

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA Y MECATRÓNICA

COMPUTACIÓN GRÁFICA

2022-2

Informe Proyecto de curso

Autores:

Estudiantes de curso
2022-2

Docente:

Miguel Ángel
Baquero Cortes

Índice

1. Bases de datos	1
1.1. Introducción	1
1.2. Descripción	2
1.3. Montaje de la base de datos	3
1.4. Metodología	3
1.5. Código	4
1.6. Empalme con la pagina web	4
1.7. Conclusiones	4
2. Página web	6
2.1. Descripción	6
2.2. Metodología	7
2.3. Conclusiones y recomendaciones	9
2.4. Contribuciones	10
3. Modelo CAD	11
3.1. Modelado Tanques	11
3.1.1. Parametrización y automatización	18
3.1.2. Conclusiones y recomendaciones	35
3.1.3. Contribuciones	35
3.2. Modelado Tableros	37
3.2.1. Descripción	37
3.2.2. Metodología	37
3.2.3. Referencias	40
3.2.4. Código	41
3.2.5. Planos de piezas	46
3.2.6. Conclusiones y recomendaciones	52
3.2.7. Contribuciones	52
4. Respuesta Automática	53
4.1. Explicación general	53
4.2. Código	55
4.3. Conclusiones	67
4.4. Referencias	67

Resumen

En el siguiente trabajo se va a mostrar brevemente el desarrollo de una pagina web que tiene como finalidad hacer respuesta automática para las necesidades de dos proyectos de aplicación. Para ello se divide en diferentes etapas principales donde se requieren para la ejecución total. En etapas encontramos la creación de pagina web, la bases de datos donde se almacenaran los registros, el modelo CAD parametrizado de los proyectos de aplicación para éste caso y la respuesta automática encargada en sincronizar toda esta información para poder enviar al cliente final una cotización.

1. Bases de datos

Integrantes:

- William David Bolivar Verano
- Carlos Andres Cardenas Ballares
- Diego Alejandro Beltran Rondon
- Luis Daniel Sánchez Molina

1.1. Introducción

Los datos en la actualidad son herramientas importantes para tomar decisiones en cualquiera que sea su campo de aplicación. Estos datos almacenan información de características de objetos reales de los cuales sirven para identificar al mismo y ser reconocido en alguna posible búsqueda. Para su manejo es importante clasificar y organizar adecuadamente ya que cuando se requiera realizar consultas o manejar el flujo de información se pueda hacer de manera más sencilla y rápida. Para esto se tiene como herramienta importante las bases de datos donde se almacenan y manipulan los datos. En medida que el flujo de información va siendo cada vez mayor se empezó a requerir métodos de almacenamiento cada vez mejores y para ello se tienen las bases de datos. De manera general una base de datos es una herramienta para almacenar información de forma eficiente y versátil. El fin de las bases de datos es almacenar datos de forma eficiente sin repetir información, teniendo la capacidad de buscar o seleccionar la información rápidamente. Estas bases de datos se han transformado en medida que la tecnología cada vez ha dado la posibilidad de que la capacidad de almacenamiento y velocidad se ha permitido.

En el manejo de la información principalmente se tiene dos tipos importantes de almacenamiento de acuerdo a las características de los datos se tienen bases de datos relacionales y no relacionales.

- Este manejo es su principal diferencia es que en las bases de datos relacionales se agrupan datos que tengan características en común para crear tablas que almacenan estos datos, por lo que se crean entonces tablas teniendo como fin no repetir la información. Luego se emplean marcadores, parámetros clave que relacionan tablas entre si, hay de dos tipos, primary key para referenciar la tabla actual y foreign keys para referenciar otras tablas con las que se interactúa, este tipo de bases de datos se caracterizan por usar SQL (Structured Query Language) o en español lenguaje estructurado de consultas, el cual es un estándar aceptado de manera internacional por la ISO desde 1987.
- Por otro lado encontramos las bases de datos no relacionales están en crecimiento constante desde aproximadamente el año 2000, sus principales ventajas son la flexibilidad, escalabilidad,

alto rendimiento y altamente funcional, la principal diferencia es que estas bases no cuentan con llaves foráneas ni primarias, la información no está estructurada, si por alguna razón requieres introducir un dato más que previamente no se esperaba lo introduces sin necesidad de realizar mayor cambio, mientras que para las relacionales si partíamos de una estructura fija, que es más compleja de modificar una vez ya está en uso, por otro lado el hecho de que este en auge tiene a más gente trabajando en su rendimiento y prestaciones actualmente está ganando espacio en las aplicaciones web, aun así para la implementación del proyecto se eligieron las bases relacionales para que no quedaran datos ni parámetros por fuera, no hubiera exceso ni ausencia de datos esenciales, requeríamos una estructura rígida para la cual las bases relacionales son mejor opción, no es sorpresa que actualmente sean las más usadas en Bancos e instituciones Gubernamentales.

1.2. Descripción

En esta sección se va a dar el desarrollo del trabajo en la parte de las bases de datos para el proyecto. Para esta parte se requiere tener claridad de la forma en como se manejarán las bases de datos para el uso en toda el proyecto, este se definió un flujo de información el cual se ve claramente en la figura 1. Para éste caso por versatilidad en nuestro proyecto la función de la base de datos será almacenar los registros realizados por el cliente en la página web. Por tanto la no se harán consultas por parte de la página web o las rutinas de visual basic a esta, simplemente se almacenara información a manera de registro.

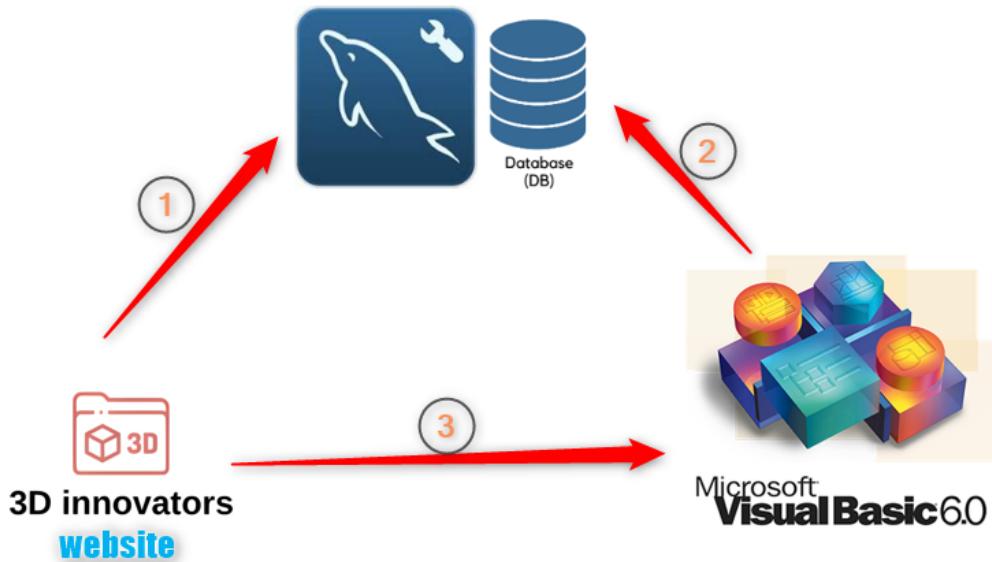


Figura 1: Flujo de información

Para nuestro proyecto se requirió seleccionar la base de datos más adecuada que cumplía con las funciones específicas que se necesitaban. Inicialmente se trabajó con una base de datos de libre acceso Maria DB, ya en el empalme de la página web se descartó debido a que los datos provenientes de los formularios como los datos del cliente y las características del objeto de cotización son enviados desde el sitio web hacia la base de datos y también a la rutina de visual basic de manera simultánea. Por tanto, decidimos en investigación previa observamos que se hacía más sencillo el manejo de la base de datos desde MySQL workbench para el montaje en wordpress en nuestro propósito.

Finalmente, la base de datos tiene como función almacenar los registros de los clientes. Ya en secuencia del funcionamiento los datos ingresados por el usuario son enviados en primer lugar a la base de datos para almacenar esta información y por otro lado a visual basic donde se recalculan los demás parámetros de construcción y manufactura. Seguidamente se actualiza el número de partes necesarias desde los modelos perimétricos en inventor programados con Ilogic para generar la lista de partes o archivo BOM el cual tiene las columnas precio unitario y cantidad las cuales permiten calcular un subtotal debido a las partes, este subtotal se pretende sumar con el valor de fabricación, IVA y demás para generar el valor total en una macro aparte y desde el VB como gráfica en la línea de cargar el valor de precio total en la base de datos.

1.3. Montaje de la base de datos

En esta sección se va a mostrar brevemente el proceso para el montaje de la página web en el word-press El hosting donde fue desarrollado el sitio web cuenta con plugins para el uso de bases de datos SQL, aun así previamente se debe diseñar lo que se conoce como el Diagrama Entidad- Relación, el cual es una herramienta gráfica para distinguir rápidamente los diferentes datos en las tablas respectivas y la relaciones que existen entre estas, primero partimos de que datos se deseaban almacenar, restringiendo a los mínimos necesarios los cuales fueron agrupados en las 4 tablas de la figura 2.

- Cliente (Nombre, Documento, Dirección, Email)
- Pedido (Cantidad, Costo)
- Tableros (Monofásicas, bifásicas, trifásicas y numero de bandejas)
- Tanques (Volumen y Empresa)

Para el montaje de estas bases de datos existen diferentes sistemas de gestión como MS Access, SQL Server, MySQL. Para cada uno de estos podemos encontrar diferentes plataformas y software que trabajan como herramientas visuales para diseñar e interactuar con nuestras bases de datos. Algunos gestores que encontramos son Maria DB y MySQL Workbench entre otros. Para nuestro caso, por compatibilidad y facilidad se trabajo con MySQL workbench.

1.4. Metodología

Para el diseño e implementación del diagrama E-R se empleo SQL sever trabajando sobre MySQL Workbench sobre el cual se propuso la siguiente implementación:

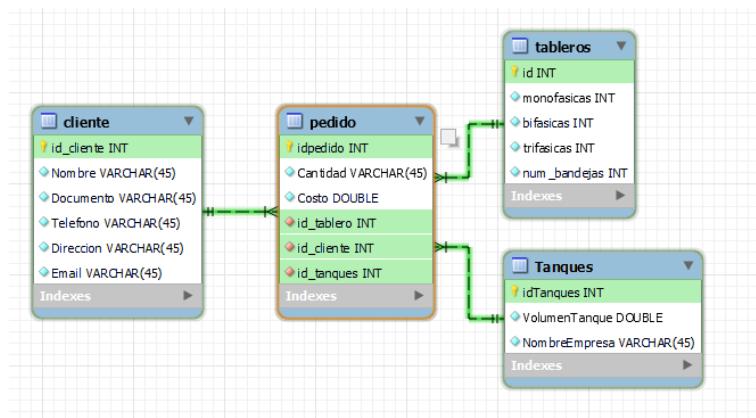


Figura 2: Diagrama entidad relación

1.5. Código

Una vez se tiene la estructura de la base, es necesario poder exportarlo, por lo que en el siguiente enlace encontramos un vídeo corto y conciso sobre como exportar el código fuente de la implementación planteada:

<https://www.youtube.com/watch?v=79rL-JCqmkU>

1.6. Empalme con la pagina web

Cuando se montó la base de datos en el hosting, el almacenamiento de los datos recopilados por los formularios de la página web estaba generando un error. Esto debido a que toda la información se ejecutaba en una sola tabla, lo cual resulta poco funcional y no cumplía con los objetivos propuestos al inicio de almacenar los datos ordenadamente en tablas distintas por medio de una relación.

Para la solución de este problema, fue necesario la implementación de Triggers dentro de la misma base de datos. Estos Triggers son propios del lenguaje de programación MySQL y son más conocidos como disparadores, los cuales están atentos a la inserción de cualquier registro en la tabla general, y cuando esto sucede, si lo que ingresa es un nombre entonces toda esa información se almacena dentro de la tabla clientes, y así sucesivamente, a continuación se muestra un fragmento del código de un Trigger el cual se encarga de almacenar los datos del usuario registrado dentro de la tabla clientes:

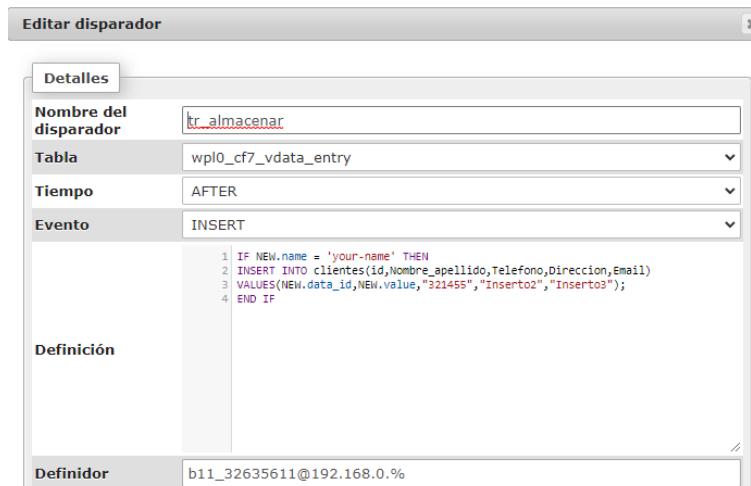


Figura 3: Trigger almacenamiento datos del cliente

1.7. Conclusiones

- La selección de la base de datos adecuada en donde el flujo de trabajo sea más fácil debe ir de la mano con la compatibilidad con la página web para un manejo de la información adecuada a las necesidades.
- Existen gestores de bases de datos y la selección dependerá del propósito de la plataforma o herramienta web donde se ejecutarán. Para ellos es importante indagar sobre las tecnologías existentes con el fin de tener el mejor rendimiento.
- MySQL workbench puede usar una herramienta simple e intuitiva con una curva de aprendizaje alta para generar utilizar bases de datos relacionales y generar código automáticamente.
- El uso e implementación de bases de datos en un sistema o proceso, es de vital importancia para el adecuado funcionamiento del mismo, con este curso aprendí a implementar de manera

adecuada una base de datos para un proyecto, aprendí como conectar una base de datos con una pagina web en linea, y como los datos almacenados en esta, exportarlos mediante Visual Basic for Applications a Excel donde se hace uso de la información requerida.

