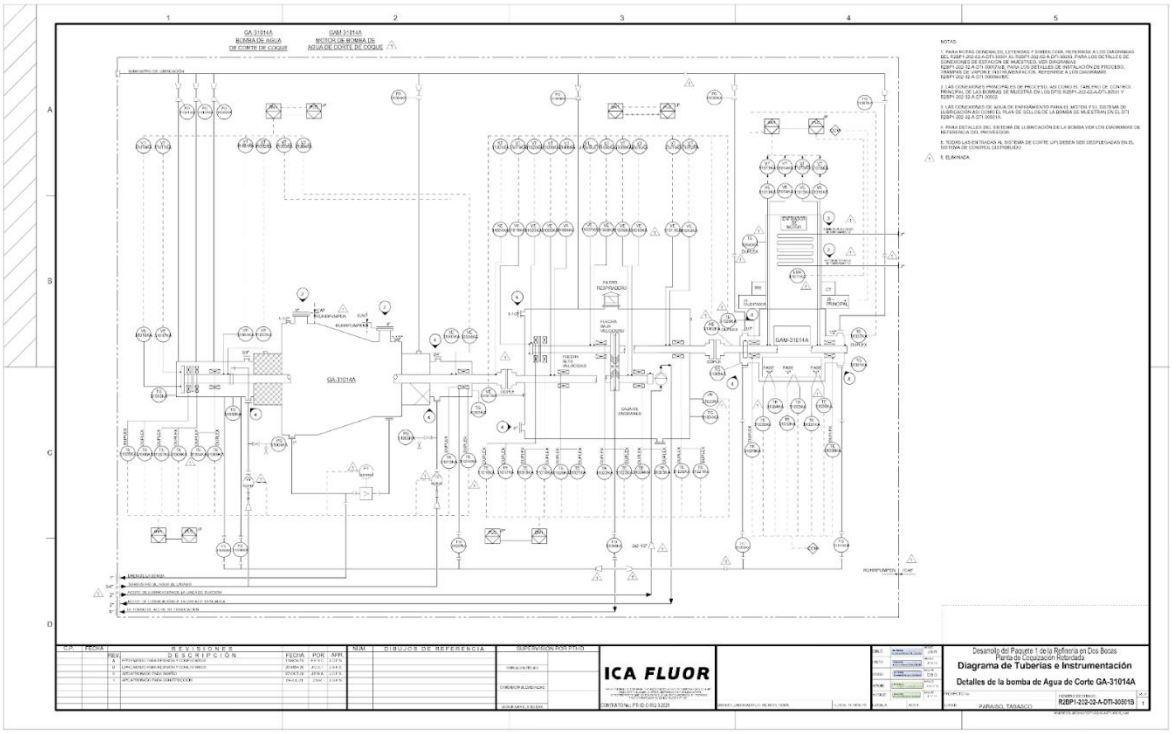
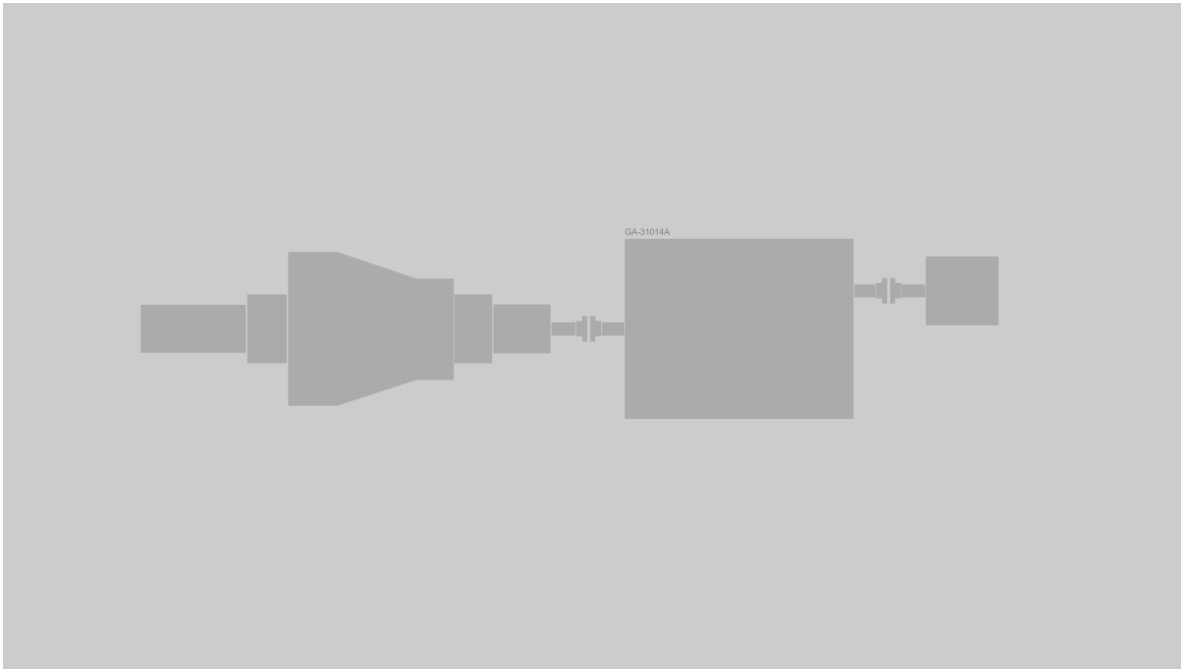


Figura 116. Plantilla HMI basada en la identificación de turbinas.



Dibujo del horno, este elemento tiene los siguientes parámetros **(Fig.117-119)**:

- Porcentaje de exactitud: 90%, todas las variaciones tienen este mismo porcentaje.
- Texto detectado por el OCR: "BA-\S+". Esta estructura indica que se busca específicamente los tres primeros caracteres "BA-", seguidos de uno o más que no sean espacios en blanco (\S+). Estos pueden ser letras, números o símbolos, pero no espacios.
- Grupo SVG: Este grupo contiene un elemento tipo path (líneas de recorrido) que logra ilustrar toda la figura.

Figura 117. Algoritmo de dibujo para hornos

```
menos = datos[datos['confidence'] > 0.90]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'BA-\S+'

    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
        # Crea los elementos

    path1 = dwg.path(d='M247.319 301.374H370.123V673.91H1.21313V301.374H124.923H125.423V300.874V1.27542'
                       'H246.819V300.874V301.374H247.319Z', fill="#ABABAB", stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    dwg.add(group)
    group.translate(x, y)
    texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
    dwg.add(texto)
```

Figura 118. Diagrama P&ID básico para detección de hornos

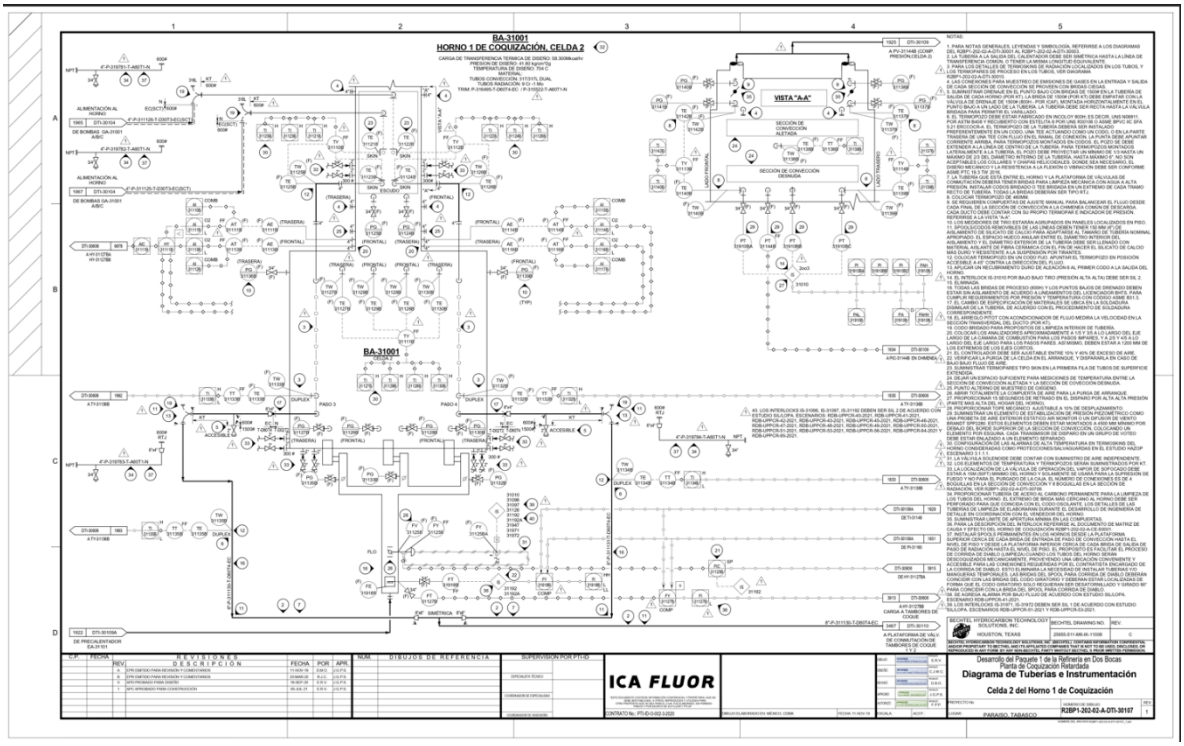
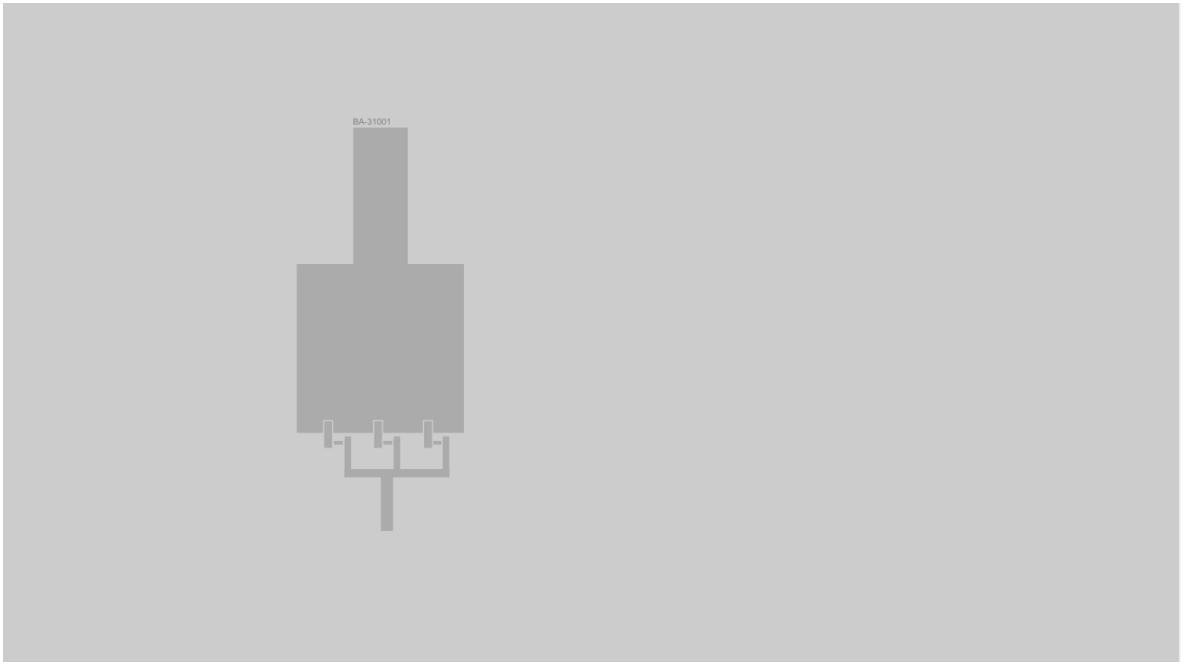


Figura 119. Plantilla HMI basada en la identificación de hornos.



Dibujo de las válvulas, este elemento tiene los siguientes parámetros **(Fig.120-122)**:

- Porcentaje de exactitud: 53%, todas las variaciones tienen este mismo porcentaje.
- Texto detectado por el OCR: por el momento este elemento no tiene reconocimiento de texto debido a que sus parámetros de detección son completamente diferentes.
- Grupo SVG: Este grupo contiene diferentes figuras de tipo path (líneas de recorrido) que logran representar de manera correcta todo el elemento.