- Rescato que también aprendí como hacer Triggers dentro de una base de datos, y me di cuenta que son un elemento de vital importancia para el correcto funcionamiento de la misma, el funcionamiento de un Trigger optimiza mucho el tiempo de ejecución de acciones dentro de una base de datos, y con ello el trabajo que se realiza es menor. (Luis Daniel Sánchez)

2. Página web

Integrantes:

- Hugo Duhazé
- Luis Daniel Sánchez Molina
- Alirio Hernando Martínez Barreto
- Sergio Eduardo Peña Santamaría

2.1. Descripción

Esta parte del proyecto se refiere a la creación de una página web para crear cotizaciones para tanques de gasolinas y tableros eléctricos. La pagina web cuenta con apartados adecuados y claros para la comodidad del usuario en su cotización del producto deseado, el estilo simple resalta cada detalle a tener en cuenta en el proceso, las imágenes implementadas son guías para tener un buen punto de referencia inicial, mientras que los formularios presentan los datos específicos a tomar para el correcto modelado de las piezas requeridas.

Elige el producto



Tanques



Tableros

Figura 4: Elige su producto

Cuando ingresamos a la pagina web, podemos elegir el tipo de producto que nos interesa entre tableros eléctricos y tanques de gasolina. Se ha optado por un diseño simple y agradable a la vista, donde el usuario puede entender directamente el propósito de la página.

Modelado Tanques de gasolina



Volumen del tanque (Galones)

Material del tanque

Nombres, Apellidos

Nombre Empresa

Correo electrónico

Dirección

Número de contacto

Figura 5: Formulario Tanques

Una vez seleccionada alguna de las opciones se redirecciona el al formulario correspondiente, cada uno es diferente del otro y solicita los datos necesarios, con los campos llenos al presionar enviar la pagina empieza su conexión con la base de datos mediante correos que llevan los datos de un punto a otro.

2.2. Metodología

Creación del sitio web:

El sitio web se creó con Wordpress. Es un sistema de gestión de contenidos (CMS) gratuito y de código abierto. Este software escrito en PHP se basa en una base de datos MySQL y es distribuido por la fundación WordPress.org. Las funcionalidades de WordPress permiten crear y gestionar diferentes tipos de sitios web: sitio de escaparate, sitio de venta online, sitio de aplicaciones, blog, portafolio, sitio institucional, sitio de enseñanza...

Para simplificar la concepción de las diferentes páginas, utilizamos el plug-in Elementor, que nos permite que nos permite organizar y colocar diferentes elementos (formularios, imágenes, botones, etc...) fácilmente.

Para la creación de la página web primero realizamos la creación de un dominio y un Hosting donde se va a alojar el sitio web, para obtener un dominio gratis, realizamos el uso de un administrador de dominios gratis llamado FREENOM, el cual nos brinda las mismas funcionalidades de un dominio pago, sin embargo las extensiones de este no son muy conocidas.

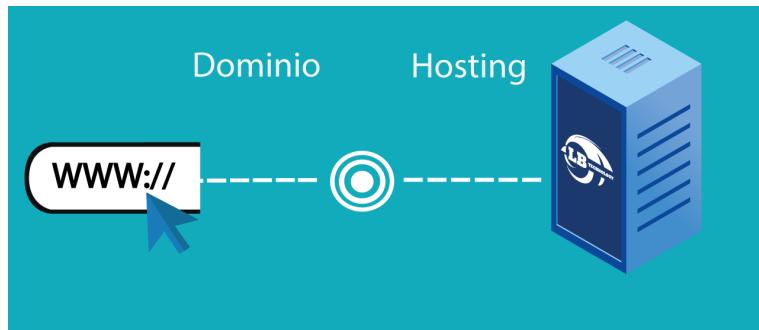


Figura 6: Ilustración Dominio y Hosting

Para el alojamiento del sitio web, usamos un servidor gratuito de hosting llamado byethost, en este podemos administrar por completo el sitio web, y agregar las demás funcionalidades y APIs de Google para el funcionamiento deseado en la página web. Una vez esta montado, realizamos la certificaciones del sitio web con un certificado SSL, el cual hace que aparezca el candadito de que el sitio es seguro, por ende proporciona fiabilidad en nuestros servicios.



Figura 7: Certificado SSL del sitio web

Para el diseño de la página web, se hizo uso de una herramienta muy poderosa llamada Figma, la cual nos permite hacer un bosquejo de lo que será ya la implementación final del diseño y funcionalidades de la página web. Cuando ya teníamos hecho el bosquejo se implementó directamente en el servidor mediante el uso de Workpress, el cual nos facilitó el procedimiento.

Para la generación de correos automáticos implementamos una API de Google, la cual nos permite, que cada vez que se ingresen datos a los formularios dispuestos en la página web, se envíe un correo automáticamente al correo del sitio web con una especificación del asunto, en este caso, los datos para cajas para tableros y tanques de gasolina. Una vez se tiene esto, podemos hacer la conexión directamente con el Excel que está parametrizado, el cual usarán los de modelado. Para esta conexión se hizo uso de un código en Visual Basic for Applications.

Creación de máquina virtual:

El objetivo de la creación de la máquina virtual es permitir que toda la información y datos del proyecto se mantengan en un solo dispositivo, además de ejecutar todos los códigos, funciones y peticiones que se requieren para la correcta ejecución del proyecto.

Su funcionamiento es posible gracias a un servidor remoto que realiza las acciones de cómputo que haría un computador común, sin embargo la diferencia radica en que el servidor permite una conexión a internet más estable, además permite realizar el procesamiento de datos y peticiones de una forma más rápida y eficiente, evitando de ese modo sobrecargar otros dispositivos y centralizando la ejecución del proyecto en un solo lugar que realiza todas las acciones necesarias para que el proyecto funcione.

La máquina virtual se desarrolló a partir de Microsoft Azure, una plataforma creada y diseñada específicamente para realizar acciones de cómputo y procesamiento de datos en la nube, gracias a que tiene una potente red de servidores alrededor del mundo.

Especificaciones de la máquina virtual:

Imagen del sistema: Sistema operativo SO Windows server 2019 Datacenter-gen2

Memoria ram: 16 gb

Grupo de recursos:

4 unidades centrales de procesamiento virtual

Disco duro SSD 140 gb

Disco duro de almacenamiento temporal 150 gb

Ancho de banda de red máxima: 6400 Mbps.

Para concluir con el desarrollo de la máquina virtual fue necesario descargar los diversos programas necesarios para que el proyecto corriera de forma satisfactoria, entre ellos están:

Excel

Matlab

Inventor

Sistemas de seguridad:

Debido a que la máquina virtual funciona gracias a un servidor remoto desde el cual se accede usando internet, es importante restringir y controlar el acceso mediante el uso de reglas de puerto de entrada, que determinarán quién tiene acceso al servidor, con el fin de salvaguardar la seguridad de la información. En este caso específico se restringe el acceso total de cualquier dispositivo, por lo tanto, para acceder es necesario crear una regla de acceso usando protocolo RDP (Remote Desktop Protocol) usando por defecto el puerto 3389 y enlazando la conexión exclusivamente con la ip pública del dispositivo desde el que se va a realizar la conexión.

2.3. Conclusiones y recomendaciones

Wordpress es una herramienta muy útil para crear rápidamente un sitio web. Tiene muchos plugins que te permiten hacer casi todo lo que quieras, sin tener que dominar un lenguaje de programación web (html, CSS, Javascript).

Es necesario plantear un concepto o idea antes de iniciar el desarrollo de la página web, el diseño de la misma es fundamental para su correcto uso y aprovechamiento, por eso mismo, se concluye en hacer una página web de tan solo dos pasos sin más complicaciones. Si bien no existe un registro, la casilla de correo en el formulario ahorra este proceso, además de que evita sobrecargar la base de datos de correos, usuarios y contraseñas que podrían complicar el almacenamiento de los demás datos necesarios presentes en el formulario.

Recomendaciones: Elija cuidadosamente su dominio de alojamiento para evitar problemas de conexión, caídas constantes del mismo o perdida de avances significativos alojados en un dominio disfuncional.

Conclusiones: El desarrollo de una página web para un proyecto o empresa, es de vital importancia, es ese medio dónde podemos conectar fácilmente el usuario y el producto final que desea. Adicional, llevar el registro de los clientes en una base de datos, hace que tengamos información relevante que más tarde usemos a favor de la empresa y del usuario, considero que el curso de computación gráfica es un gran medio para ampliar nuestros conocimientos y esa curiosidad por aprender y optimizar procesos de la vida cotidiana. En general me llevo grandes ideas y curiosidad por explorar procesos de optimización y ponerlos en práctica en mi vida profesional y académica. (Luis Sánchez)

2.4. Contribuciones

- Hugo Duhazé : Diseño de páginas web y creación del logotipo.
- Luis Sánchez : Creación de las interfaces principales y los formularios para el ingreso de los datos solicitados, implementación y conexión de la API de Google para la generación de correos automáticos, y su posterior conexión con el Excel de modelado, adicional la conexión con la base de datos alojada en el hosting del sitio web y conexión con las tablas de la base de datos mediante la implementación de Triggers de MySQL.
- Alirio Martínez : Innovación creativa, aporte al diseño y funcionamiento de la página web, corrección de detalles y disposición al aprendizaje.
- Sergio Peña : Implementación y creación de la maquina virtual

3. Modelo CAD

3.1. Modelado Tanques

INTEGRANTES:

- Raul Alejandro Gil Zapata
- David Alejandro Niño Barrero
- Juan Sebastián Sánchez Martín
- David Aguirre Rocha

En esta sección se presenta el proceso para el modelado parametrizado de los tanques de almacenamiento, utilizando el software Autodesk Inventor en su versión 2023. Para comenzar, la empresa nos brindo un archivo .dwg de AutoCAD, donde se pueden observar tanto la geometría general del tanque de almacenamiento, como las medidas que serían paramétricas en el modelado posterior:

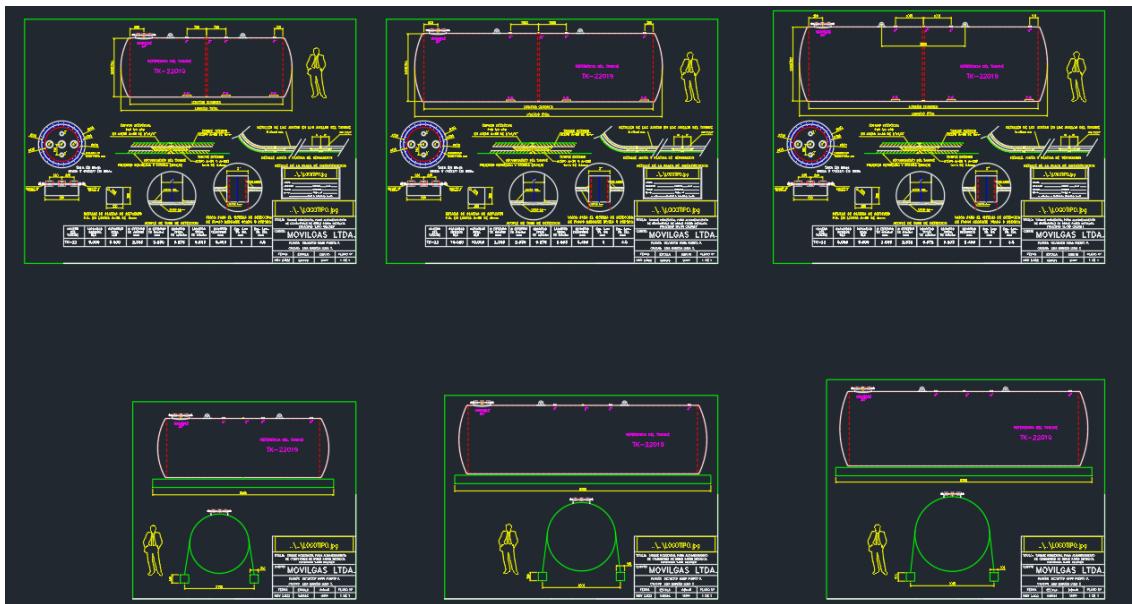


Figura 8: Plano base para el modelado de los tanques de forma paramétrica.