Figura 120. Algoritmo de dibujo para válvulas.

```
menos = datos[datos['confidence'] > 0.90]
coo = menos[['xmin', 'ymin', 'xmax', 'ymax']].copy()
coo = coo.astype(int)
final = np.array(coo)
final2 = final.tolist()
filas, columnas = final.shape
# Crear un grupo para contener todos los elementos
group = svgwrite.container.Group()
# recorremos la matriz y creamos
for pos in final2:
    x, y, x1, y1 = pos[:4]
    recorte = img_pil.crop(((x * 2), ((y * 2)), (x1 * 2), ((y1 * 2))))
    txt = tess.image_to_string(recorte)
    patron = r'BA-\S+'

    resultado = re.search(patron, txt)
    if resultado:
        txt = resultado.group()
        group = dwg.add(dwg.g())
    else:
        txt = ''
        # Crea los elementos

    path1 = dwg.path(d='M247.319 301.374H370.123V673.91H1.21313V301.374H124.923H125.423V300.874V1.27542'
                       'H246.819V300.874V301.374H247.319Z', fill="#ABABAB", stroke='#CDCDCD', stroke_width='3')
    # Agrega los elementos al grupo
    group.add(path1)
    dwg.add(group)
    group.translate(x, y)
    texto = dwg.text(txt, insert=(x + 50, y), fill=fill_i, font_size=14, font_family=font_i)
    dwg.add(texto)
```

Figura 121. Diagrama P&ID básico para detección de válvulas.

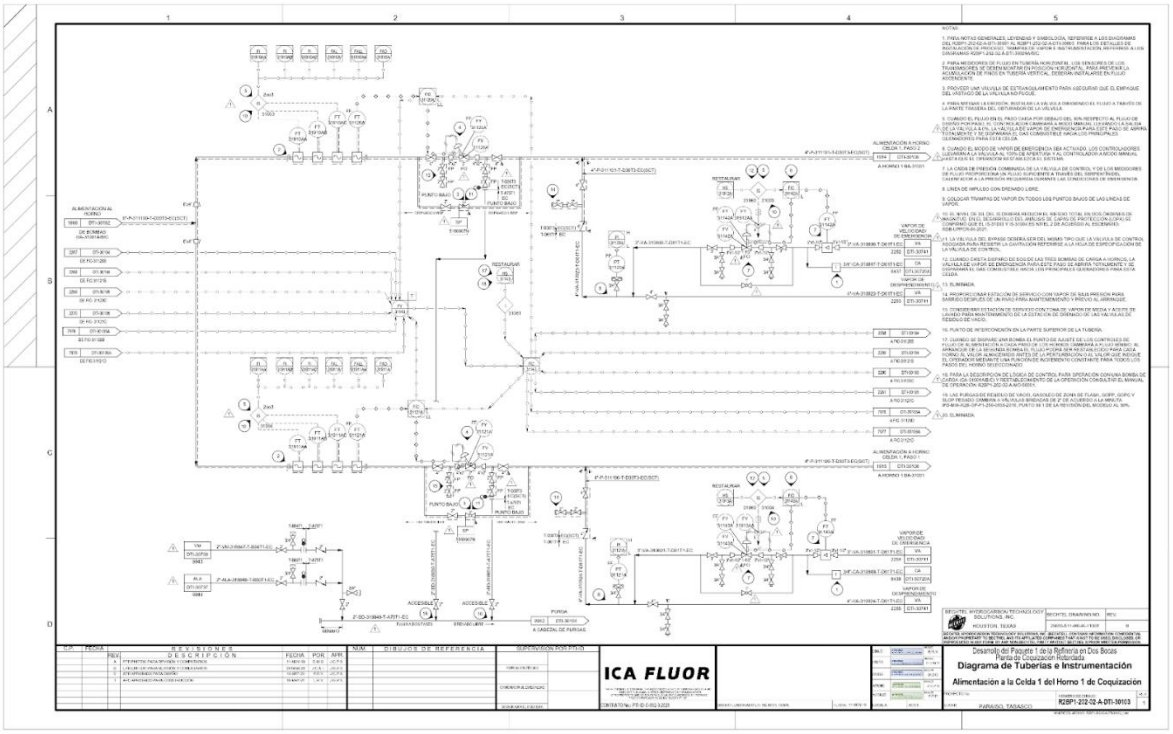


Figura 122. Plantilla HMI basada en la identificación de válvulas.

Durante la revisión de los algoritmos para generación de dibujos SVG por parte del asesor, se ha obtenido la aprobación con una observación relevante respecto a la capacidad de realizar cambios en el diseño de los elementos en el futuro, si así lo solicita el cliente. Se valora positivamente esta flexibilidad y disposición para adaptarse a posibles modificaciones.

Se ha acordado que, en caso de que sea necesario cambiar el diseño de algún elemento, esto no debería representar un problema significativo para el algoritmo desarrollado. En esencia, solo sería necesario realizar modificaciones en el código SVG correspondiente a cada elemento, lo cual se considera manejable y no afectaría de manera drástica el funcionamiento general del algoritmo.

Esta adaptabilidad y la capacidad de realizar cambios en el diseño de manera eficiente demuestran la versatilidad del sistema implementado y la disposición para satisfacer las necesidades y preferencias del cliente en cuanto a la representación gráfica de los elementos en el dibujo final.

Semana 10: Generación de líneas de conexión

Dando los últimos detalles a la herramienta se propone realizar el dibujo de las líneas de conexión, esto ayudará a mantener una coherencia entre los elementos, siendo estas líneas más un elemento guía para ayudar al diseñador un entendimiento del resultado final y no entorpecer el trabajo que debe ejecutar una vez la plantilla ha sido generada.

Se desarrolla un algoritmo donde se comienza importando las bibliotecas necesarias: numpy, svglib, reportlab y OpenCV. Estas bibliotecas proporcionan funciones y herramientas para procesar imágenes, manipular píxeles, trabajar con gráficos vectoriales y realizar operaciones de dibujo.

Posteriormente se lee un archivo SVG que contiene la imagen original y lo convierte en un objeto de dibujo utilizando la función svg2rlg de la biblioteca svglib. Luego, se renderiza este

objeto en una imagen PNG utilizando la función drawToPIL de la biblioteca reportlab, y se guarda en un archivo llamado 'imagen_f_convert.png'. A continuación, la imagen PNG recién creada se lee nuevamente utilizando la función imread de OpenCV, pero esta vez se carga en escala de grises (representada por el argumento 0 en la función). Esto facilita la manipulación de píxeles y el procesamiento posterior de la imagen.

Figura 123. Tratamiento y duplicado de imagen para generación de líneas.

```
import cv2
import numpy as np
from svglib.svglib import svg2rlg
from reportlab.graphics import renderPM

drawing = svg2rlg('imagen_f.svg')
# Renderizar el archivo SVG como imagen PNG
png_image = renderPM.drawToPIL(drawing)
png_image.save('imagen_f_convert.png')
img = cv2.imread('imagen_f_convert.png', 0)
new_img = np.full((1080, 1920), 205, dtype=np.uint8)
# Tamaño del lienzo
width = 1920
height = 1080
```

Se define una función llamada draw que se utilizará como controlador de eventos del mouse. Esta función se ejecutará cada vez que el usuario interactúe con la imagen en la ventana. Toma como argumentos el tipo de evento, las coordenadas (x, y) del mouse, las banderas y los parámetros adicionales. Dentro de la función draw, se utilizan las variables ix e iy para realizar un seguimiento de las coordenadas del punto inicial del trazo. Cuando el usuario hace clic con el botón izquierdo del mouse, se comprueba si ya se está dibujando una línea. Si es así, se calcula la distancia y la dirección entre los puntos inicial y final del trazo y se dibuja una línea recta entre ellos. Para asegurarse de que la línea dibujada tenga un ángulo recto (90 grados), se realiza una modificación en las coordenadas del punto final según la diferencia entre las coordenadas x e y. Si la diferencia en x es mayor que la diferencia en y, se ajusta la coordenada y del punto final

para que sea igual a la coordenada y del punto inicial, y viceversa. Esto garantiza que la línea siempre tenga un ángulo recto. Durante el proceso de dibujo de la línea, se utilizan las funciones line de OpenCV para dibujar la línea tanto en la imagen original (img) como en una nueva imagen en blanco (new_img). Se especifican los puntos inicial y final de la línea, el color (representado por una tupla RGB.

Además, se agrega la línea dibujada al objeto de dibujo original utilizando la función add de la biblioteca svglib. Esto asegura que las líneas también se reflejen en el archivo SVG original. Si el usuario hace clic con el botón derecho del mouse, se reinicia la imagen estableciendo todos los píxeles a negro (valor 0).) y el grosor de la línea.

Figura 124. Captura de eventos del mouse.

```
# Crear una función para manejar los eventos del mouse
def draw(event, x, y, flags, params):
    nonlocal ix, iy, drawing
    # Si se hace clic con el botón izquierdo, se dibuja un punto
    if event == cv2.EVENT_LBUTTONDOWN:
        if drawing:
            # Dibujar una línea recta entre los dos puntos
            dx, dy = x - ix, y - iy
            if abs(dx) > abs(dy):
                steps = abs(dx)
            else:
                steps = abs(dy)
            xstep = dx / steps
            ystep = dy / steps
            x1, y1 = ix, iy
            for i in range(steps):
                x2, y2 = int(round(x1 + xstep)), int(round(y1 + ystep))
                # Modificar la línea para cumplir con la restricción de 90 grados
                if abs(x2 - x1) > abs(y2 - y1):
                    y2 = y1
                else:
                    x2 = x1
                cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.line(new_img, (x1, y1-5), (x2, y2-5), (180, 180, 180), 2)
                x1, y1 = x2, y2
            dwg.add(dwg.line((ix, iy), (x2, y2), stroke='#ABABAB'))
            cv2.line(img, (x2, 0), (x2, height), (128, 128, 128), 1)
            cv2.line(img, (0, y2), (width, y2), (128, 128, 128), 1)
            drawing = False
        else:
            ix, iy = x, y
            drawing = True

    # Si se hace clic con el botón derecho, se reinicia la imagen
    elif event == cv2.EVENT_RBUTTONDOWN:
        img.fill(0)
```

Luego, se inicializan las variables drawing, ix e iy que se utilizarán en el proceso de dibujo. La variable drawing indica si se está dibujando una línea actualmente, mientras que ix e iy almacenan las coordenadas del punto inicial del trazo. A continuación, se crea una ventana de visualización utilizando la función namedWindow de OpenCV y se asocia la función draw al evento de clic del mouse utilizando la función setMouseCallback.

En el ciclo principal, se muestra la imagen en la ventana utilizando la función imshow de OpenCV. El ciclo se repite continuamente hasta que el usuario presione la tecla 'Esc' en el teclado. En ese momento, se rompe el ciclo y el programa continúa. Después del ciclo principal, se guarda la imagen modificada con las líneas dibujadas en un archivo llamado 'background_lineas.png' utilizando la función imwrite de OpenCV.

Figura 125. Visualización del generador de líneas.

```
# InicialDizar variables
drawing = False
ix, iy = -1, -1

# Crear la ventana y asociar la función del mouse a la imagen
cv2.namedWindow('image')
cv2.setMouseCallback('image', draw)

# Ciclo principal
while True:
    cv2.imshow('image', img)
    if cv2.waitKey(1) & 0xFF == 27:
        break
#
# Mostrar la imagen copiada con las líneas
cv2.imwrite('background_lineas.png', new_img)
```

A continuación, se presenta el proceso para realizar el dibujo, una vez que se ha completado la detección y se han cargado los elementos:

- Despliega una ventana de dibujo: Se muestra una interfaz gráfica donde se llevará a cabo el proceso de dibujo. **(Fig.126)**

- Generación de líneas: Se trazan líneas de forma manual, conectando los puntos extremos de los elementos detectados. El diseñador realiza este proceso de manera interactiva, dibujando las líneas en dirección horizontal y vertical para crear una estructura de interconexión entre los elementos. Para facilitar la distinción entre el momento de diseño y el resultado final, las líneas se destacan en un color negro. **(Fig.127)**

- Finalización del proceso: Una vez se han generado todas las líneas, se concluye el proceso presionando la tecla "Esc". En este momento, se generará el archivo SVG con el resultado final, donde los elementos están interconectados según las decisiones tomadas por el diseñador. **(Fig.128)**

Este proceso permite agilizar la creación del dibujo, generando las líneas de interconexión de manera eficiente y ajustándose a las decisiones de diseño tomadas por el diseñador.

Figura 126. Interfaz para dibujo de líneas.