Si bien se tienen una cantidad de medidas considerables, se observó que las más importantes a parametrizar en el modelado son los siguientes 4:

- Diámetro exterior (Dext)
- Longitud del cilindro (Lcil)
- Distancia al centro Refuerzos (DCR)
- Distancia Elementos colgantes (DSC)

Con esto en cuenta se comienza con el modelado en Autodesk Inventor, donde se plantea el tanque como tal como una pieza única soldada, dividida por subsolidos, 3 piezas referentes a la tapa de dicho tanque, y piezas normalizadas para realizar las uniones pernadas, como tuercas, tornillos y arandelas. Respecto al modelado del tanque se comienza con la creación del cilindro central que define el cuerpo del mismo, donde se crea el parámetro de longitud en la extrusión del cilindro:

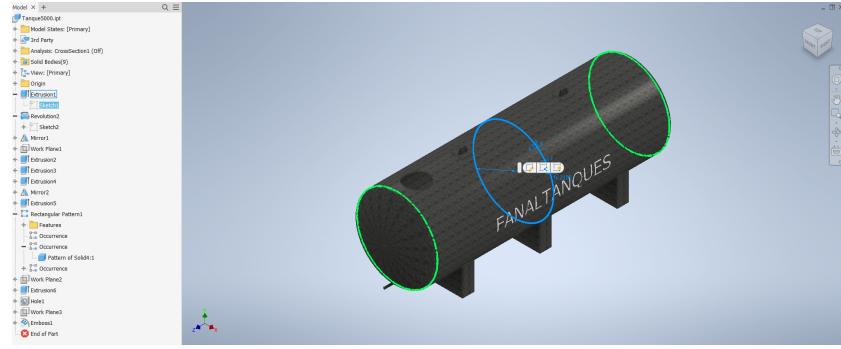


Figura 9: Extrusión del cilindro que forma el tanque.

A continuación se realiza una revolución para dar la curvatura al tanque y la formación de las zonas internas del mismo, incluyendo también la ubicación de puntos con la longitud paramétrica DCR, la distancia al centro de los refuerzos internos:

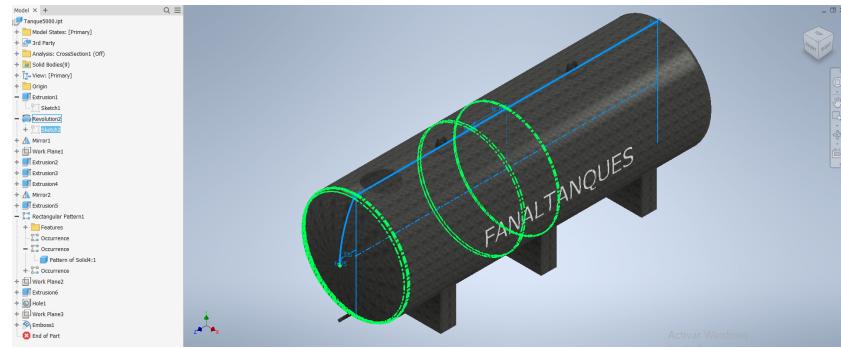


Figura 10: Operación de revolución en el tanque.

A esta operación se le aplica un "mirror" para que se replique en el lado derecho del tanque. Una vez realizado esto se realiza el agujero donde irá la tapa del tanque, el cuál no requiere como tal de medidas parametrizadas porque su posición se define respecto al borde como se observa:

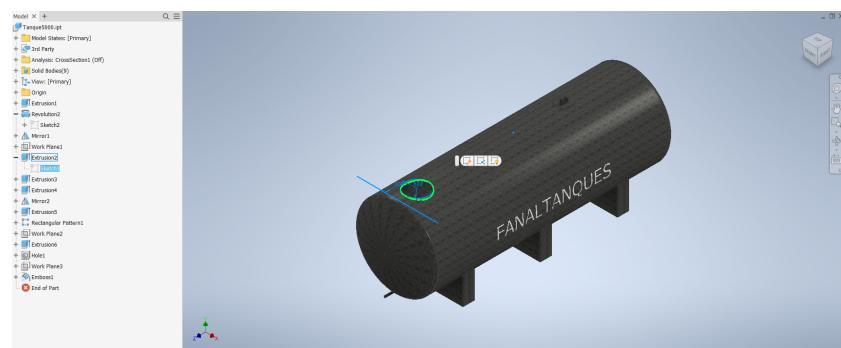


Figura 11: Operación de agujero donde se ubicará la tapa del tanque.

A continuación se realizan las extrusiones para poder hacer los componentes para poder tomar el tanque desde la parte superior como se muestra, utilizando otra medida paramétrica:

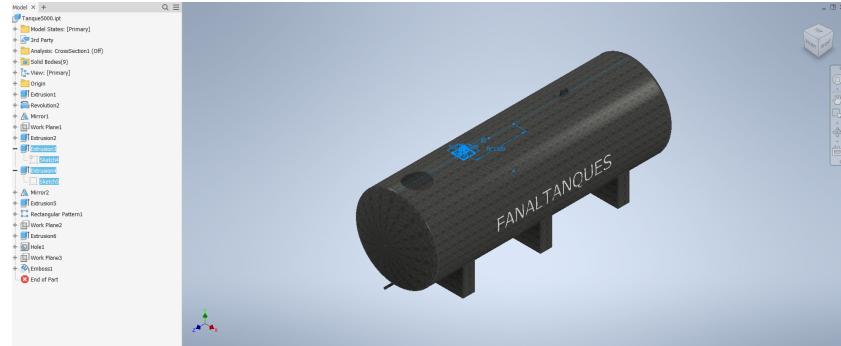


Figura 12: Operación de Extrusión salientes superiores del tanque

Se realiza una operación de mirror para tener esta saliente a ambos lados, y luego se definen tanto la extrusión como el arreglo en línea para cada uno de los apoyos inferiores del tanque como se muestra:

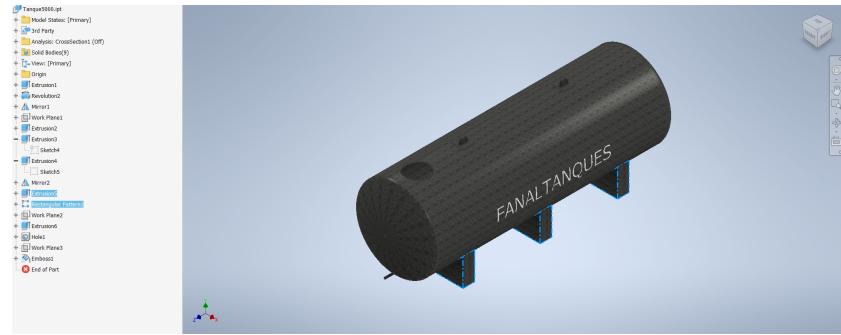


Figura 13: Operación de Extrusión y creación del patrón lineal de los apoyos inferiores del tanque

Y finalmente se realizan las operaciones correspondientes a la creación de la boquilla de salida, y el grabado del nombre de la empresa en el tanque, como se observa a continuación:

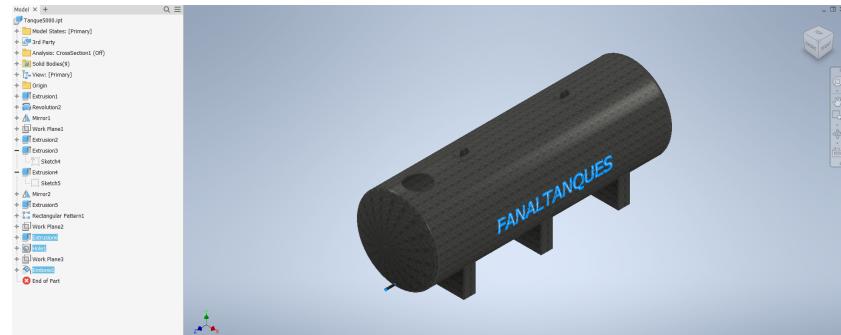


Figura 14: Operación de Extrusión y agujereado de la boquilla y grabado del nombre de la empresa en el tanque.

De esta forma se tiene el modelo del tanque final, el cuál en el entorno de ensamble se le agregan las piezas requeridas para la tapa y se utilizan piezas normalizadas para las uniones pernadas como se muestra en la siguiente figura:

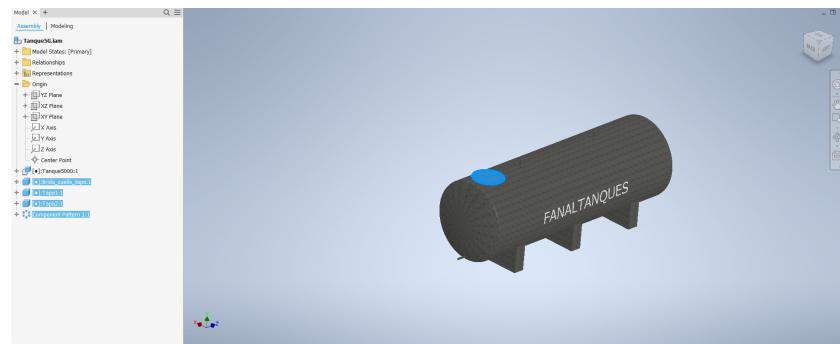


Figura 15: Ensamble general del Tanque.

Teniendo ya el ensamble se procede a la creación de los planos, para lo cuál se crea un único archivo .dwg y multiples hojas o sheets donde se ubicarán los planos de cada pieza y los de ensamble. A continuación se expone la interfaz:

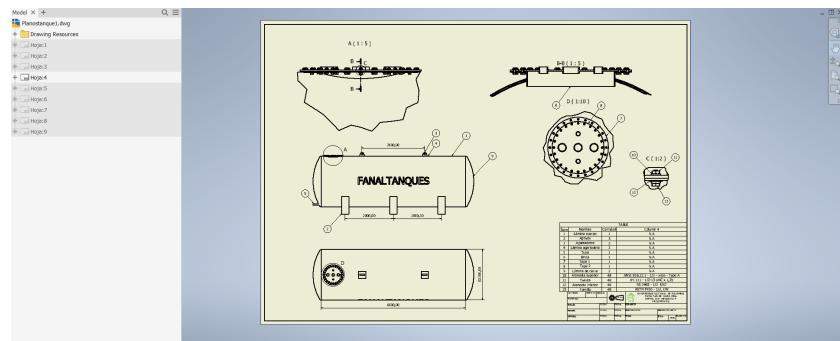


Figura 16: Interfaz Inventor Planos del Tanque.

A continuación se muestran los planos generados:

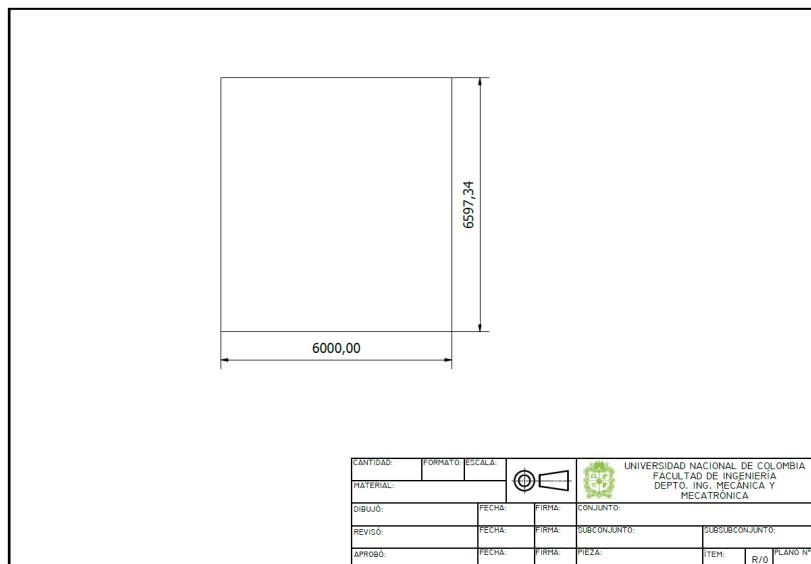


Figura 17: Plano 1 generado en pdf desde Inventor.

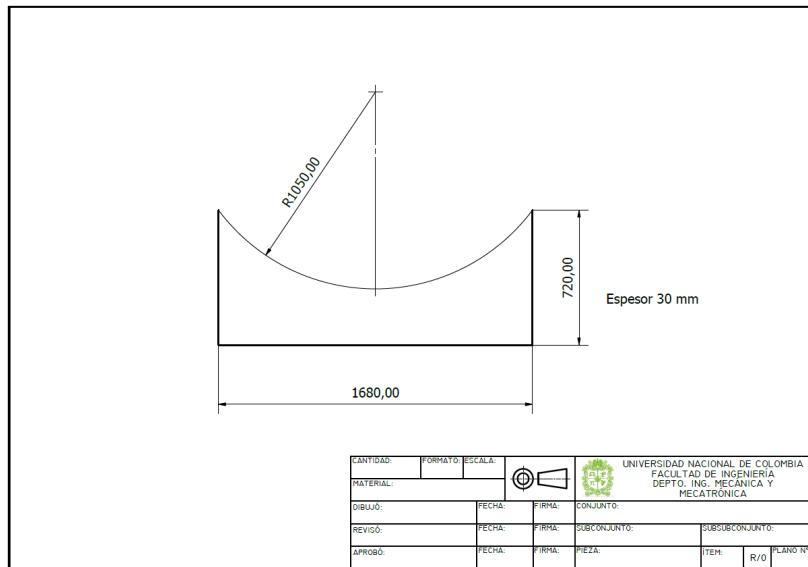


Figura 18: Plano 2 generado en pdf desde Inventor.

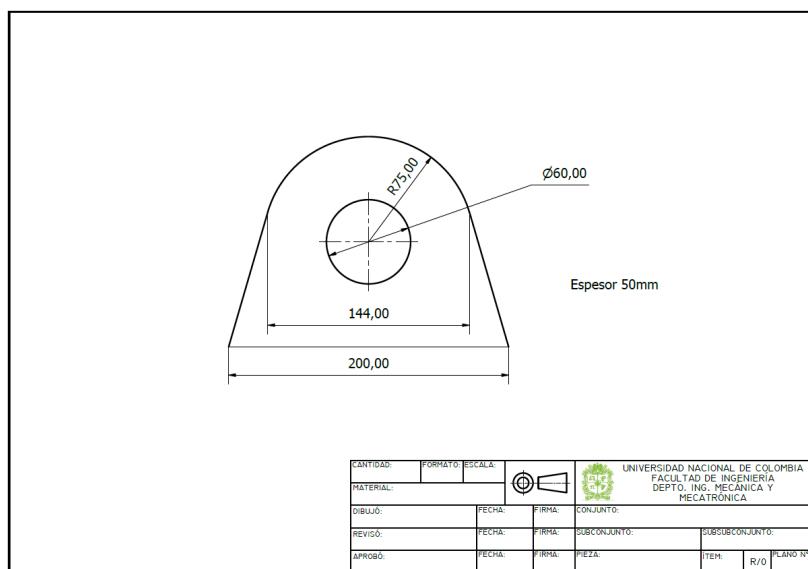


Figura 19: Plano 3 generado en pdf desde Inventor.

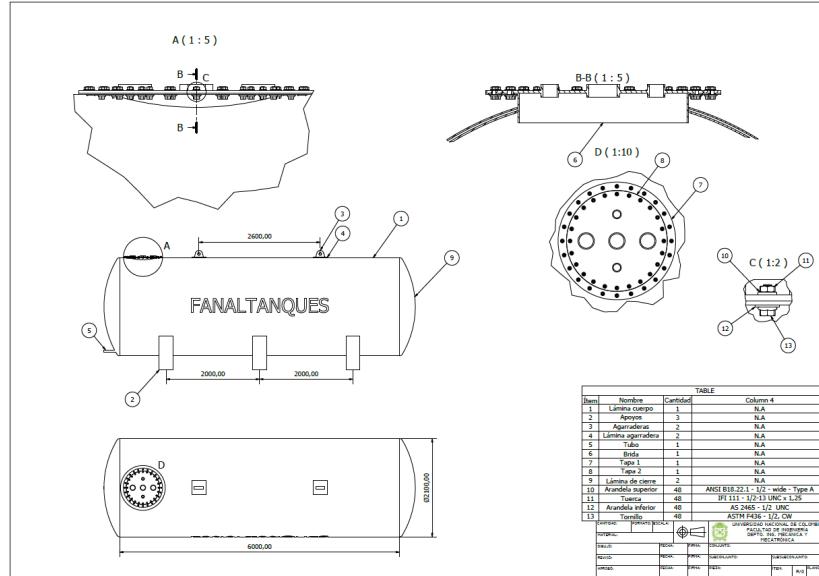


Figura 20: Plano 4 generado en pdf desde Inventor.

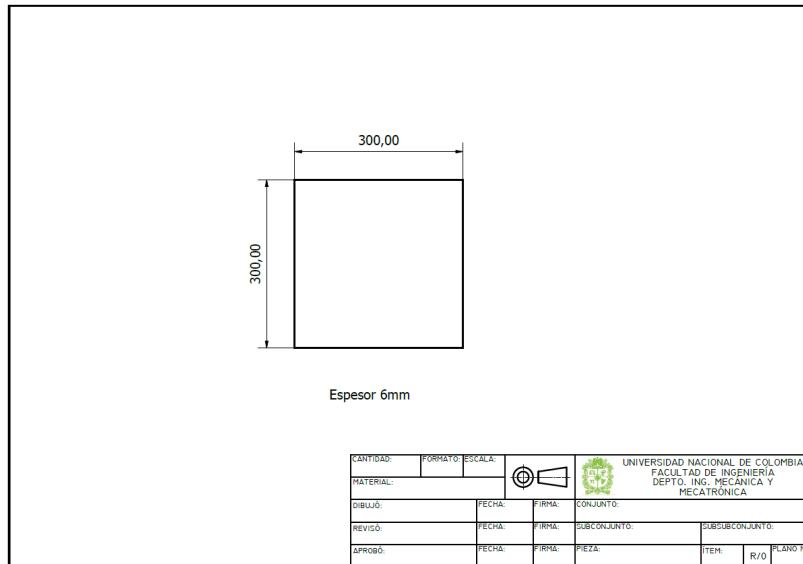


Figura 21: Plano 5 generado en pdf desde Inventor.

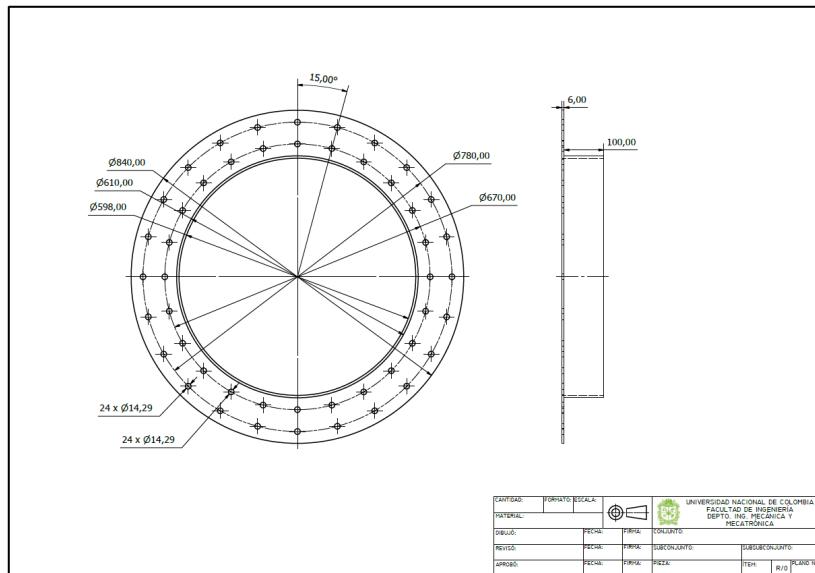


Figura 22: Plano 6 generado en pdf desde Inventor.

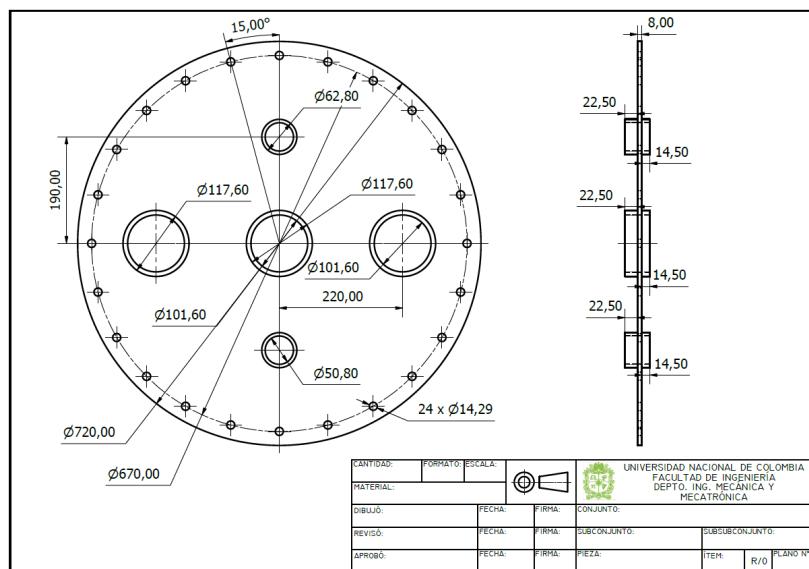


Figura 23: Plano 7 generado en pdf desde Inventor.

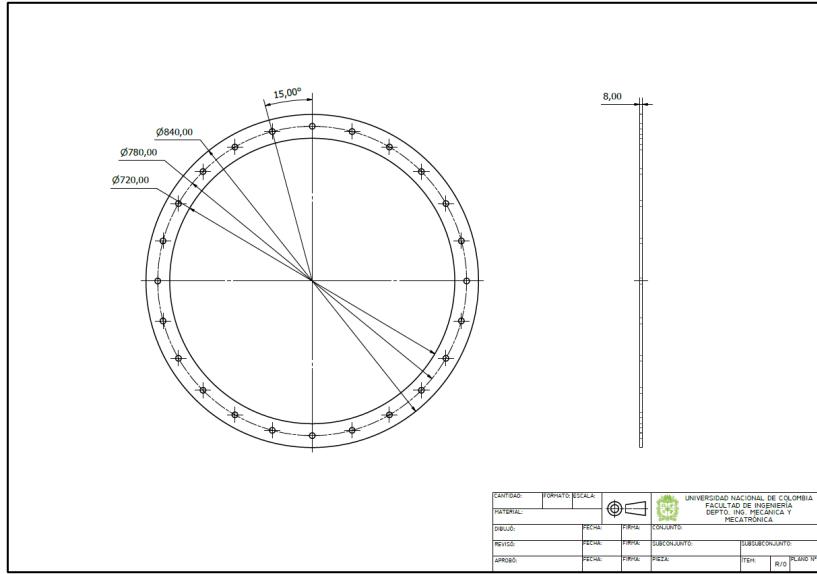


Figura 24: Plano 8 generado en pdf desde Inventor.

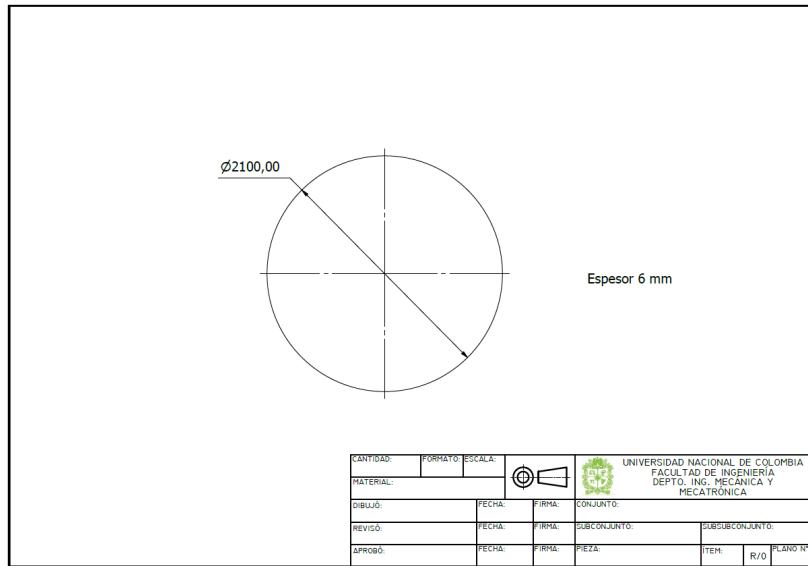


Figura 25: Plano 9 generado en pdf desde Inventor.

Teniendo todo esto el siguiente paso es la unión y automatización de todos estos procesos.

3.1.1. Parametrización y automatización

Dados los requerimientos del problema, se requiere entonces definir 4 medidas paramétricas que estén asociadas en este caso a un documento de Excel, ya que estas medidas serán las dadas por el usuario en la página web y serán llevadas a dicho documento. A continuación se observan los parámetros definidos como user parameters en inventor, y en la hoja de cálculo mencionada:

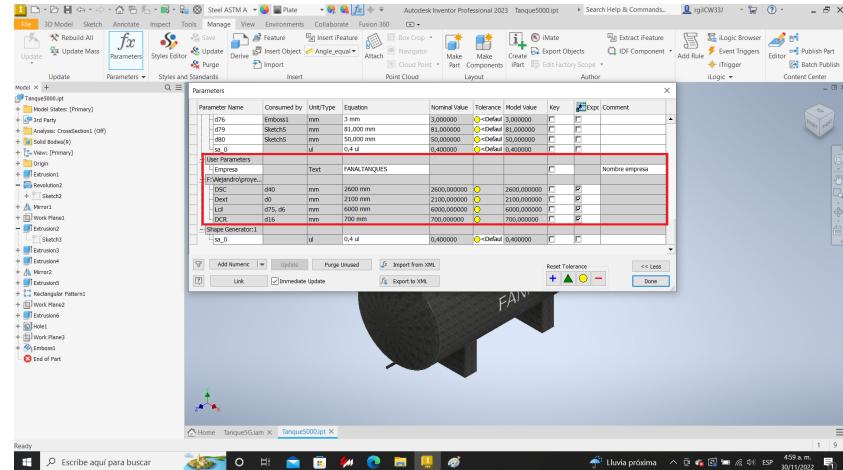


Figura 26: Medidas parametrizadas definidas en Autodesk Inventor.

Figura 27: Medidas parametrizadas definidas en Excel.

Respecto a los planos obtenidos de parte de la empresa habían dimensiones para tanques de 5000, 10000 y 12000 galones, pero no habían intermedios, así que por el uso de la misma hoja de excel que se observa se realizaron la automatización de estas medidas respecto a un valor de galonaje cualquiera entre 3000 y 12000 galones, para así poder variar el tamaño lo mas personalizable que se podía, y como detalle extra se coloca el nombre de la empresa al lado del tanque.

Cabe mencionar que los parámetros deben tener el mismo nombre para poder relacionar los archivos, y que en la primera de estas dos figuras previas se puede observar un botón denominado "Link", donde se selecciona el archivo de excel que se desea relacionar.