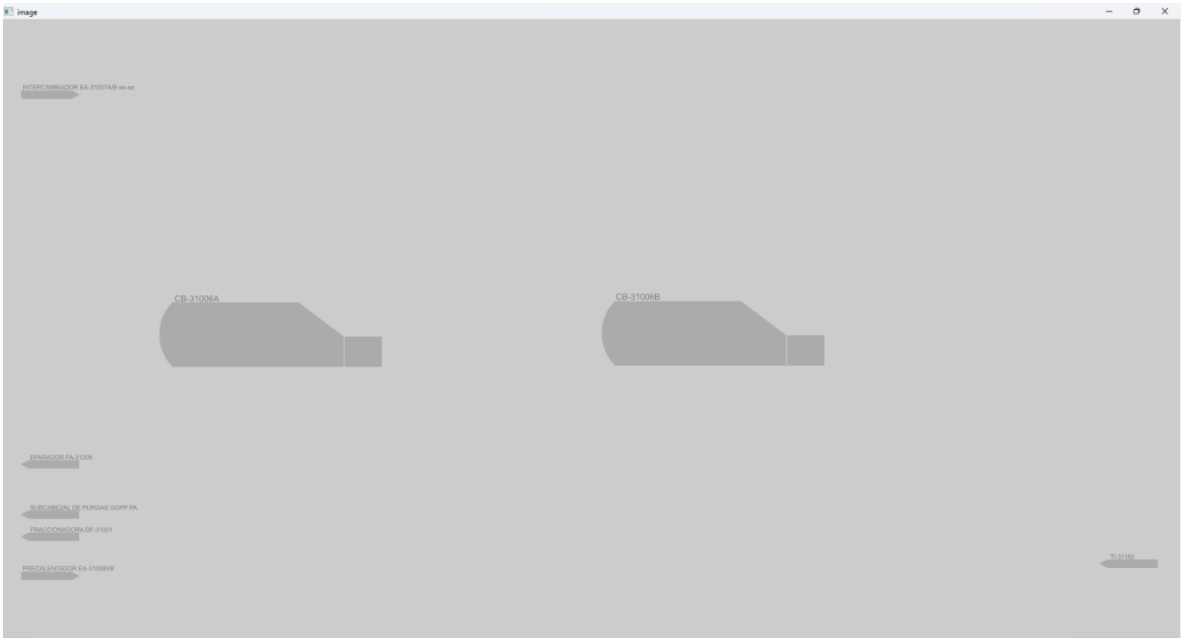


Figura 127. Dibujo de líneas en la interfaz gráfica.

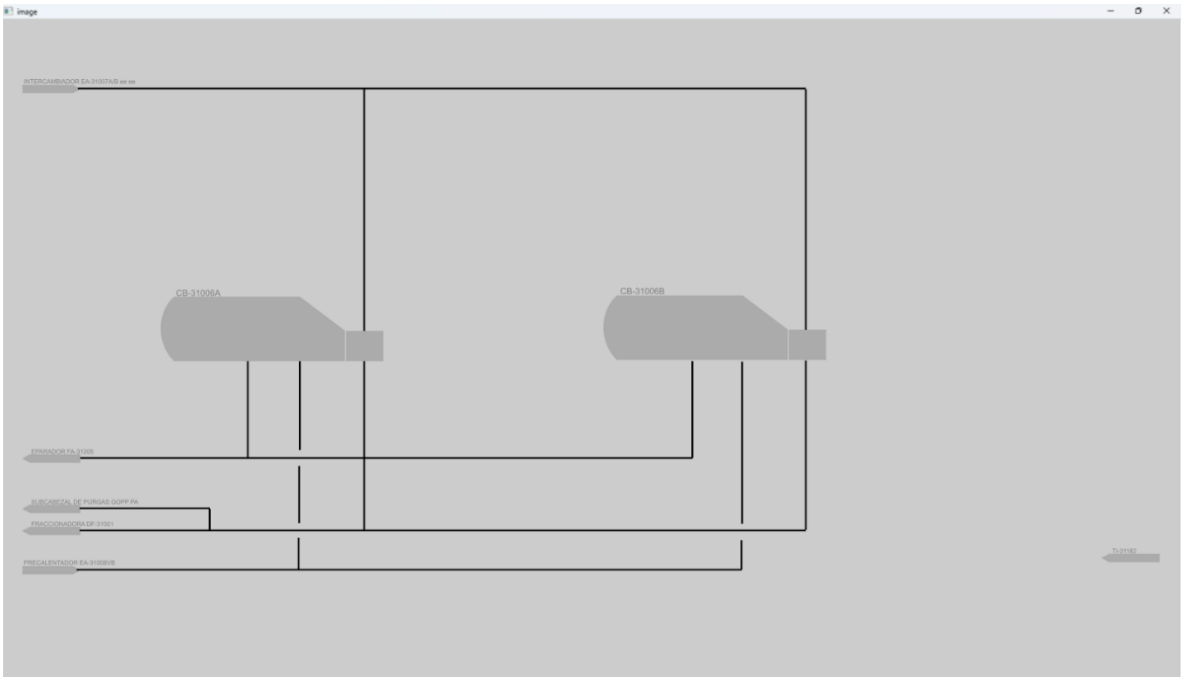
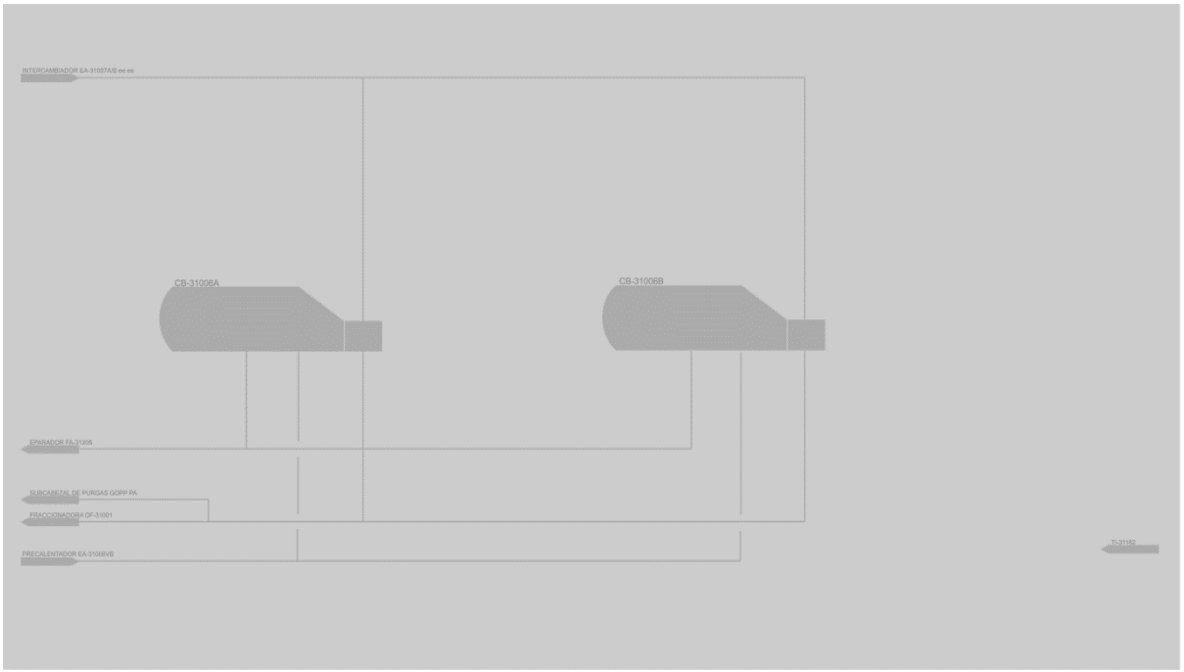


Figura 128. Resultado de plantilla HMI con líneas de conexión.



Nota: En la plantilla se puede observar que hay una bandera que no está conectada a ningún otro elemento. Esto se debe a que este objeto está relacionado con un elemento dinámico, como un medidor o indicador de una variable de proceso en tiempo real. En este caso, es preferible no realizar ningún dibujo de interconexión en esta etapa, ya que podría generar confusión al continuar con el proceso. Al omitir la conexión de este tipo de elementos dinámicos en la etapa de dibujo, se asegura que la representación gráfica se mantenga coherente y no genera malentendidos en cuanto a la interconexión de los elementos estáticos en la plantilla.

Al finalizar la semana y llevar a cabo la revisión programada, se logra un avance significativo en el desarrollo y se obtiene una respuesta muy favorable por parte del asesor. Hasta el momento, el desarrollo ha sido aprobado. Sin embargo, en las próximas dos semanas se llevarán a cabo pruebas de calidad adicionales utilizando diversos gráficos y con la participación de diferentes personas utilizando la herramienta. Estas pruebas serán cruciales para obtener un veredicto de aprobación y garantizar la calidad del producto.

3.2.4. Fase 4: Evaluación de la herramienta

Es la última etapa del desarrollo en la que se plantean dos actividades principales. La primera actividad consiste en unificar todos los algoritmos desarrollados a lo largo de las 10 semanas anteriores, creando un código unificado que pueda recibir la ruta de una imagen y automáticamente detecta y dibuja los elementos correspondientes. Posteriormente, se realizará una intervención manual para agregar las líneas de acuerdo con la interpretación de cada diseñador. Esta tarea no debería llevar más de un día y se realizará durante dos semanas consecutivas todo el proceso de evaluación, se evaluará la confiabilidad de la herramienta en base a tres factores fundamentales: exactitud, tiempo y comprensión. Cada uno de estos factores tendrá un peso que refleja su importancia a la hora de determinar si un gráfico es confiable o no. Cada diseñador evaluará cada gráfico, lo que suma un total de ocho diseñadores encargados de

realizar una versión sin la herramienta y otra versión con la ayuda de esta. Al final del proceso, se contará con una comparativa entre ambos resultados para analizar y evaluar el desempeño de la herramienta en cada caso.

Semana 11: Unificación y evaluación de la herramienta

La primera tarea de esta semana es unificar todos los algoritmos desarrollados para esto se crea un código donde se comienza importando el módulo svgwrite y algunas funciones desde un archivo llamado funciones que contiene todos los algoritmos personalizados para detectar y dibujar elementos específicos en una imagen.

Se define una ruta de archivo llamada “ruta” que apunta a una imagen en el sistema de archivos local. Esta imagen será el punto de partida para las operaciones de detección y dibujo. A continuación, se definen las dimensiones del lienzo (canvas) en el cual se creará el archivo SVG. Las dimensiones se establecen en 1920 píxeles de ancho y 1080 píxeles de alto. Se crea un objeto dwg de la clase svgwrite.Drawing con el nombre de archivo 'imagen_f.svg' y las dimensiones especificadas. Este objeto representa el archivo SVG en el cual se realizan las operaciones de dibujo. Se dibuja un cuadrado en el lienzo utilizando el método rect del objeto dwg. El cuadrado abarca toda el área del lienzo y se rellenará con el color '#CCCCCC'. Luego, se agrega el cuadrado al objeto dwg utilizando el método add.

Posteriormente, se realizan una serie de llamadas a diferentes funciones de detección y dibujo. Estas son llamadas con el objeto dwg y la ruta de la imagen como argumentos. Cada una se encarga de detectar un tipo específico de elemento en la imagen (por ejemplo, banderas, válvulas, etc.) y dibujarlo en el archivo SVG representado por el objeto dwg. Cada vez que se llama a una función, se actualiza el objeto dwg con los elementos detectados y dibujados. Al final de todas las llamadas a las funciones de detección y dibujo, se invoca la función draw_svg_lines2 con el objeto dwg como argumento. Esta se encarga de dibujar líneas adicionales en el archivo SVG.

Finalmente, se guarda el archivo SVG utilizando el método save del objeto dwg. El archivo se guarda con el nombre 'imagen_f.svg' en el directorio de trabajo actual. El resultado se puede observar en la siguiente figura:

Figura 129. Algoritmo principal de ejecución de la herramienta.

```
import svgwrite
from funciones import detectar_banderas, detectar_cb, detectar_ea, detectar_ec, detectar_fa, detectar_fa2, detectar_ga
from funciones import detectar_horno, detectar_tanques, detectar_valve,draw_svg_lines2
#imagen
ruta = r"C:/Users/Camil/Documents/Pasantia" \
       r"/pythonProject/train/ilovepdf_merged_page-0093.jpg"
# Defino las dimensiones del lienzo
width, height = 1920, 1080
# Crear el archivo SVG y el grupo principal
dwg = svgwrite.Drawing('imagen_f.svg', size=(width, height))
# Dibujar el cuadrado
square = dwg.rect((0, 0), (width, height), fill='#CCCCCC')
dwg.add(square)
dwg = detectar_banderas(dwg, ruta)
dwg = detectar_valve(dwg, ruta)
dwg = detectar_cb(dwg, ruta)
dwg = detectar_ea(dwg, ruta)
dwg = detectar_ec(dwg, ruta)
dwg = detectar_fa(dwg, ruta)
dwg = detectar_fa2(dwg, ruta)
dwg = detectar_ga(dwg, ruta)
dwg = detectar_horno(dwg, ruta)
dwg = detectar_tanques(dwg, ruta)
dwg = draw_svg_lines2(dwg)
# Guarda el archivo SVG
dwg.save()
```

Las bibliotecas importadas para el desarrollo de las funciones mencionadas anteriormente incluyen:

- cv2: para el procesamiento de imágenes con OpenCV.
- PIL: para la manipulación de imágenes.
- pytesseract: para el reconocimiento óptico de caracteres.
- re: para trabajar con expresiones regulares.
- svgwrite: para generar archivos SVG.
- svglib y renderPM: para convertir archivos SVG en imágenes de mapa de bits.
- mysql.connector: para trabajar con bases de datos MySQL.
- qlalchemy.create_engine: para interactuar con bases de datos SQL en general.