Para lo que respecta a la automatización, esta es especialmente importante en la generación de planos, donde el objetivo es que se actualicen los planos cuando cambie alguna de las medidas del tanque, para que luego se genere un solo archivo pdf con todos los planos incluidos. Para tal fin se consideró apropiado programar reglas a través de iLogic, una capa basada en VBA que permite programar reglas y a través de los disparadores o triggers, se pueden ejecutar. Entonces en la pestaña manage, se pueden observar las reglas programadas pulsando el botón iLogic Browser, comenzando con la última regla listada que se denomina "guardar todo":

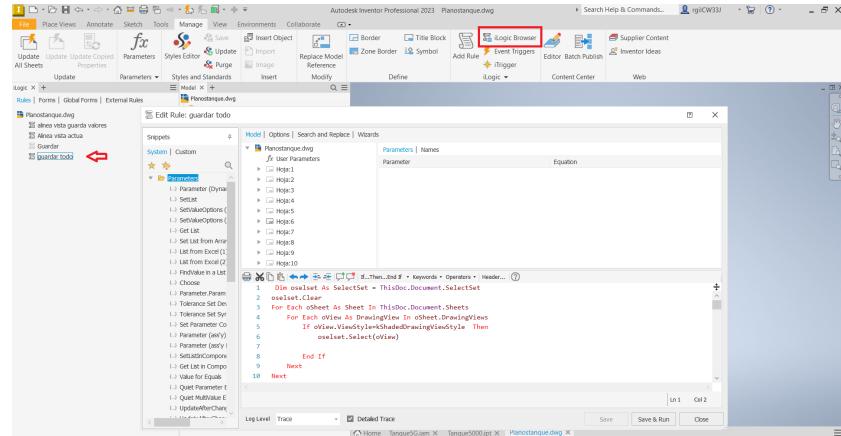


Figura 28: Medidas parametrizadas definidas en Excel.

Se observa entonces la interfaz que se abre al dar click derecho en la regla para editarla, aunque para crear la regla por primera vez, esto se realiza a través del botón Add Rule. A continuación se presenta el código para esta regla:

```

1 Dim oSelset As SelectSet = ThisDoc.Document.SelectSet
2 oSelset.Clear
3 For Each oSheet As Sheet In ThisDoc.Document.Sheets
4     For Each oView As DrawingView In oSheet.DrawingViews
5         If oView.ViewStyle=kShadedDrawingViewStyle Then
6             oSelset.Select(oView)
7
8         End If
9     Next
10
11 ThisApplication.CommandManager.ControlDefinitions.Item("DrawingUpdateViewComponentCmd").Execute
12
13 If ThisDoc.Document.DocumentType <> kDrawingDocumentObject Then
14     MessageBox.Show("This can only be run on drawing documents.", "Error")
15     Exit Sub
16 End If
17
18 Try
19
20 Dim oDraw As DrawingDocument = ThisApplication.ActiveDocument
21 Dim oPDFAddIn As TranslatorAddIn = Nothing
22 Dim oNameValueMap As NameValueMap
23 Dim oContext As TranslationContext
24 Dim oDataMedium As DataMedium
25
26 Dim oPath As String
27
28 'Save to desktop
29 oPath = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Desktop)
30
31 Dim oFileName As String = ThisDoc.FileName(False) 'without extension
32
33 oPDFAddIn = ThisApplication.ApplicationAddIns.ItemById("{0AC6FD96-2F4D-42CE-8BE0
-8AEA580399E4}")
34
35 If oPDFAddIn Is Nothing Then

```

```

36    MsgBox("The DWG add-in could not be found.")
37    Exit Sub
38 End If
39
40 ' Check to make sure the add-in is activated.
41 If Not oPDFAddIn.Activated Then oPDFAddInActivate()
42
43 oContext = ThisApplication.TransientObjects.CreateTranslationContext
44 oContext.Type = IOMechanismEnum.kFileBrowseIOMechanism
45 oOptions = ThisApplication.TransientObjects.CreateNameValuePair
46 oDataMedium = ThisApplication.TransientObjects.CreateDataMedium
47
48 oOptions.Value("All_Color_AS_Black") = 0
49 oOptions.Value("Remove_Line_Weights") = 0
50 oOptions.Value("Vector_Resolution") = 400
51 'oOptions.Value("Sheet_Range") = Inventor.PrintRangeEnum.kPrintCurrentSheet
52 oOptions.Value("Sheet_Range") = Inventor.PrintRangeEnum.kPrintAllSheets
53
54 ' Set the PDF target file name
55 'oDataMedium.FileName = oPath & "\\" & oFileName & ".pdf"
56 oDataMedium.FileName = ThisDoc.PathAndFileName(False) & ".pdf"
57
58 ' Publish document
59 Try
60     oPDFAddIn.SaveCopyAs(oDraw, oContext, oOptions, oDataMedium)
61 Catch ex As Exception
62     MsgBox("Failed To Export: " & FileName)
63 End Try
64
65 'MessageBox.Show("PDF can be found on Desktop", "Export Complete")
66
67 Catch ex As Exception
68     'MessageBox.Show(ex.Message)
69 End Try

```

Se observa entonces que las primeras 10 líneas del código se utilizan para actualizar cada uno de los planos, que están definidos en hojas o "sheets" dentro de un solo archivo .dwg. Luego de esto si comienza la sección del código donde se define la ruta donde se encuentra actualmente el .dwg, que será también la que el programa utilizará para guardar el pdf final, se busca el nombre del archivo .dwg, y se activa el addin denominado oPDFAddIn, el cuál permite realizar esta generación del pdf. Con lo anterior es posible definir las capas de dibujo de los planos, tal y como lo hace inventor cuando se guarda un plano de forma manual, se le da el nombre al pdf a generar y finalmente se salva.

El trigger o activador de esta regla es el hecho de guardar el documento, de modo que cuando se da la instrucción de esto, se comienza a ejecutar la regla mencionada. Esto se configura en el botón de Event Triggers, donde se arrastra la regla a uno o varios activadores que ofrece Autodesk, como se muestra:

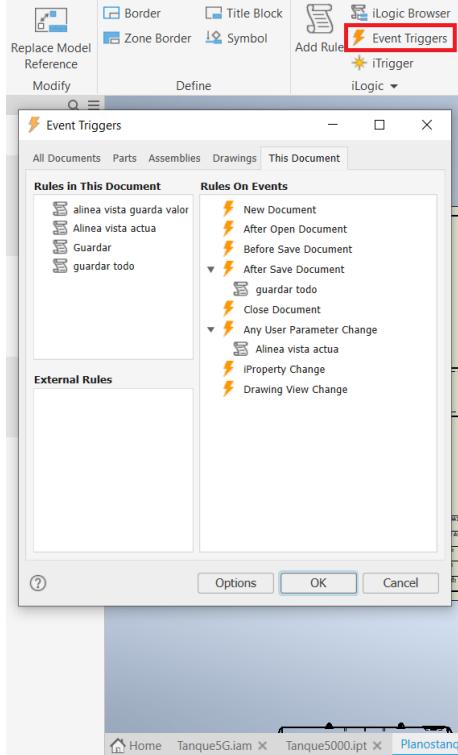


Figura 29: Pestaña de Eventos de activadores.

Para este y los demás códigos de iLogic fue necesario investigar bastante en los foros y ayudas de Inventor y de iLogic, ya que era un lenguaje que nunca se había utilizado antes. Se encontraron entonces algunas porciones de código abierto dadas directamente por Autodesk, que tuvieron que adaptarse a las necesidades del proyecto como tal. Se encontraron algunas porciones de código de utilidad como las que muestran mensajes cuando hay errores en alguna parte de la ejecución de la regla, lo cuál permite identificar los problemas de mejor forma.

Otra de las reglas que se observan son las primeras dos, denominadas ".alinea vista guarda valores" y ".alinea vista actua". Estas se aplican en un plano específico (la hoja 6, donde se encuentra el plano de la saliente del tanque), debido a que cuando cambian las dimensiones de longitud del tanque este tiende a salirse del plano. El primer código se muestra a continuación:

```

1 Sub Main()
2 Dim oDoc As DrawingDocument
3 oDoc = ThisDoc . Document
4
5 'Dim Hoja_6 = ThisDrawing . Sheets . ItemByName( "Hoja :6" )
6 'Dim VIEW14 = Hoja_6 . DrawingViews . ItemByName( "VIEW14" )
7 'Dim Face0 = VIEW14 . GetIntent( "Face0" )
8 'Dim VIEW36 = Hoja_6 . DrawingViews . ItemByName( "VIEW36" )
9 'Dim Edge0 = VIEW36 . GetIntent( "Tanque5000:1" , "Edge0" )
10 'Dim Edge1 = VIEW36 . GetIntent( "Tanque5000:1" , "Edge1" )
11
12
13 Dim oCurve1 , oCurve2 As DrawingCurveSegment
14 oCurve1 = GetCurve1(oDoc)
15 oCurve2 = GetCurve2(oDoc)
16
17 SharedVariable( "oCurve1" ) = oCurve1

```

```

18 SharedVariable("oCurve2")=oCurve2
19
20 End Sub
21
22 Private Function GetCurve1(ByVal oDoc As DrawingDocument) As DrawingCurveSegment
23 Dim Curve As DrawingCurveSegment
24 'Curve = Edge0
25 'Curve=ThisApplication.CommandManager.Pick(SelectionFilterEnum.
26 'kDrawingCurveSegmentFilter, Edge0)
26 Curve = ThisApplication.CommandManager.Pick(SelectionFilterEnum.
27 'kDrawingCurveSegmentFilter, "Select first line to align")
27 Return Curve
28 End Function
29
30 Private Function GetCurve2(ByVal oDoc As DrawingDocument) As DrawingCurveSegment
31 Dim Curve As DrawingCurveSegment
32 'Curve= Face0
33 'Curve=ThisApplication.CommandManager.Pick(SelectionFilterEnum.
34 'kDrawingCurveSegmentFilter, Face0)
34 Curve=ThisApplication.CommandManager.Pick(SelectionFilterEnum.
35 'kDrawingCurveSegmentFilter, "Select second line to align")
35 Return Curve
36 End Function

```

Este código se tuvo que realizar porque para la primera vez que se abre el archivo, se requiere con este código alinear de forma manual el borde izquierdo de la pieza con uno de los bordes de otra vista que no aparece como tal en el plano, sino que tiene la única función de mantener la pieza centrada. Sin embargo, una vez se realiza esta alineación manual, siempre que no se cierre el archivo, se ejecuta el segundo código, donde ya tiene almacenados los datos de los bordes que se utilizan para la alineación como se muestra,:

```

1 Sub Main()
2 Dim oDoc As DrawingDocument
3 oDoc = ThisDoc.Document
4
5 If SharedVariable.Exists("oCurve1") And SharedVariable.Exists("oCurve2") Then
6 Dim oCurve1, oCurve2 As DrawingCurveSegment
7 oCurve1 = SharedVariable("oCurve1")
8 oCurve2 = SharedVariable("oCurve2")
9
10 Dim oView1, oView2 As DrawingView
11 oView1 = oCurve1.Parent.Parent
12 oView2 = oCurve2.Parent.Parent
13 Dim Curve1Point1, Curve1Point2, Curve2Point1, Curve2Point2, View1Point,
14     View2Point As Point2d
14 Curve1Point1=oCurve1.StartPoint
15 Curve1Point2=oCurve1.EndPoint
16 Curve2Point1=oCurve2.StartPoint
17 Curve2Point2=oCurve2.EndPoint
18 If oView1.Name = oView2.Name Then
19     MessageBox.Show("Select lines from different views", "Align view error",
20                     MessageBoxButtons.OK, MessageBoxIcon.Hand, MessageBoxDefaultButton.Button1)
20 Exit Sub
21 End If
22 If (Round((Curve1Point1.X - Curve1Point2.X)*1e8) = 0 And Round((Curve2Point1.X -
23     Curve2Point2.X)*1e8) = 0) Then
23 MoveView = Curve1Point1.X - Curve2Point1.X

```

```

24 oView2Point = oView2.Position
25 oView2Point.X = oView2Point.X + MoveView
26 oView2.Position = oView2Point
27 Else If (Round((Curve1Point1.Y - Curve1Point2.Y)*1e8) = 0 And Round((Curve2Point1
    .Y - Curve2Point2.Y)*1e8) = 0) Then
28 MoveView = Curve1Point1.Y - Curve2Point1.Y
29 oView2Point = oView2.Position
30 oView2Point.Y = oView2Point.Y + MoveView
31 oView2.Position = oView2Point
32 Else
33 MessageBox.Show("not in same orientation", "Aline view error", MessageBoxButtons.
    OK, MessageBoxIcon.Hand, MessageBoxButtons.DefaultButton.Button1)
34 MsgBox(Curve1Point1.Y - Curve1Point2.Y)
35 MsgBox(Curve2Point1.Y - Curve2Point2.Y)
36 Exit Sub
37 End If
38 End If
39
40 End Sub

```

Esta es la razón por la cuál se tienen 2 códigos separados en lugar de 1 solo, y por la cuál se utiliza la función SharedVariable, la cuál permite transferir variables entre diferentes reglas como es el caso. Se intentó conocer qué datos exactamente guarda para identificar las líneas que se utilizan para alinear el dibujo, pero por el tipo de variable que son, no permite mostrarlos en pantalla, por lo cuál se acudió a este método.

Con lo anterior en cuenta, lo que realiza esta segunda regla es guardar las curvas compartidas por SharedVariable, para luego extraer las propiedades de estas, como la vista a la que pertenecen, los puntos iniciales y finales, para luego realizar la alineación con la función Position, en la dirección X. Cabe aclarar que por la forma en que está escrito el código, en la primera regla se debe seleccionar primero el borde de la vista auxiliar y luego el de la vista que si se mostrará en el plano. El trigger de este código es cuando se presenten cambios en los parámetros de usuario, pero no se ha logrado que efectivamente realice dicha actualización de forma 100 % automática, sino que requiere dar click en el botón ittrigger para que funcione.

Debido a lo anterior se considera que una solución es omitir este plano y esta regla, y cambiar el trigger de la regla que actualiza y guarda todos los planos en un solo PDF, de modo que este proceso inicie apenas se abra el documento .dwg, con la opción After Open Document.

- Extracción del BOM

El BOM dentro de Inventor es conocido como la lista de materiales, esta lista contiene información importante de todas las piezas del ensamblado como el nombre, cantidad, costo, entre otras características, esta normalmente es exportada hacia excel con el objetivo de facilitar el manejo y uso de los elementos que se encuentran allí.

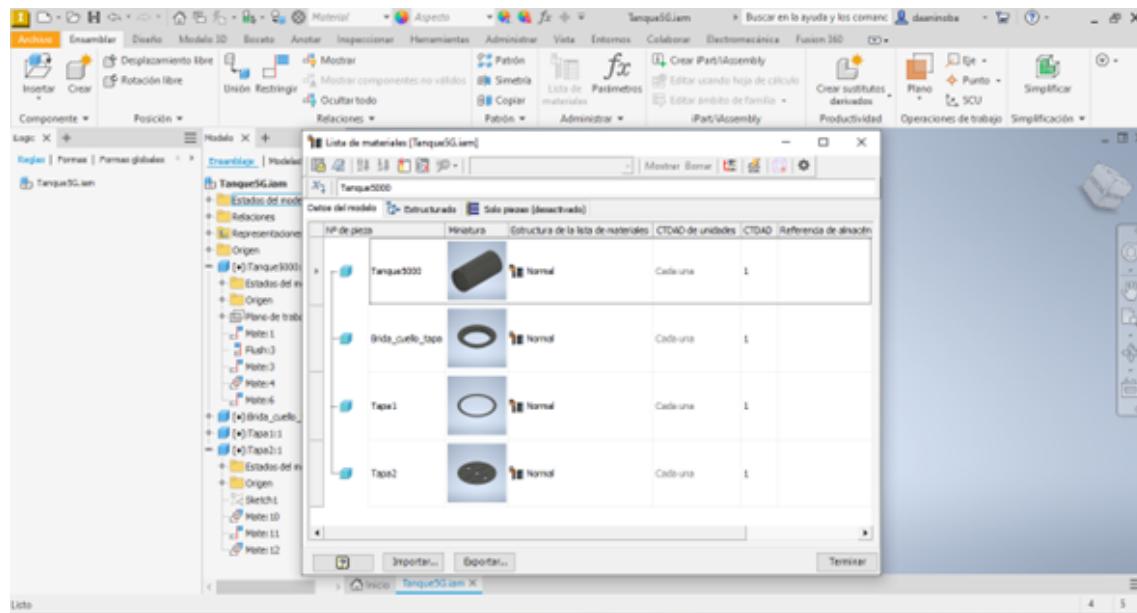


Figura 30: Hoja del BOM en Inventor.

El BOM fue exportado a excel a través de un código hecho en visual basic dentro de inventor, este código posteriormente fue transformado al visual basic de excel con el fin de lograr automatizar este proceso de exportación y poder ser utilizado para cualquier modelado, sin necesidad de abrir Inventor. El código se presenta a continuación:

```

1 Public Sub ExportarBOM()
2     Dim oDoc As AssemblyDocument
3     Set oDoc = ThisApplication.ActiveDocument
4
5     Dim oBOM As BOM
6     Set oBOM = oDoc.ComponentDefinition.BOM
7     oBOM.StructuredViewFirstLevelOnly = False
8     oBOM.StructuredViewEnabled = True
9
10    Dim oStructuredBOMView As BOMView
11    Set oStructuredBOMView = oBOM.BOMViews.Item("Estructurado")
12    oStructuredBOMView.Export "C:\Users\Personal\Desktop\BOM.xlsx",
13                                kMicrosoftExcelFormat
14 End Sub

```

En este código es importante tener en cuenta en que lenguaje está instalado el programa Inventor, si este está en inglés en la linea 11 se debe escribir "Structured", por su parte, si el programa Inventor está en español en esta linea debe estar *Estructurado* como aparece en el código de muestra. Así mismo, el primer parámetro de la función export es la ruta donde queremos que quede guardado el BOM, el segundo parámetro hace referencia a en que formato queremos que este la información exportada.

■ Parámetros adicionales en el BOM

Normalmente dentro del BOM propiedades como el costo o el diseñador de la pieza no aparecen, estas propiedades se añaden de manera sencilla a este, este proceso será presentado en seguida:

1. Abrir el BOM.

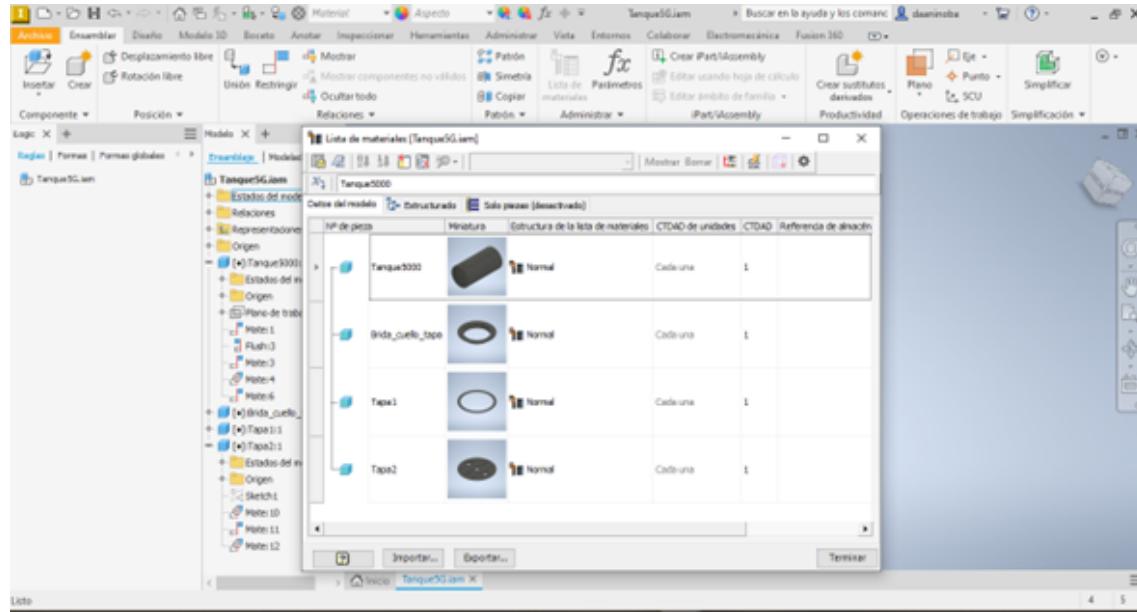


Figura 31: Hoja del BOM en Inventor.

2. Dar click derecho en cualquier parte de la barra de los títulos y dar click izquierdo en personalización de columna... en el menú desplegado.

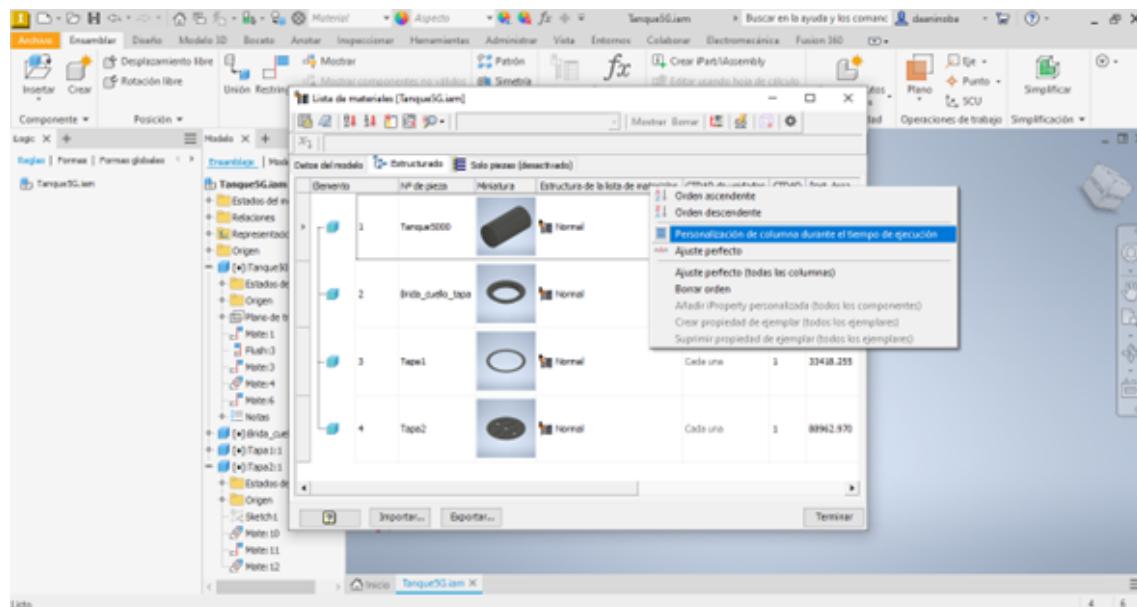


Figura 32: Menú personalización de columna.

3. Finalmente buscar el campo a añadir en la ventana que se abre y arrastrar el campo hasta la barra de títulos (en la imagen como ejemplo esta el campo Ingeniero).

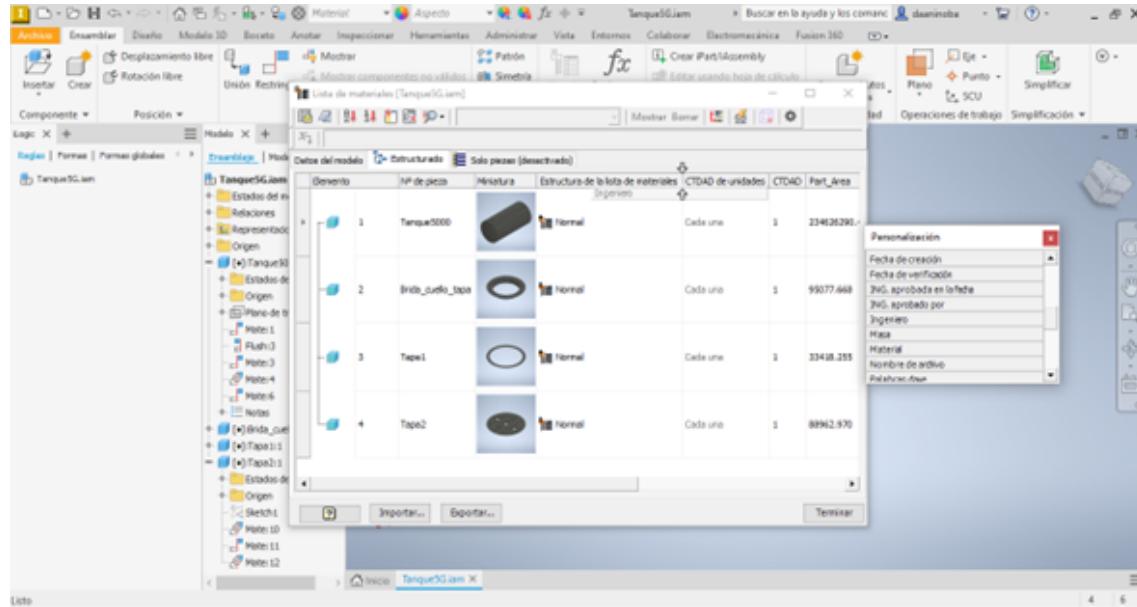


Figura 33: Adición de campo en el BOM.

Por otra parte, también se pueden añadir propiedades físicas de las piezas como masa, área, volumen, entre otras al BOM, para este proceso es necesario el uso de iLogic. El proceso para ello será presentado inmediatamente:

1. Se abre alguna de las piezas del ensamblaje.

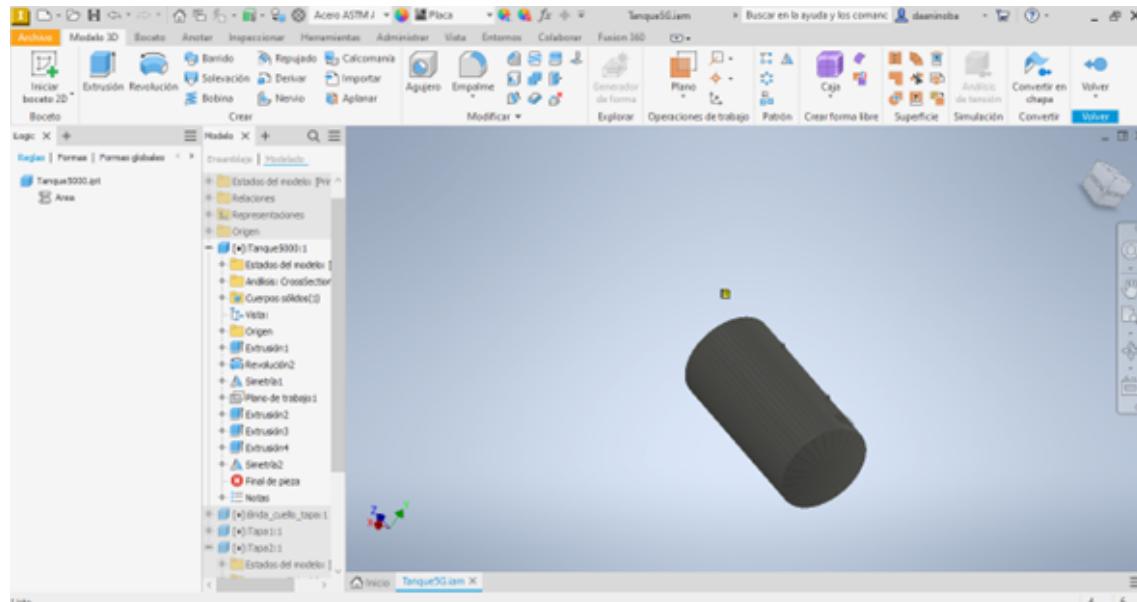


Figura 34: Pieza individual seleccionada.