Las funciones desarrolladas son:

- detectar_banderas: utilizada para detectar banderas en las imágenes.
- detectar_cb: utilizada para detectar elementos de control y bloqueo en las imágenes.
- detectar_ea: utilizada para detectar elementos de alarma en las imágenes.
- detectar_ec: utilizada para detectar elementos de control eléctrico en las imágenes.
- detectar_fa: utilizada para detectar elementos de flujo en las imágenes.
- detectar_ga: utilizada para detectar generadores de aire en las imágenes.
- detectar_horno: utilizada para detectar hornos en las imágenes.
- detectar_tanques: utilizada para detectar tanques en las imágenes.
- draw_svg_lines2: utilizada para dibujar líneas SVG en función de los resultados obtenidos.
- detectar_valve: utilizada para detectar válvulas en las imágenes.

Estas funciones se utilizan en el algoritmo principal, al cual se le deben pasar los argumentos "ruta" y "dwg". El resultado de ejecutar el algoritmo es el objeto "dwg" con los elementos identificados en caso de que se hayan encontrado.

Figura 130. Algoritmo de alojamiento de las funciones para detección.

```
# Importamos librerías
import numpy as np
import pandas as pd
import torch
import cv2
from PIL import Image, ImageDraw, ImageFont
import pytesseract as tess
import re
import svgwrite
from svglib.svglib import svg2rlg
from reportlab.graphics import renderPM
import mysql.connector
from sqlalchemy import create_engine

def detectar_banderas(dwg, ruta):...

def detectar_cb(dwg, ruta):...

def detectar_ea(dwg, ruta):...

def detectar_ec(dwg, ruta):...

def detectar_fa(dwg, ruta):...

def detectar_ga(dwg, ruta):...

def detectar_horno(dwg, ruta):...

def detectar_tanques(dwg, ruta):...

def draw_svg_lines2(dwg):...

def detectar_valve(dwg, ruta):...
```

Cada diseñador se encargará de evaluar y realizar una versión de cada gráfico de manera manual y otra versión con la ayuda de la herramienta. Esto nos dará un total de 8 evaluaciones por gráfico, lo que resulta en un total de 160 evaluaciones en total. Con esta distribución, se busca tener una muestra representativa de la población de gráficos y obtener una comparativa visual entre las versiones realizadas con y sin la herramienta, permitiendo evaluar la confiabilidad de esta en términos de exactitud, tiempo y comprensión.

Se proponen los siguientes indicadores para evaluar la efectividad de un gráfico generado por la herramienta:

- Porcentaje de objetos detectados: Se establecen los elementos que se deben identificar y se registra la cantidad de objetos detectados para cada gráfico. Se calcula dividiendo la cantidad de objetos detectados por la cantidad total de objetos esperados. Este indicador muestra la capacidad de la herramienta para identificar correctamente los elementos deseados en el gráfico.

- Cantidad de objetos indeseados: Se realiza un conteo de los objetos indeseados presentes en la plantilla generada. Este indicador permite identificar la presencia de elementos no deseados en el gráfico y puede afectar la calidad y claridad de la representación. En la evaluación, se resta un 5% de exactitud por cada objeto indeseado detectado.

- Porcentaje de textos detectados: Se definen los textos que se deben identificar y se registra la cantidad de textos detectados en cada gráfico. Se calcula dividiendo la cantidad de textos detectados por la cantidad total de textos esperados. Este indicador evalúa la capacidad de la herramienta para reconocer correctamente los textos relevantes en el gráfico.

Cada indicador tiene un peso asignado para reflejar su importancia en la evaluación global. Estos indicadores nos permiten medir la exactitud y calidad de la herramienta al generar gráficos, identificar áreas de mejora y proporcionar una evaluación integral de su desempeño. En la siguiente tabla se pueden observar las fórmulas matemáticas para el indicador de exactitud:

Tabla 4. Indicadores para evaluar la exactitud.

Indicadores	Fórmula	Peso	Subtotal
%objetos detectados	$\frac{cantidad\ de\ objetos\ detectados}{número\ de\ objetos\ que\ se\ deben\ detectar}$	0.6	*Fórmula * Peso*
Cantidad de objetos indeseados	$cantidad\ de\ objetos\ indesados\ en\ la\ detección$	0.05	*- (Fórmula * Peso)*
%textos detectados	$\frac{cantidad\ de\ textos\ detectados}{número\ de\ textos\ que\ se\ deben\ detectar}$	0.35	*Fórmula * Peso*
Total		1.0	*Suma de subtotales*

Una vez obtenida la exactitud para cada gráfico, se puede utilizar una escala de evaluación para clasificar el desempeño de la herramienta. A continuación, se presenta una escala propuesta:

- Excelente: 90% - 100%
- Bueno: 80% - 89%
- Regular: 60% - 79%
- Mal: 0% - 60%

Para evaluar la mejora en el tiempo de generación de los gráficos, se propone comparar el tiempo de realización de un gráfico utilizando la herramienta con el tiempo necesario para crear el mismo gráfico sin utilizarla. A partir de esta comparación, se puede calcular el porcentaje de mejora en términos de tiempo.

El cálculo del porcentaje de mejora se realiza mediante la siguiente fórmula:

$$\text{Porcentaje de mejora} = \frac{\text{tiempo desarrollo sin herramienta} - \text{tiempo desarrolló con herramienta}}{\text{tiempo desarrollo sin herramienta}}$$

Este indicador nos permite cuantificar la eficiencia y rapidez obtenida al utilizar la herramienta en comparación con los métodos tradicionales de creación de gráficos. Un mayor porcentaje de mejora indica una mayor eficiencia en el proceso de generación de los gráficos. Escala propuesta para la evaluación de la mejora en el tiempo:

- Excelente: mayor al 30%
- Bueno: 20% - 29%
- Regular: 10% - 19%
- Mal: 0% - 10%

La comprensión del gráfico se refiere a cómo cada diseñador percibe y asimila el resultado de la plantilla generada. A continuación, se presenta una escala de evaluación para la compresión del gráfico:

- Excelente: 100%
- Bueno: 80% - 99%
- Regular: 60% - 79 %
- Mal: Menor al 60%

En esta escala, se considera que la compresión del gráfico es excelente cuando el diseñador puede entender completamente el resultado de la plantilla, sin ninguna ambigüedad o confusión. Un nivel bueno de compresión implica que el diseñador tiene una legibilidad adecuada del gráfico, aunque puede haber algunos aspectos que requieran aclaración o explicación adicional.

Una compresión regular indica que el diseñador tiene cierta dificultad para comprender algunos aspectos del gráfico, lo que puede requerir una mayor explicación o clarificación. Por último, una compresión clasificada como mala significa que el diseñador tiene dificultades significativas para entender el gráfico, lo que puede dificultar su trabajo y requerir una revisión exhaustiva o una explicación detallada.

Mientras que la confiabilidad se determina mediante la suma ponderada de los indicadores anteriores: exactitud, tiempo y compresión. Cada indicador tiene asignado un peso que refleja su importancia en la evaluación general. En este caso, se asigna un peso del 70% a la exactitud y un peso del 15% tanto al tiempo como a la compresión, al asignar pesos diferenciados, se enfatiza la importancia de la exactitud sin descuidar el tiempo de diseño y la compresión del gráfico. Esto garantiza una evaluación equilibrada y abarca los aspectos esenciales para una herramienta confiable en la generación de gráficos.

El peso del 70% para la exactitud refleja la prioridad de obtener resultados precisos y confiables en la detección y dibujo de los elementos en el gráfico. Se considera que la exactitud es el factor más crítico y determinante en la confiabilidad de la herramienta.

Los pesos del 15% para el tiempo y la compresión reconocen la importancia de ambos aspectos en la evaluación general. El tiempo de diseño es un factor relevante, ya que una herramienta eficiente y rápida puede agilizar el proceso de generación de gráficos. La compresión de la plantilla también es significativa, ya que un diseño claro y comprensible facilita el trabajo de los diseñadores y evita posibles confusiones o retrasos.

Escala propuesta para la evaluación de la confiabilidad:

- Excelente: 90% - 100%
- Bueno: 80% - 89%
- Regular: 70% - 79%
- Mal: 0% - 70%

Semana 12: Recolección y manejo de información

Para la recolección de toda la información, se solicita a los diseñadores registrar los siguientes datos para cada gráfico evaluado:

- Tiempo de desarrollo con la herramienta: Los diseñadores deben indicar el tiempo en horas que les tomó completar la versión del gráfico utilizando la herramienta.
- Tiempo de desarrollo sin la herramienta: Los diseñadores deben indicar el tiempo en horas que les tomó completar la versión del gráfico sin utilizar la herramienta.
- Nivel de comprensión de la plantilla suministrada por la herramienta: Los diseñadores deben evaluar en una escala del 1 al 100% el nivel de comprensión de la plantilla proporcionada por la herramienta, donde 1 indica una baja comprensión y 100 indica una alta comprensión.
- Cantidad de objetos a detectar: Se especifica la cantidad total de objetos que se deben detectar en el gráfico.
- Cantidad de objetos detectados: Los diseñadores deben registrar la cantidad de objetos que han sido detectados correctamente en el gráfico.
- Cantidad de objetos indeseados: Se realiza un conteo de los objetos indeseados presentes en el gráfico.
- Cantidad de textos a detectar: Se establece la cantidad total de textos que se deben detectar en el gráfico.
- Cantidad de textos detectados: Los diseñadores deben registrar la cantidad de textos que han sido detectados correctamente en el gráfico.

Los datos suministrados por los diseñadores recolectados en la siguiente tabla:

Tabla 5. Registro de datos para evaluación.

Nombre Gráfico	Tiempo Herramienta	Tiempo Manual	Compresión	Objetos por detectar	Objetos dectectados	Objetos indeseados	Textos por detectar	Textos detectados
DTI-30103	10	15	100.00%	17.00	17.00	0.00	15.00	15.00
DTI-30107	13	18	90.00%	15.00	15.00	0.00	15.00	15.00
DTI-30139	12	16	70.00%	14.00	13.00	0.00	13.00	11.00
DTI-30142A	8	14	90.00%	6.00	6.00	0.00	6.00	6.00
DTI-30143	18	25	90.00%	8.00	8.00	0.00	8.00	8.00
DTI-30145	15	20	80.00%	12.00	12.00	0.00	12.00	12.00
DTI-30146A	7	10	80.00%	9.00	8.00	0.00	7.00	7.00
DTI-30151	20	35	90.00%	21.00	18.00	0.00	18.00	16.00
DTI-30154	15	25	80.00%	9.00	8.00	0.00	6.00	6.00
DTI-30162	18	23	100.00%	18.00	18.00	0.00	17.00	17.00
DTI-30203A	10	15	90.00%	6.00	6.00	1.00	6.00	6.00
DTI-30204	15	22	100.00%	10.00	9.00	0.00	9.00	9.00
DTI-30205	18	25	90.00%	9.00	9.00	1.00	8.00	8.00
DTI-30210	14	17	100.00%	9.00	6.00	0.00	6.00	6.00
DTI-30214	15	18	70.00%	6.00	6.00	0.00	6.00	4.00
DTI-30225	20	26	70.00%	5.00	5.00	0.00	5.00	5.00
DTI-30226	14	20	70.00%	8.00	8.00	0.00	6.00	6.00
DTI-30229	10	13	80.00%	7.00	7.00	0.00	5.00	5.00
DTI-30304A	12	15	100.00%	10.00	10.00	0.00	10.00	10.00
DTI-30501B	32	40	70.00%	1.00	1.00	0.00	1.00	1.00
DTI-30103	12	18	90.00%	17.00	17.00	0.00	15.00	15.00

Nota: Esta tabla solo muestra la información proporcionada por uno de los diseñadores, en los anexos se adjuntará el archivo de Excel con todos los datos.

Una vez se tienen los datos registrados, se procede a realizar los cálculos correspondientes para determinar la confiabilidad de cada gráfico. Estos cálculos se realizan de forma individual para cada pantalla de un diseñador, después de calcular la confiabilidad, se obtiene un promedio ponderado de esta para el conjunto de gráficos evaluados. **(Tabla.6)**

El resultado final será un indicador de confiabilidad general que refleje el desempeño promedio de la herramienta por cada gráfico en términos de detección de objetos, detección de textos, nivel de comprensión y eficiencia en el tiempo de desarrollo. **(Tabla.7)**

Tabla 6. Cálculo de confiabilidad por diseñador.

Nombre Gráfico	Diseñador	% Tiempo	Compresión	Exactitud	Confiabilidad
DTI-30103	1	33.33%	100.00%	100.00%	90.00%
DTI-30107	1	27.78%	90.00%	100.00%	87.67%
DTI-30139	1	25.00%	70.00%	89.56%	76.94%
DTI-30142A	1	42.86%	90.00%	100.00%	89.93%
DTI-30143	1	28.00%	90.00%	100.00%	87.70%
DTI-30145	1	25.00%	80.00%	100.00%	85.75%
DTI-30146A	1	30.00%	80.00%	93.33%	81.83%
DTI-30151	1	42.86%	90.00%	86.98%	80.82%
DTI-30154	1	40.00%	80.00%	93.33%	83.33%
DTI-30162	1	21.74%	100.00%	100.00%	88.26%
DTI-30203A	1	33.33%	90.00%	95.00%	85.00%
DTI-30204	1	31.82%	100.00%	94.00%	85.57%
DTI-30205	1	28.00%	90.00%	95.00%	84.20%
DTI-30210	1	17.65%	100.00%	80.00%	73.65%
DTI-30214	1	16.67%	70.00%	86.67%	73.67%
DTI-30225	1	23.08%	70.00%	100.00%	83.96%
DTI-30226	1	30.00%	70.00%	100.00%	85.00%
DTI-30229	1	23.08%	80.00%	100.00%	85.46%
DTI-30304A	1	20.00%	100.00%	100.00%	88.00%
DTI-30501B	1	20.00%	70.00%	100.00%	83.50%

Nota: Cada diseñador tiene los mismos 20 gráficos, aquí solo se expone un diseñador para no saturar el documento, en el libro Excel de anexos se puede observar toda la información.

Tabla 7. Cálculo de confiabilidad por gráfico.

Nombre Gráfico	Confiabilidad	Escala
DTI-30103	88.86%	Bueno
DTI-30107	84.59%	Bueno
DTI-30139	79.26%	Regular
DTI-30142A	90.70%	Excelente
DTI-30143	86.69%	Bueno
DTI-30145	86.32%	Bueno
DTI-30146A	82.47%	Bueno
DTI-30151	80.86%	Bueno
DTI-30154	82.94%	Bueno
DTI-30162	86.20%	Bueno
DTI-30203A	85.33%	Bueno
DTI-30204	84.03%	Bueno
DTI-30205	79.89%	Regular
DTI-30210	44.36%	Mal
DTI-30214	76.48%	Regular
DTI-30225	86.10%	Bueno
DTI-30226	86.21%	Bueno
DTI-30229	86.29%	Bueno
DTI-30304A	86.61%	Bueno
DTI-30501B	86.38%	Bueno

Después de revisar los datos y discutirlos con el asesor, se concluye que la herramienta cumple con el objetivo planteado, aunque se identifican oportunidades de mejora. Al promediar la confiabilidad obtenida en cada gráfico, se obtiene un total de 82,53%, lo cual clasifica la herramienta en la escala "Bueno". Esto indica que ha demostrado ser efectiva en términos de detección de objetos, detección de textos, nivel de comprensión y eficiencia en el tiempo de desarrollo, pero aún existen áreas en las que se puede seguir trabajando para alcanzar una mayor confiabilidad. Es importante tener en cuenta que esta evaluación inicial proporciona una visión general de la confiabilidad de la herramienta, pero también se deben considerar los

comentarios y sugerencias recibidos durante el proceso de evaluación para continuar mejorando y optimizando su rendimiento. Con base en estos resultados, se pueden identificar aspectos específicos que requieren atención y se pueden implementar acciones correctivas para aumentar aún más el rendimiento de la herramienta en futuras iteraciones.

A continuación, se documentan los resultados más relevantes en el ejercicio, es decir las plantillas que generan particularidades o exponen casos especiales que requieren una mención especial. Para cada plantilla se muestra primero el diagrama P&ID en el cual fue basado su diseño, posteriormente el resultado de la herramienta y por último la versión generada por el diseñador.

Figura 131. P&ID gráfico DTI-30142A.

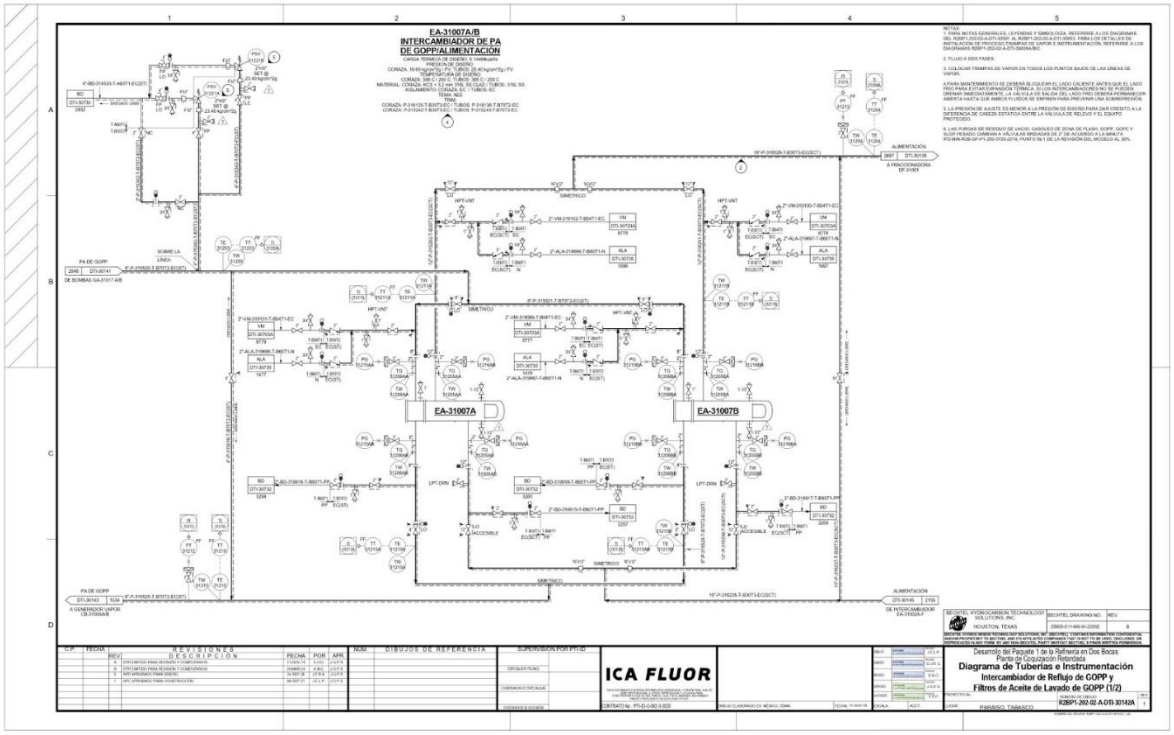


Figura 132. Plantilla HMI gráfico DTI-30142A.

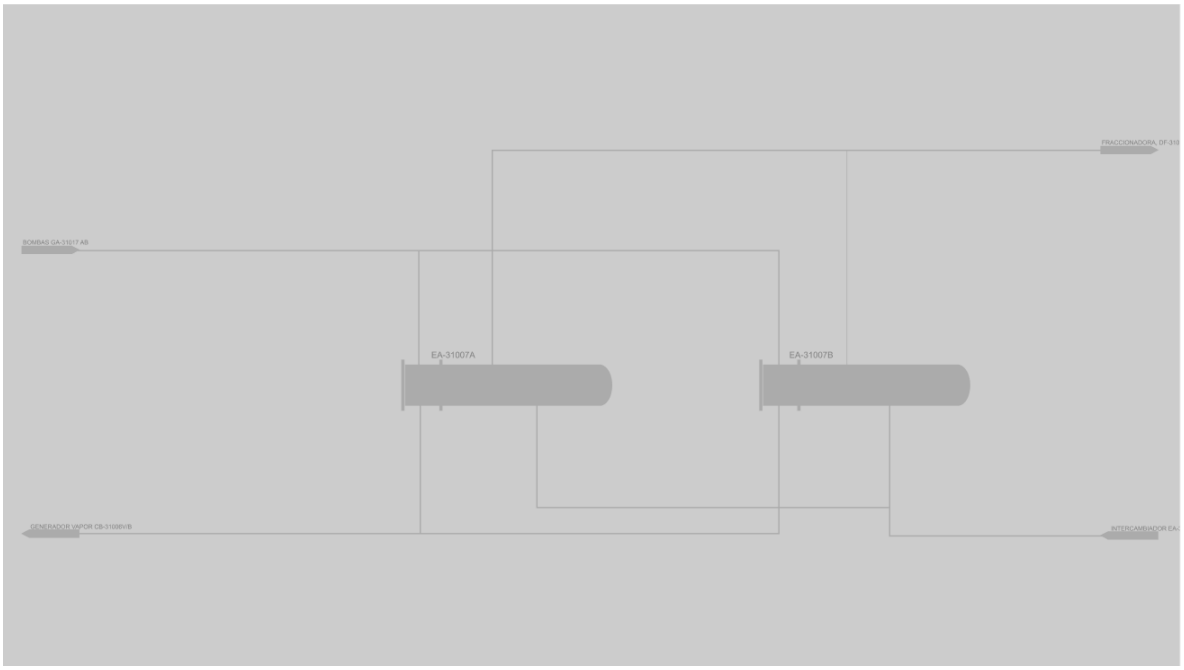
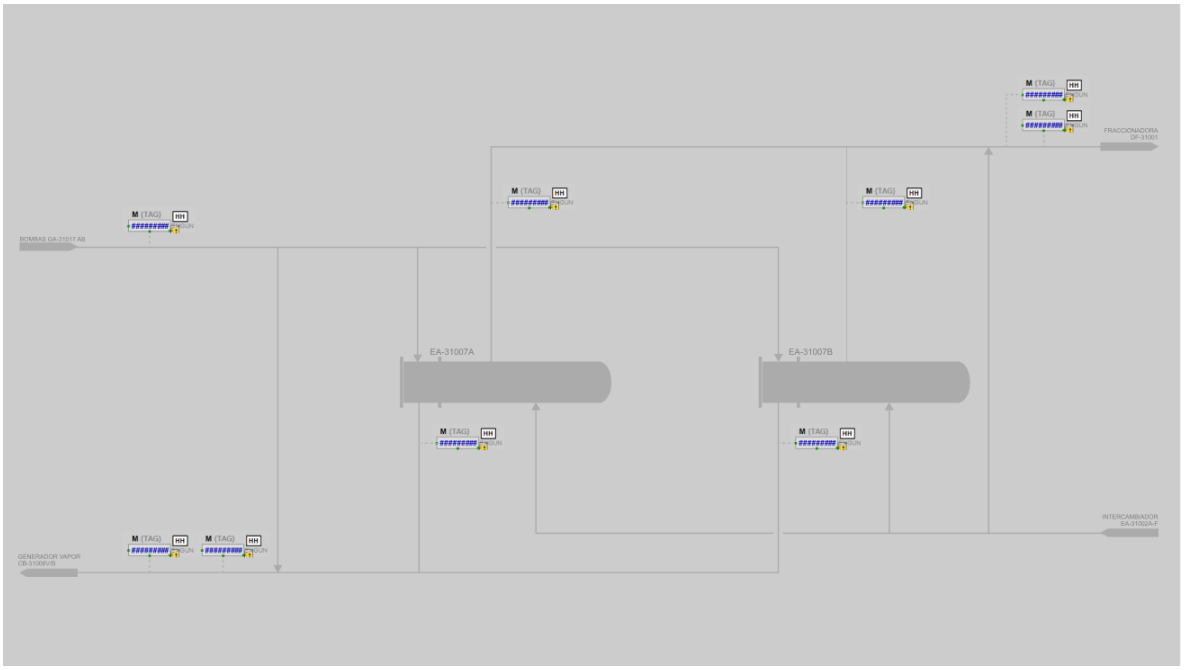


Figura 133. HMI gráfico DTI-30142A.



Este gráfico obtuvo el porcentaje de confiabilidad más alto y demuestra un excelente rendimiento de la herramienta. Se observa que en general que esta funciona muy bien, ya que los elementos estáticos se encuentran correctamente ubicados y la plantilla generada es comprensible. Además, se destaca que el trabajo necesario y el tiempo requerido para obtener el resultado deseado son bajos.

Con base en esta observación, se puede concluir que la herramienta funciona de manera excelente para gráficos con una densidad de elementos media. Esto significa que en casos donde los gráficos presentan una cantidad moderada de elementos, es altamente efectiva y eficiente, permitiendo obtener resultados precisos y en un tiempo razonable.

Es importante destacar un evento en el cual se agregó un elemento no deseado en un gráfico, a pesar de que la confiabilidad general de la herramienta se clasifica como "Bueno". Este incidente resalta la necesidad de seguir mejorando la precisión y la capacidad de detección de la herramienta, incluso en casos donde su desempeño general es satisfactorio.

La detección errónea de un elemento adicional en el costado derecho de la imagen puede deberse a diversas razones, como la presencia de características similares a los elementos deseados o limitaciones en el algoritmo de detección utilizado. Es importante analizar este caso particular y revisar los resultados obtenidos para comprender las posibles causas y determinar qué ajustes o mejoras pueden implementarse para evitar situaciones similares en el futuro. Este evento resalta la importancia de realizar pruebas exhaustivas, tanto durante el desarrollo de la herramienta como en la etapa de evaluación. Es fundamental contar con mecanismos de retroalimentación y revisión continua para identificar y corregir posibles errores o limitaciones de la herramienta. De esta manera, se podrá seguir mejorando su precisión y confiabilidad, y garantizar una detección más precisa y libre de elementos no deseados. A continuación, se ilustra el caso mencionado:

Figura 134. P&ID gráfico DTI-30203A.