2. Dar click izquierdo en parámetros.

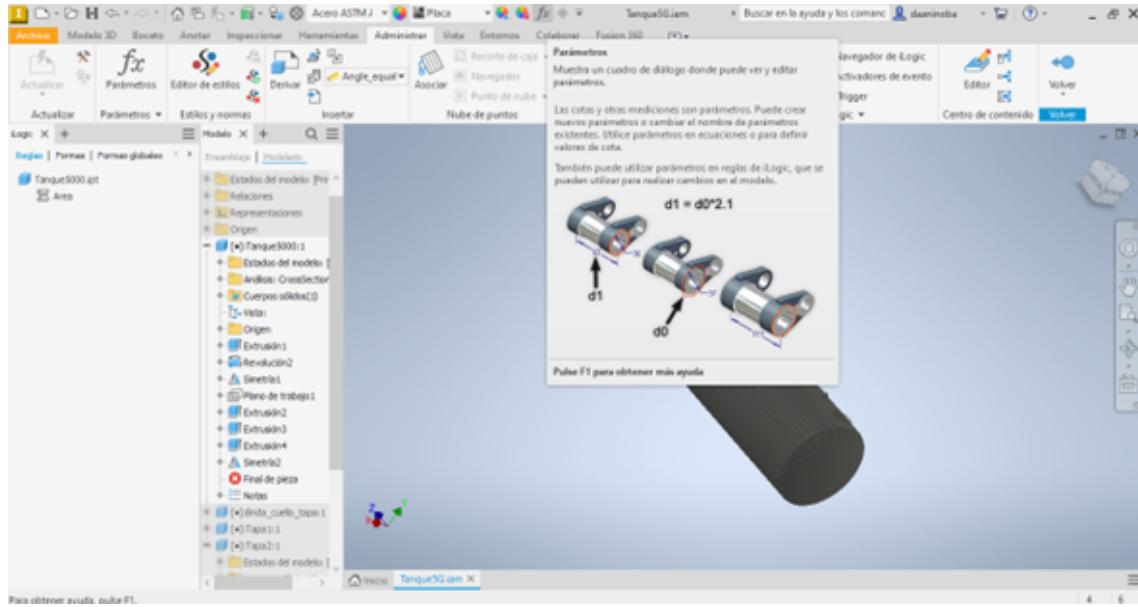


Figura 35: Icono parámetros.

3. Dar click izquierdo en Añadir numérico.

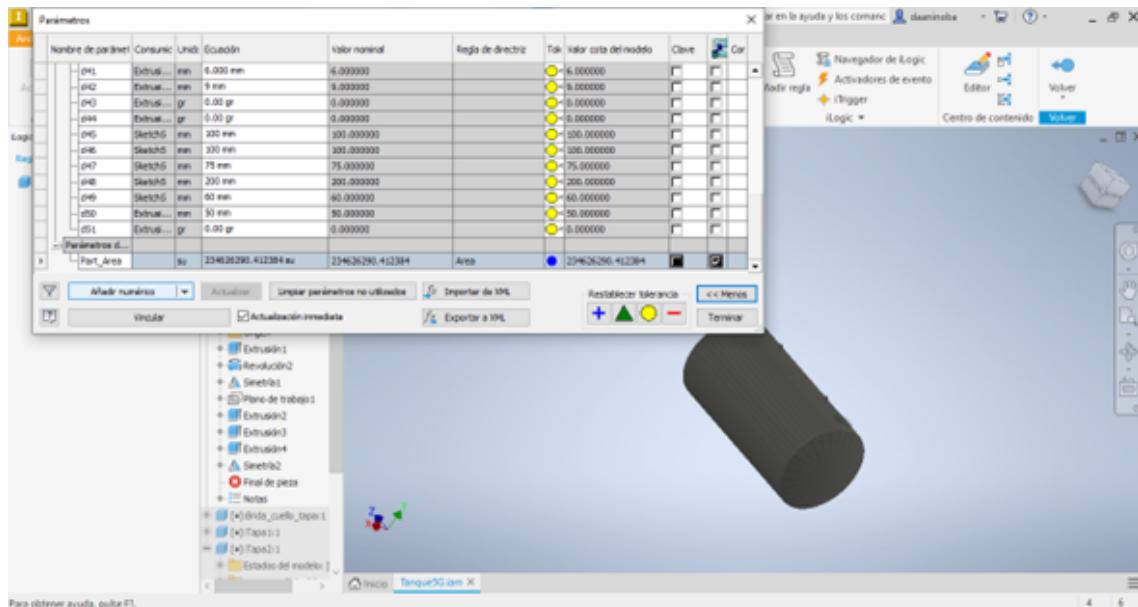


Figura 36: Añadir parámetro.

4. Poner un nombre (como ejemplo Part_Area) y dar click derecho en la columna exportar parámetro.

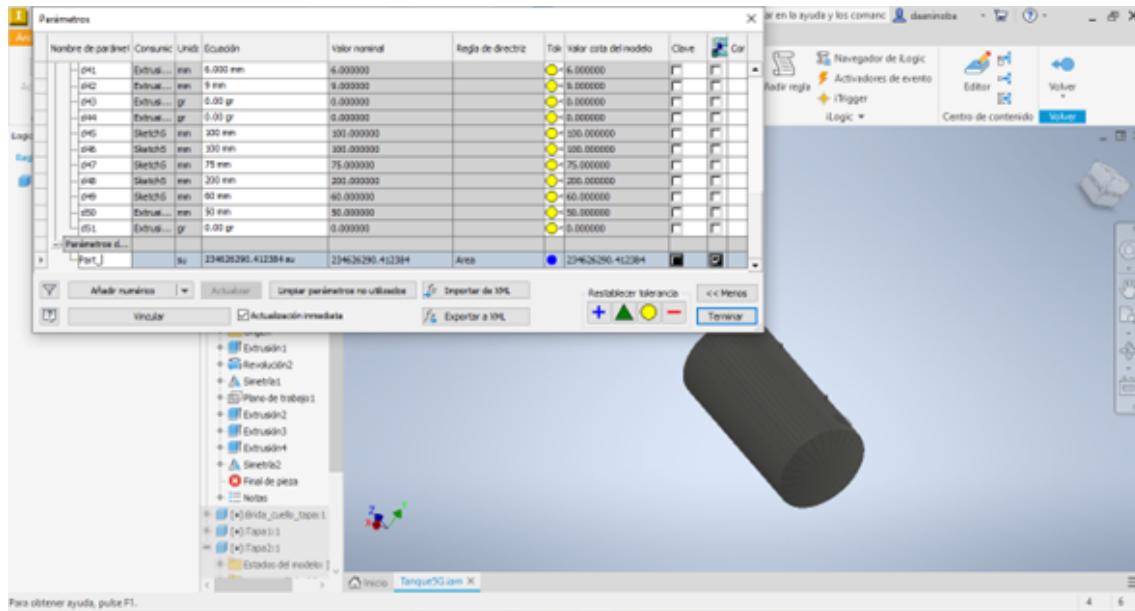


Figura 37: Creación nuevo parámetro.

5. Dar click izquierdo en Navegador de iLogic.

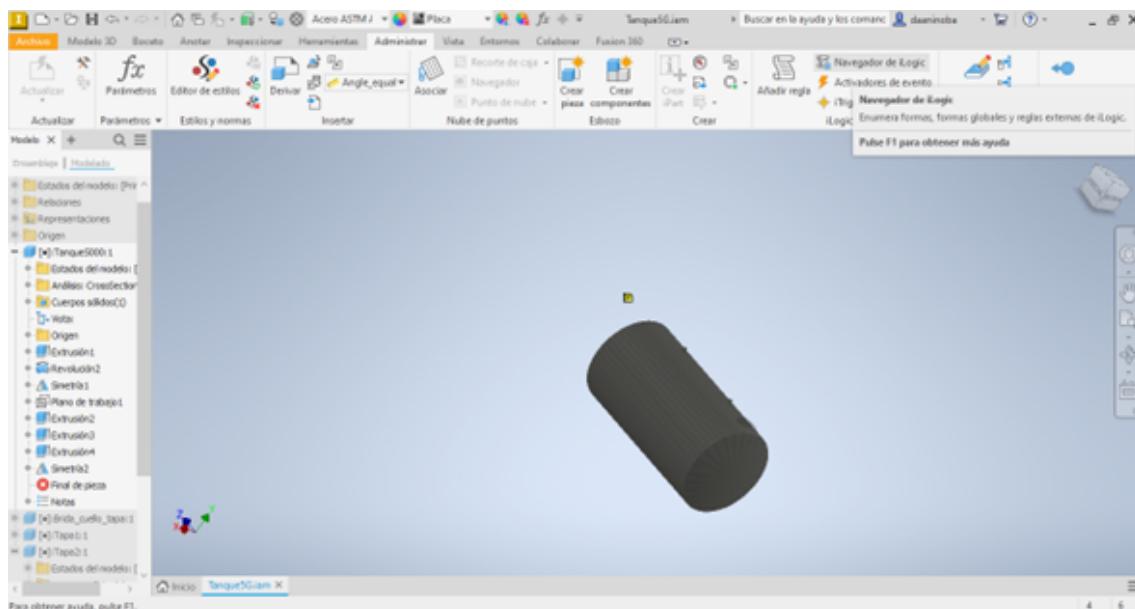


Figura 38: Logo de iLogic.

6. Dentro de la sección reglas en el navegador de Ilogic dar click derecho y luego en el menú desplegado dar click izquierdo en Añadir regla.

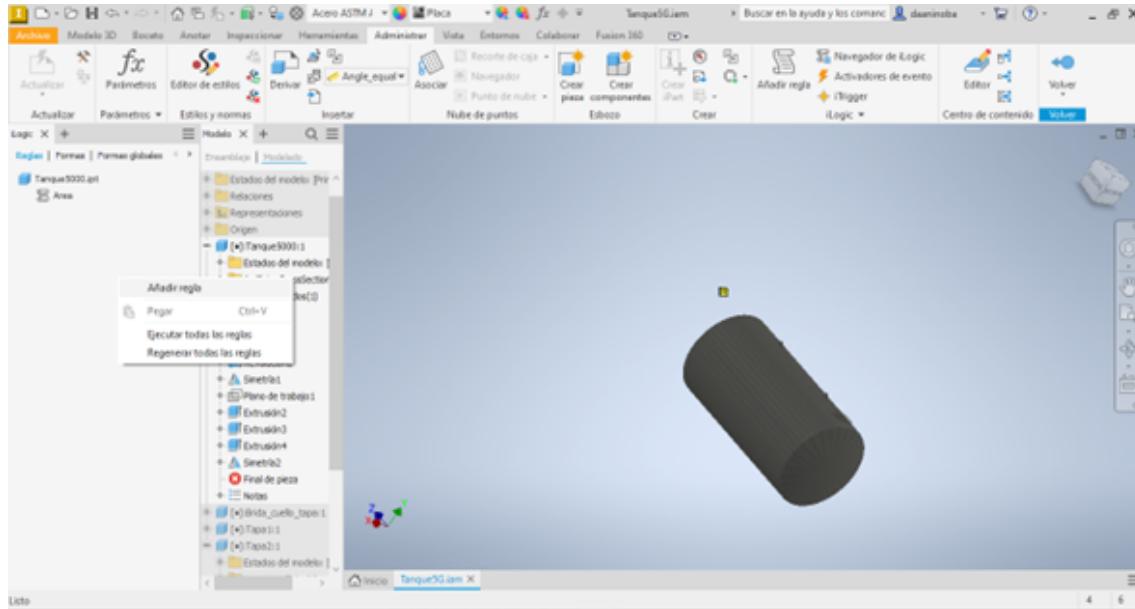


Figura 39: Creación nueva regla.