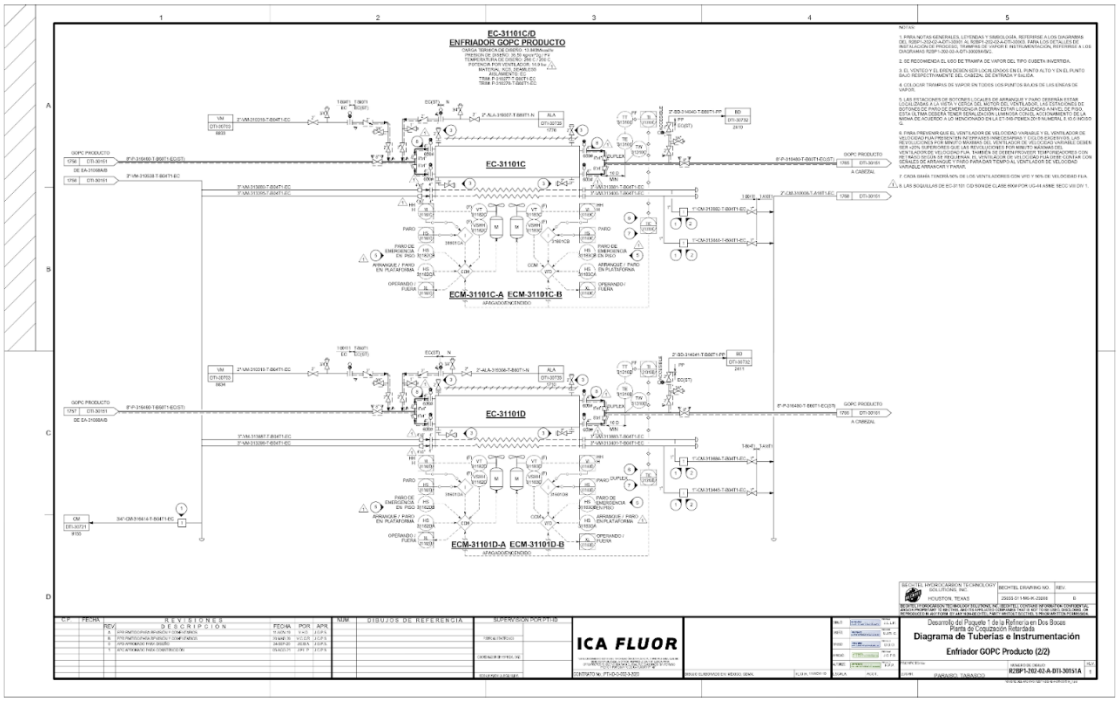


Figura 135. Plantilla HMI gráfico DTI-30203A.

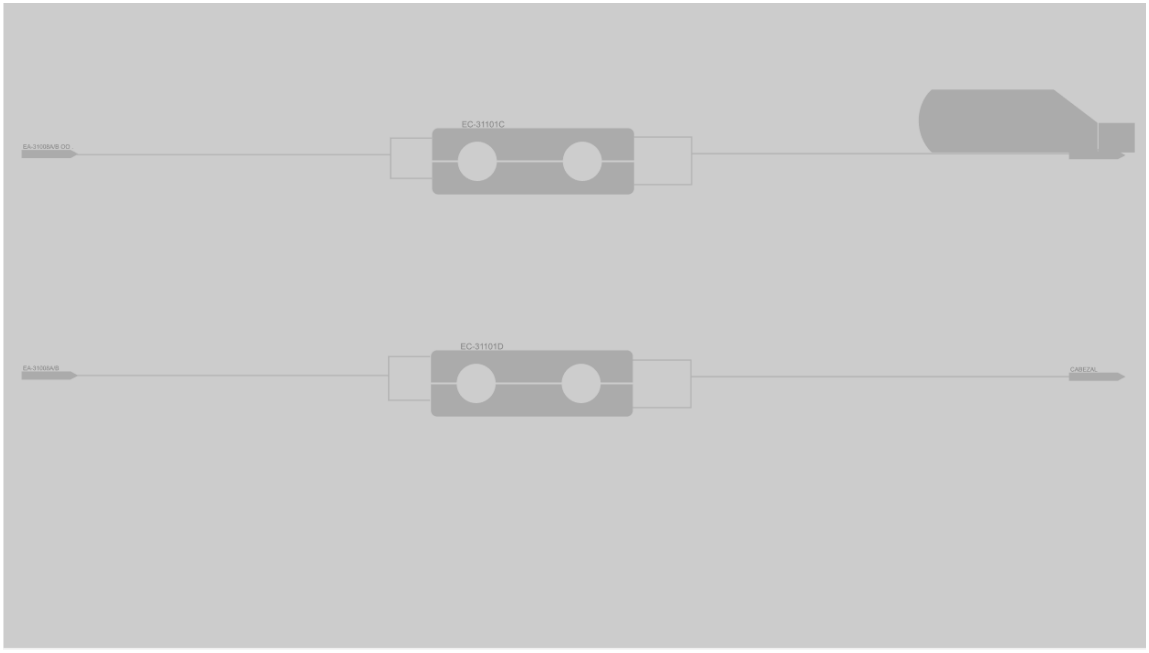
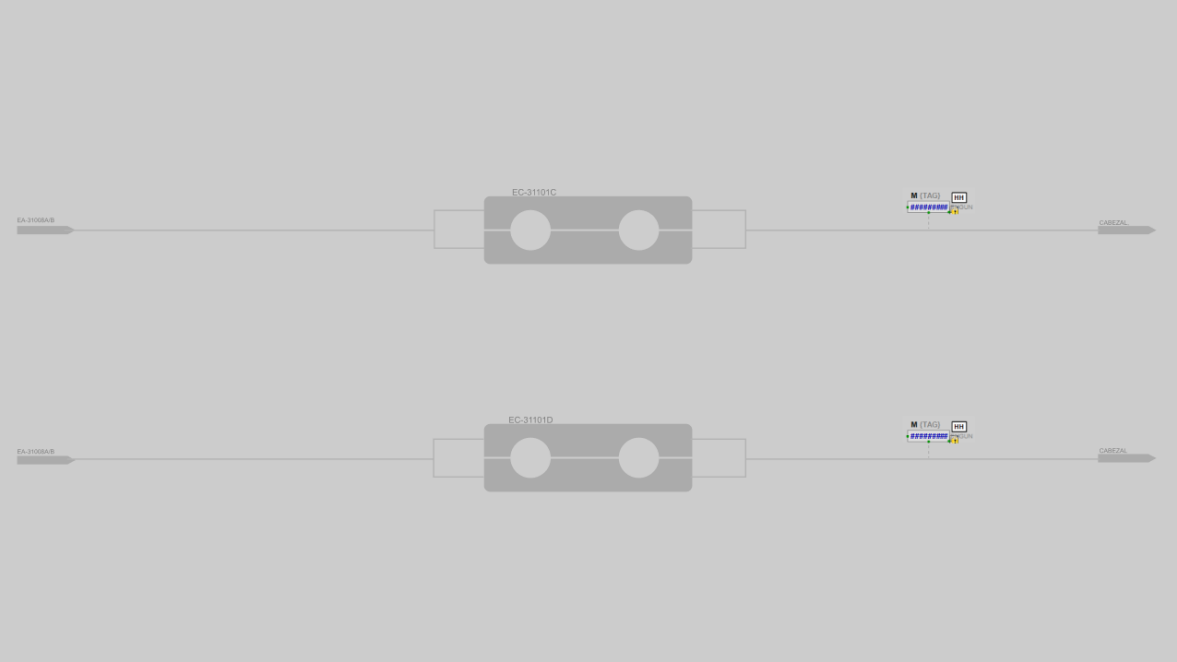


Figura 136. HMI gráfico DTI-30203A.



En el siguiente gráfico, el algoritmo de detección ha presentado un rendimiento deficiente. Solo se han detectado la mitad de los elementos esperados y solo se han reconocido dos de los seis textos presentes en la plantilla. Esto resulta en una baja confiabilidad, con un porcentaje por debajo del 50%. Según la escala establecida se clasifica como "Mal".

Este caso ejemplifica una situación en la que la herramienta no cumple con las expectativas y presenta limitaciones en la detección de elementos y textos en el gráfico. Es importante tener en cuenta estos escenarios para poder identificar las áreas de mejora necesarias en el algoritmo y trabajar en su perfeccionamiento. La detección deficiente de elementos y textos puede deberse a diferentes factores, como la complejidad de la plantilla, la calidad de la imagen o limitaciones en los algoritmos de detección utilizados. Es fundamental analizar estos casos de fallo para comprender las posibles causas y tomar medidas correctivas que permitan mejorar el rendimiento y la confiabilidad de la herramienta en situaciones similares.

Figura 137. P&ID gráfico DTI-30210.

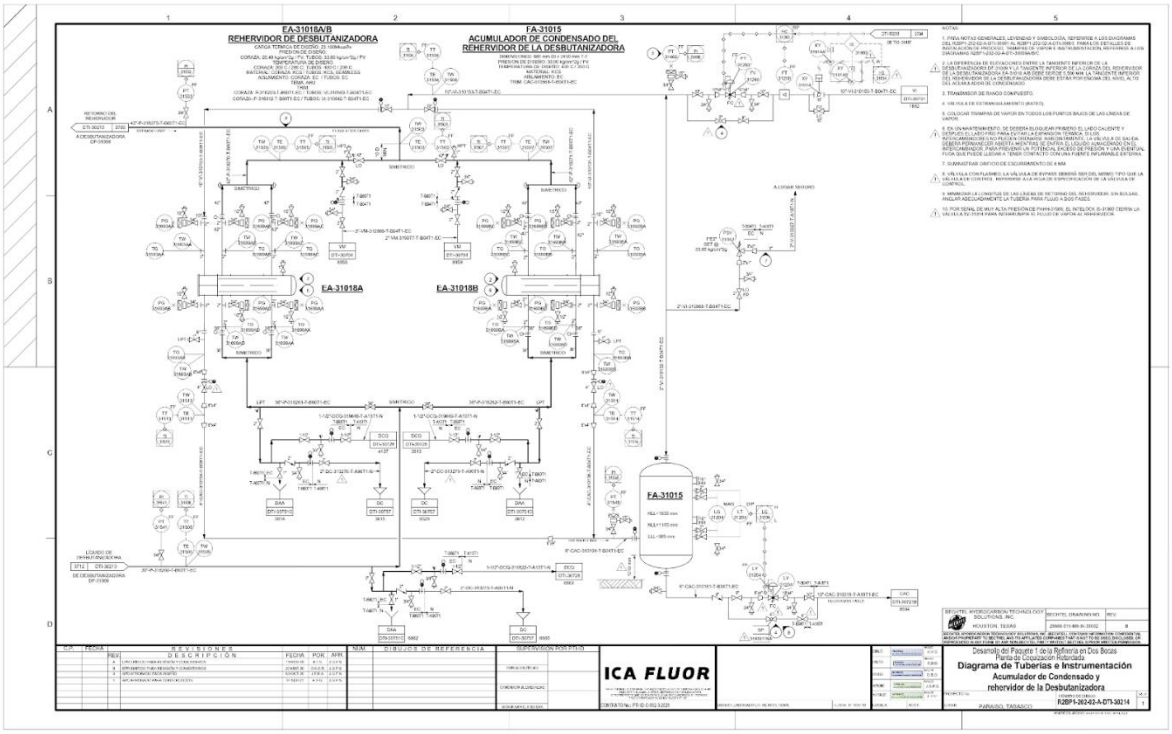
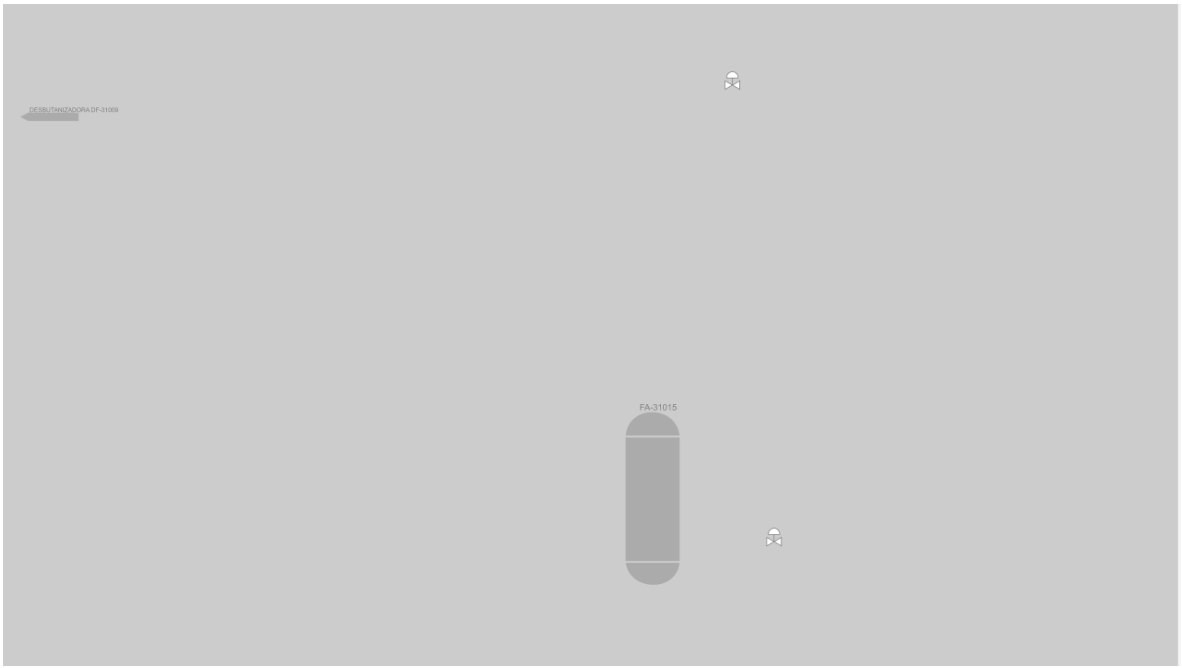


Figura 138. Plantilla HMI gráfico DTI-30210.



En este caso en particular, la información proporcionada era tan limitada que no se pudo completar el dibujo de la línea, lo que indica que es un gráfico que requiere ser realizado sin la ayuda de la herramienta. No obstante, es importante resaltar que la herramienta ha tenido mayormente casos exitosos. A continuación, se presentan algunos ejemplos donde ha demostrado su capacidad destacada. Aunque las plantillas no son perfectas, cumplen su propósito de agilizar el proceso y ahorrar tiempo.

En promedio, utilizar la herramienta para el diseño ha resultado ser un 28.62% más rápido en comparación con realizarlo sin ella. Este dato se ha calculado al promediar la mejora temporal obtenida en todos los gráficos evaluados durante la prueba.

Figura 139. P&ID gráfico DTI-30143.

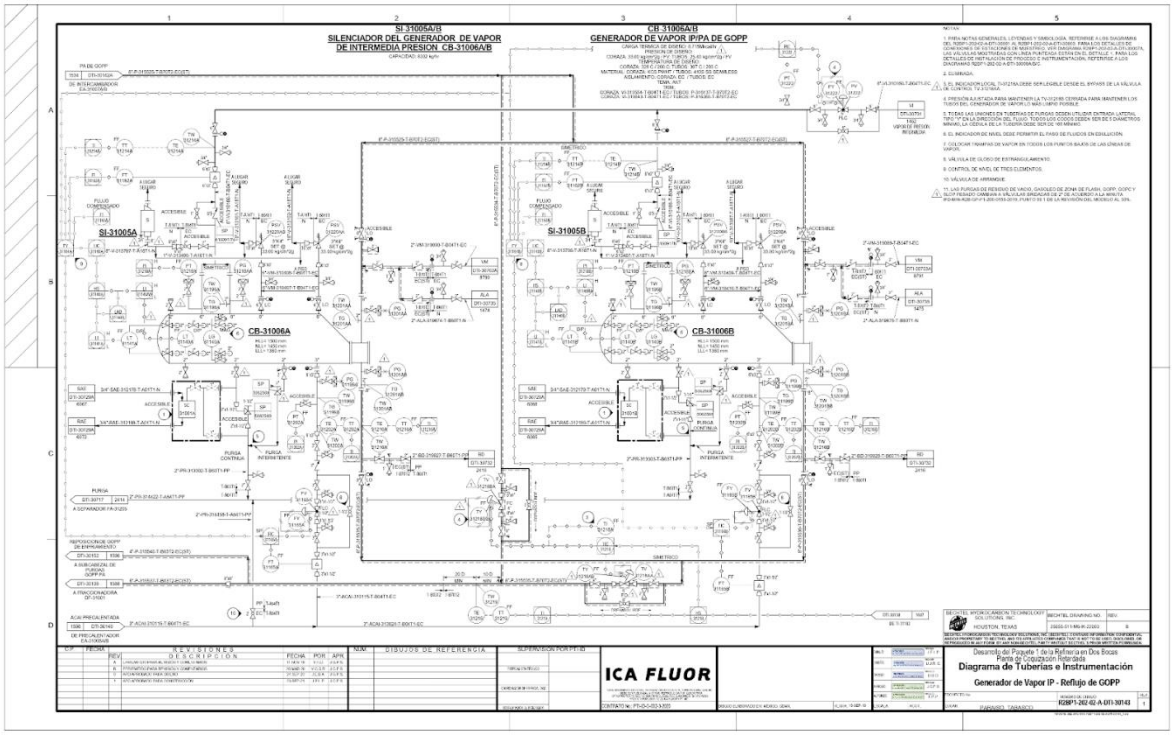


Figura 140. Plantilla HMI gráfico DTI-30143.

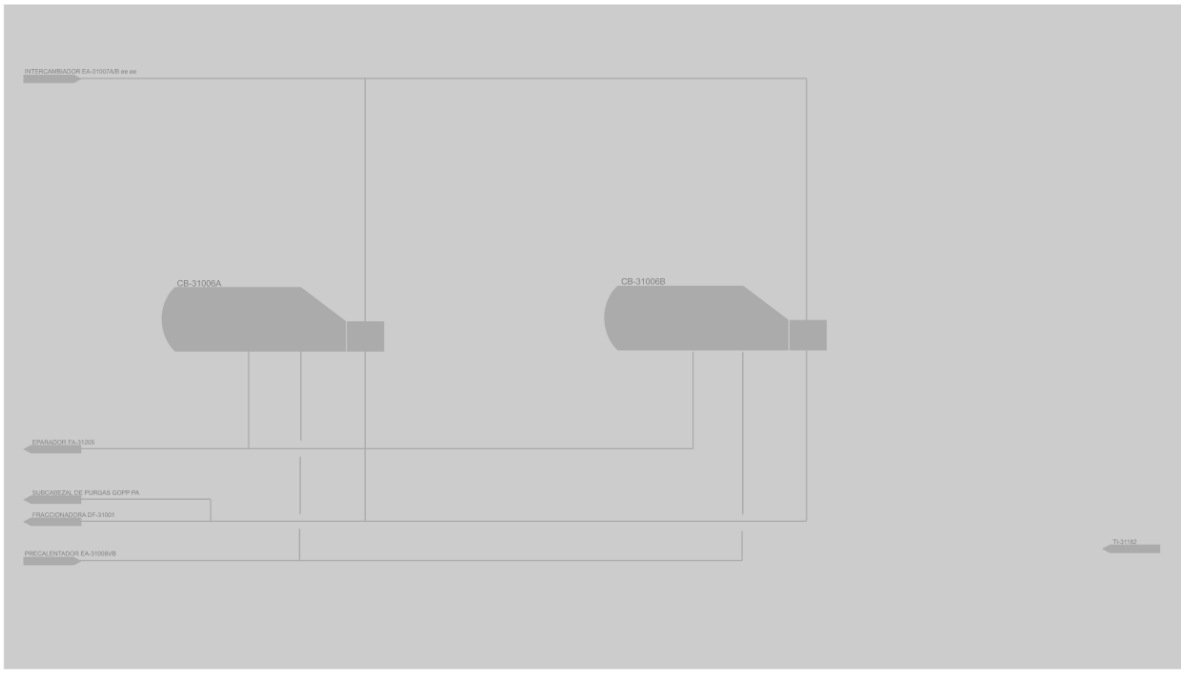


Figura 141. HMI gráfico DTI-30143.

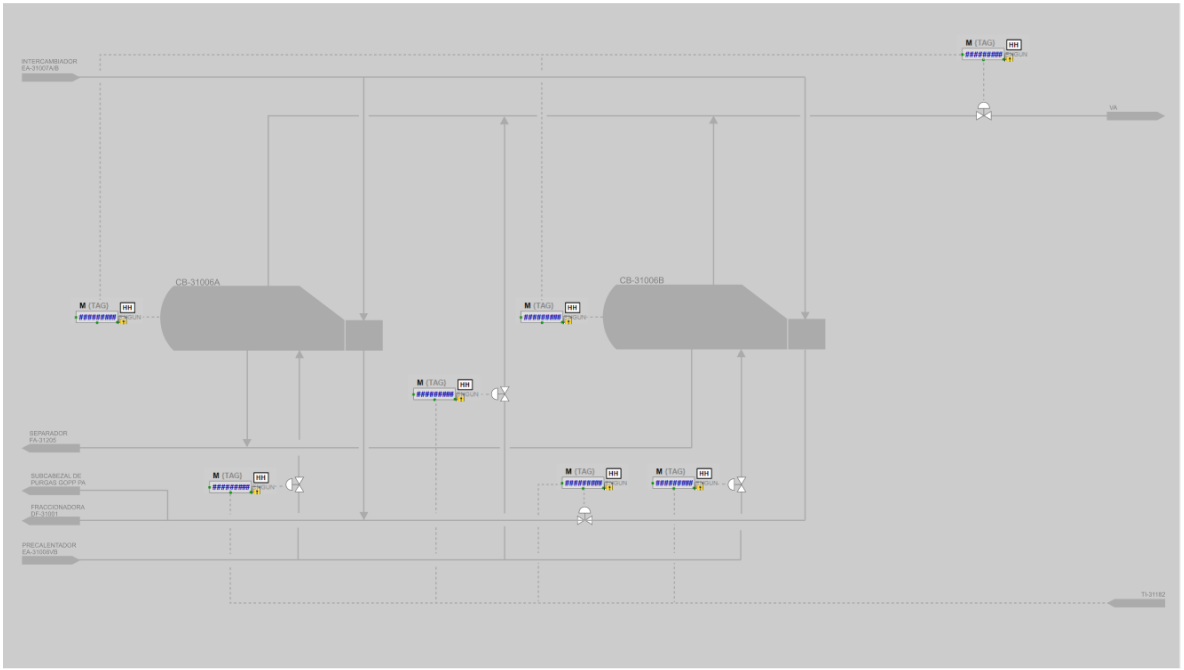


Figura 142. P&ID gráfico DTI-30162.

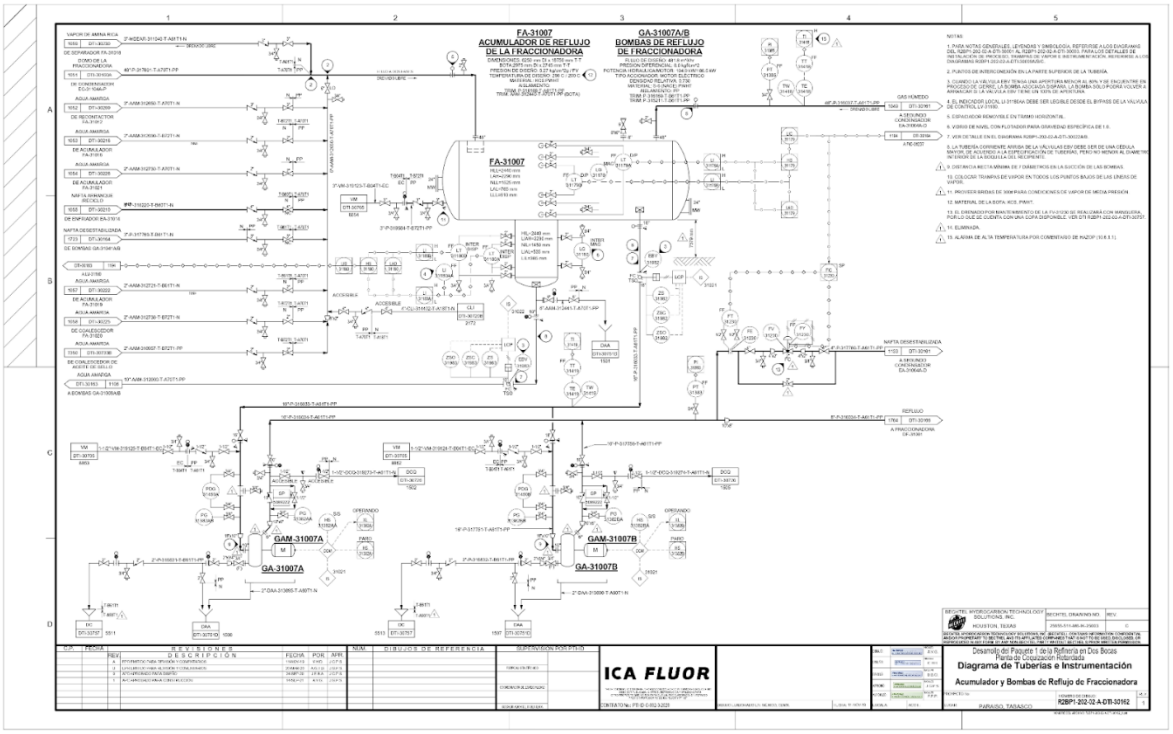


Figura 143. Plantilla HMI gráfico DTI-30162.

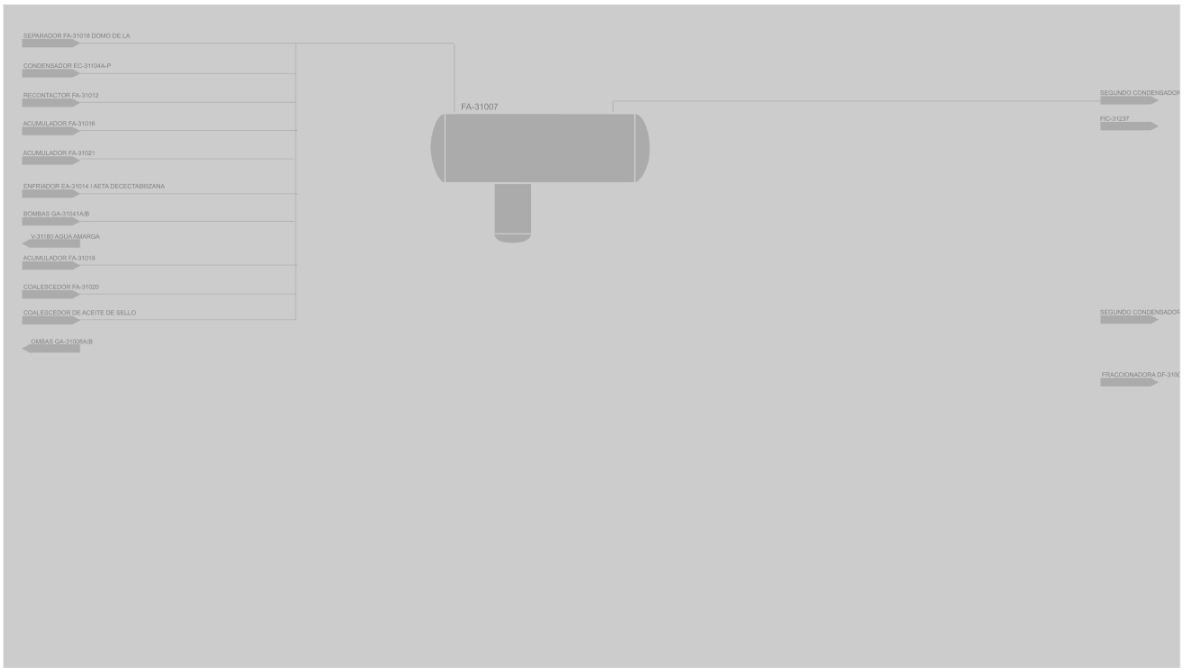
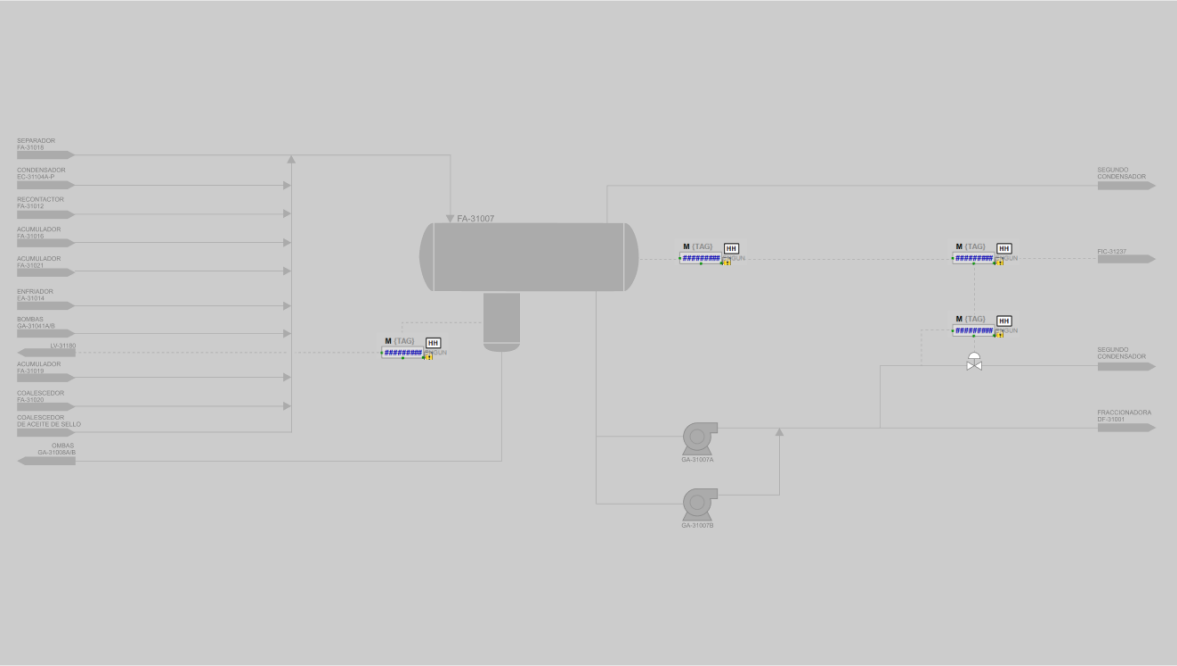


Figura 144. HMI gráfico DTI-30162.



Como se mencionó anteriormente, la herramienta muestra un desempeño destacado en P&ID que no están sobrecargados de información. Con modificaciones simples, se logra alcanzar el objetivo de una pantalla HMI, lo que representa un buen punto de partida para la primera versión de la aplicación. Este desarrollo continuará avanzando e incluso se espera contar con un equipo de trabajo para su evolución y mejora continua.

Capítulo 4

4. Conclusiones

Durante el desarrollo de la primera versión de esta herramienta, se ha observado el impacto positivo que la inteligencia artificial puede tener en la productividad de la industria. Es innegable que estas nuevas tecnologías han llegado para quedarse y es crucial incorporarlas en nuestras labores, no con el propósito de reemplazar o eliminar empleos, sino para mitigar tareas tediosas, de manera que se logren aumentar significativamente la eficiencia y productividad en cualquier actividad, como el desarrollo y diseño de HMIs.

La creación de esta inteligencia artificial ha pasado por diversas etapas, desde la idea inicial de detectar todos los elementos al mismo tiempo, hasta el enfoque de identificar únicamente el texto y luego dibujar en base a él. Después de un largo proceso de experimentación, se ha llegado a la conclusión que, debido a la complejidad de los diagramas y las limitaciones computacionales, es necesario generar modelos especializados para cada elemento que se desea identificar.

Es importante destacar que esta solución puede no ser la más depurada, pero se adapta de manera óptima a la situación. En las pruebas de confiabilidad realizadas, se han obtenido resultados interesantes. Se evidenciaron mejoras significativas en el tiempo de creación de los gráficos. Según las estadísticas recopiladas, los gráficos tienen un tiempo promedio de realización de 20 minutos, mientras que, utilizando la herramienta, el tiempo promedio de ejecución se reduce a cerca de 15 minutos. Esto representa una reducción de tiempo del 30%.

El porcentaje de comprensión de las plantillas generadas es en promedio del 85%, lo que indica que los gráficos son fácilmente asimilables. En cuanto a la exactitud, se ha obtenido un excelente promedio del 90%. Es importante destacar que puede haber casos en los que la exactitud no alcance dicho porcentaje, como se ha mostrado anteriormente. En general, todos

estos indicadores nos generan una sensación de confiabilidad para la herramienta del 80%, lo cual es clasificado como "Bueno". Sin embargo, es importante destacar que aún hay margen de mejora y se requiere recopilar cientos y cientos de datos de entrenamiento para considerarla una herramienta altamente confiable e integrarla de forma nativa en el proceso de diseño.

Esta primera versión ha arrojado excelentes resultados y experiencias que serán de gran utilidad para la segunda versión, donde se buscará elevar significativamente la confiabilidad y alcanzar la máxima calidad.

Por último, vale la pena dejar esta reflexión: la inteligencia artificial está aquí para quedarse y depende de nosotros, como ingenieros, implementarla para mejorar nuestros procesos, o corremos el riesgo de ser reemplazados por aquellos que se han adaptado o incluso, en el futuro, por esta misma tecnología.

5. Bibliografía

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. LeCun, Y., Bengio, Y., & Hinton, G. (1998). Convolutional Networks for Images, Speech, and Time Series. The Handbook of Brain Theory and Neural Networks, 3361(10).
3. Ludwig, G. (2009). Piping and Instrumentation Diagram (P&ID). Encyclopedia of Chemical Technology.
4. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
5. Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Pearson Education.
6. Smith, M. (2007). Optical Character Recognition: An Illustrated Guide to the Frontier. Springer.
7. Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.
8. Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345-1359.
9. Navathe, S. B., & Shamkant, N. (2015). Fundamentos de sistemas de bases de datos. Pearson Educación.
10. Rouhiainen, L. (2018). Inteligencia artificial. Madrid: Alienta Editorial.

11. Rauch-Hindin, W. B. (1989). Aplicaciones de la inteligencia artificial en la actividad empresarial, la ciencia y la industria. Ediciones Díaz de Santos.

12. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

13. Carballo Sierra, J., & Romero Lara, D. (2011). Tutorial norma ISA S5.1 y diagramas P&ID.

14. ISA. (2021). What Is an HMI?. Recuperado de: https://blog.isa.org/what-is-an-hmi

15. ANSI/ISA. (2015). Human Machine Interfaces for Process Automation Systems (ANSI/ISA-101-2015). Research Triangle Park, NC: ISA.

16. Anexos

16.1. Anexo A: Hoja de cálculos de los indicadores de evaluación

El Anexo comprende un archivo generado en Microsoft Excel donde se han registrado todos los datos por parte de los diseñadores y además se llevan a cabo las operaciones matemáticas para calcular porcentaje de tiempo, exactitud y confiabilidad de la herramienta.

El Anexo está disponible en formato digital dentro del paquete de entregables del proyecto de grado “HERRAMIENTA PARA EL DESARROLLO DE ELEMENTOS ESTÁTICOS EN HIGH PERFORMANCE HMI PARA LA EMPRESA INFODESIGN COLOMBIA”.

Nombre Gráfico	Diseñador	Nivel	Tiempo Herramienta	Tiempo Manual	% Tiempo	Compresión	Objetos por detectar	Objetos dectectados	% objetos detectados	Objetos indeseados	Textos por detectar	Textos detectados	% porcentaje textos detectados	Exactitud	Confiabilidad
DTI-30146A	1	1	7.00	10.00	30.00%	80.00%	9.00	8.00	88.89%	0.00	7.00	7.00	100.00%	93.33%	81.83%
DTI-30151	1	2	20.00	35.00	42.86%	90.00%	21.00	18.00	85.71%	0.00	18.00	16.00	88.89%	86.98%	80.82%
DTI-30146A	2	1	8.00	12.00	33.33%	70.00%	9.00	8.00	88.89%	0.00	7.00	7.00	100.00%	93.33%	80.83%
DTI-30107	3	2	16.00	19.00	15.79%	70.00%	15.00	15.00	100.00%	0.00	15.00	15.00	100.00%	100.00%	82.87%
DTI-30139	3	2	10.00	14.00	28.57%	90.00%	14.00	13.00	92.86%	0.00	13.00	11.00	84.62%	89.56%	80.48%
DTI-30151	3	2	18.00	33.00	45.45%	90.00%	21.00	18.00	85.71%	0.00	18.00	16.00	88.89%	86.98%	81.21%
DTI-30154	3	2	15.00	25.00	40.00%	70.00%	9.00	8.00	88.89%	0.00	6.00	6.00	100.00%	93.33%	81.83%
DTI-30203A	5	1	12.00	15.00	20.00%	80.00%	6.00	6.00	100.00%	1.00	6.00	6.00	100.00%	95.00%	81.50%
DTI-30139	6	2	11.00	15.00	26.67%	100.00%	14.00	13.00	92.86%	0.00	13.00	11.00	84.62%	89.56%	81.69%
DTI-30204	6	2	17.00	25.00	32.00%	80.00%	10.00	9.00	90.00%	0.00	9.00	9.00	100.00%	94.00%	82.60%
DTI-30139	7	2	11.00	17.00	35.29%	90.00%	14.00	13.00	92.86%	0.00	13.00	11.00	84.62%	89.56%	81.49%
DTI-30146A	7	1	5.00	8.00	37.50%	70.00%	9.00	8.00	88.89%	0.00	7.00	7.00	100.00%	93.33%	81.46%
DTI-30151	7	2	19.00	34.00	44.12%	100.00%	21.00	18.00	85.71%	0.00	18.00	16.00	88.89%	86.98%	82.51%
DTI-30139	8	2	10.00	14.00	28.57%	90.00%	14.00	13.00	92.86%	0.00	13.00	11.00	84.62%	89.56%	80.48%
DTI-30151	8	2	20.00	35.00	42.86%	90.00%	21.00	18.00	85.71%	0.00	18.00	16.00	88.89%	86.98%	80.82%
DTI-30154	8	2	14.00	20.00	30.00%	80.00%	9.00	8.00	88.89%	0.00	6.00	6.00	100.00%	93.33%	81.83%
DTI-30204	8	2	16.00	23.00	30.43%	80.00%	10.00	9.00	90.00%	0.00	9.00	9.00	100.00%	94.00%	82.37%
			14.71	**20.62**	**28.62%**	**85.82%**	**9.99**	**9.44**	**94.85%**	**0.10**	**8.99**	**8.48**	**93.68%**	**93.89%**	**82.84%**