7. Colocar un nombre adecuado y dar click en aceptar.

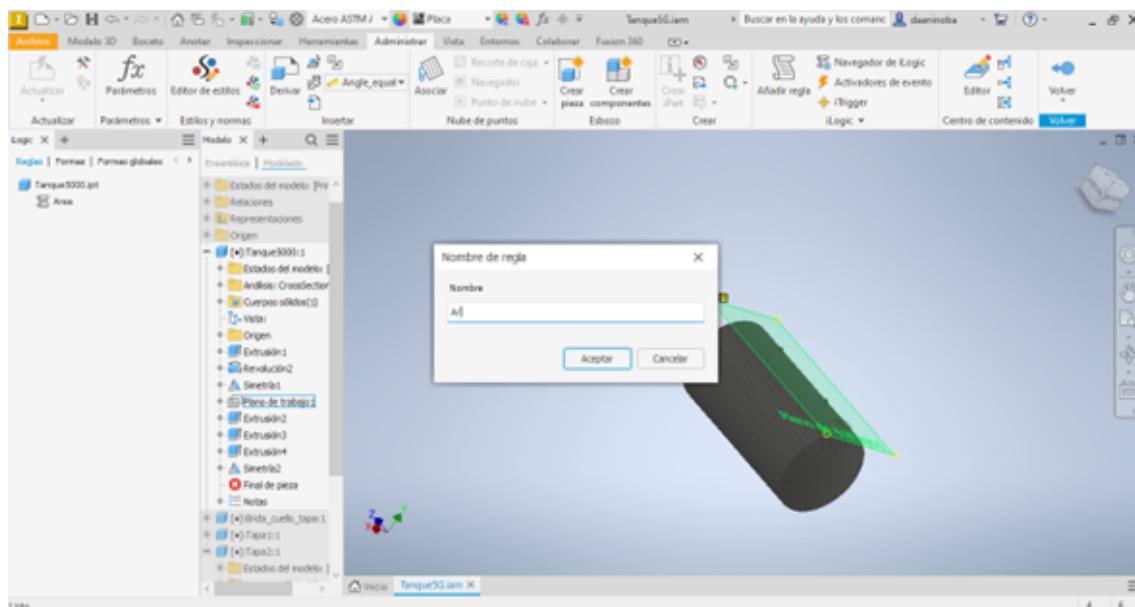


Figura 40: Nombre para regla.

8. Dentro de la ventana que se abre buscar la propiedad de interés en la carpeta iProperties y dar doble click izquierdo en esta propiedad (se debe escribir automáticamente la línea que se visualiza).

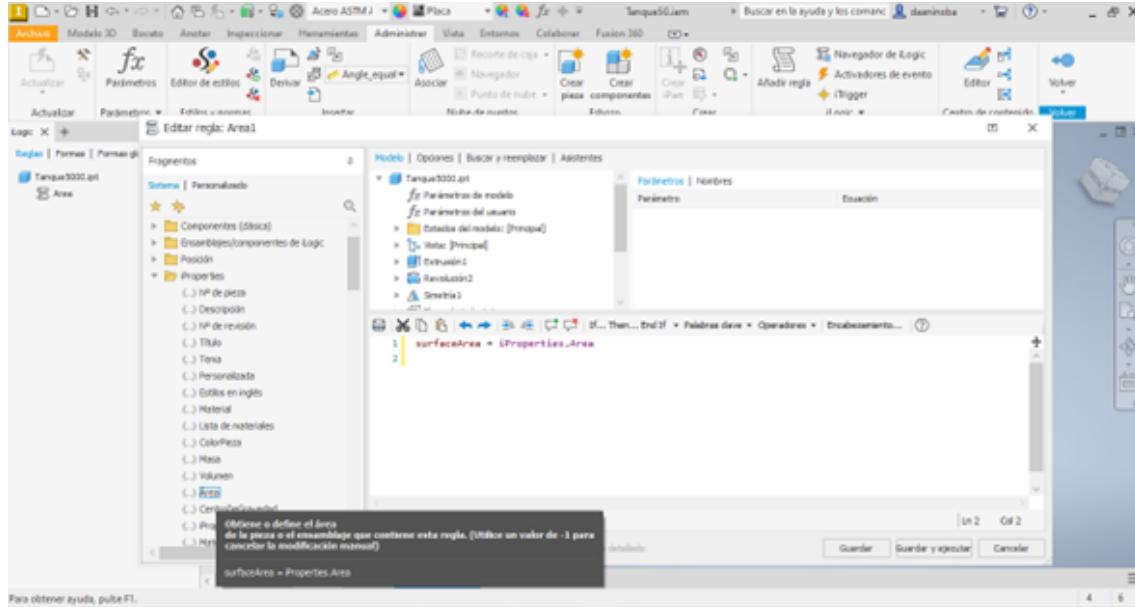


Figura 41: Código para nuevo parámetro.

9. Seleccionar la primera parte de la linea y dar doble clik en parámetros de usuario.

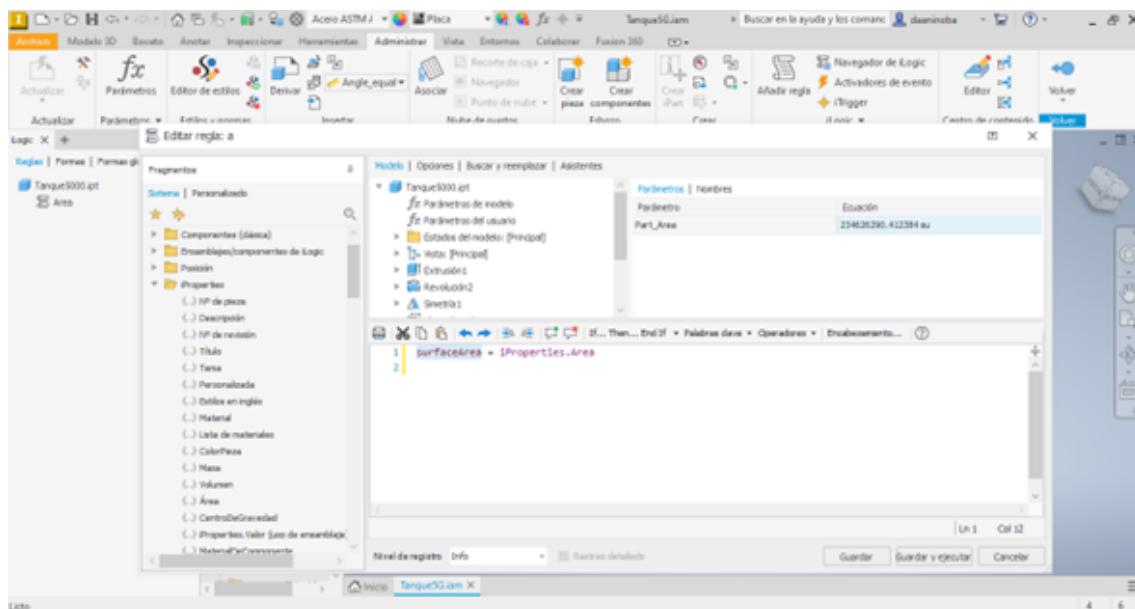


Figura 42: Cambio en la linea.

10. Dar doble clik en el parámetro creado anteriormente y finalmente dar click en guardar y ejecutar.

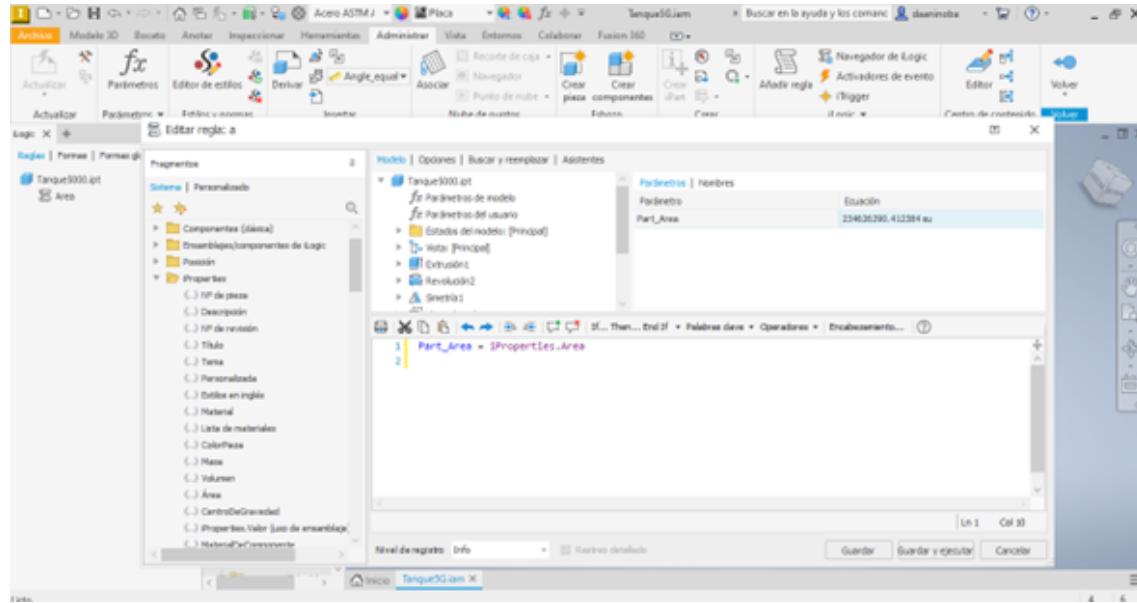


Figura 43: Finalización de lógica en iLogic.

11. Volver al archivo iam.

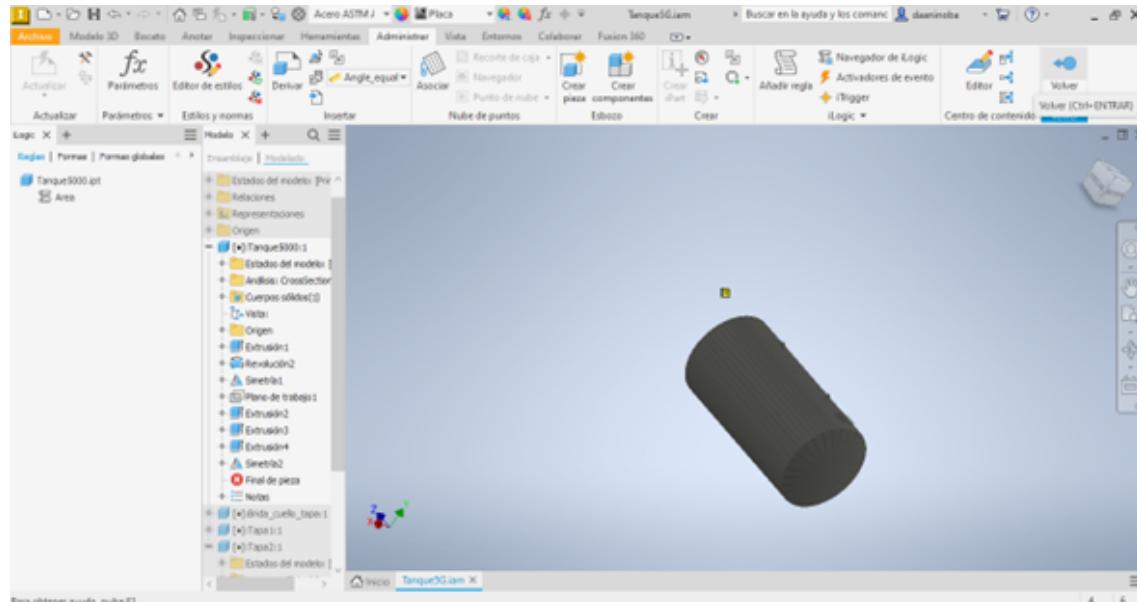


Figura 44: Vuelta al ensamble.

12. Abrir el BOM y dar click en Añadir columnas

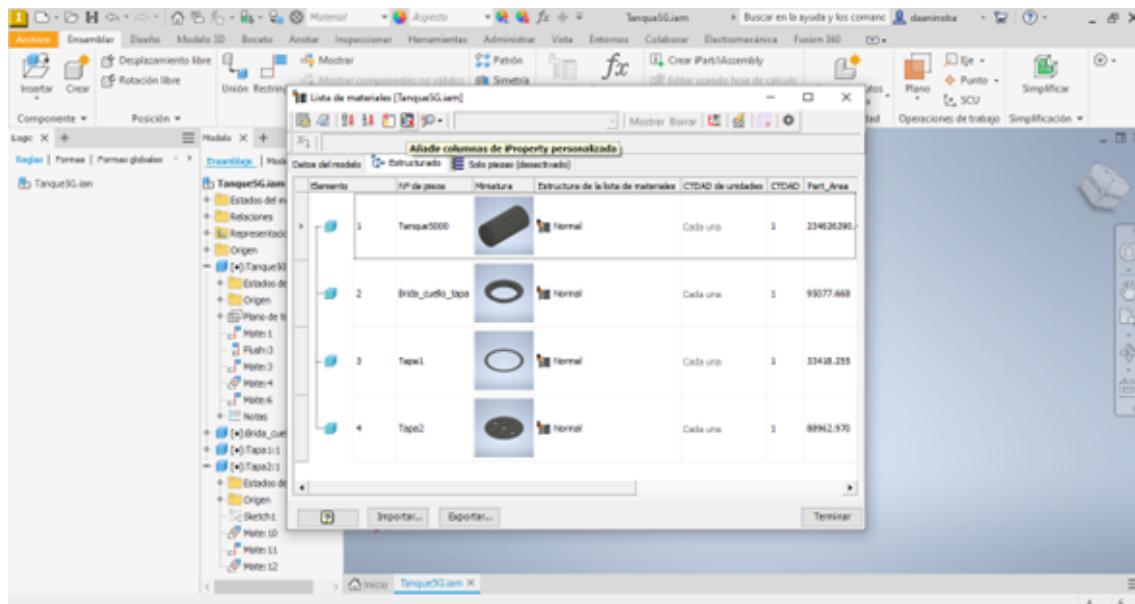


Figura 45: Añadir columna en BOM.

13. Poner el mismo nombre del parámetro creado y seleccionar el tipo de dato.

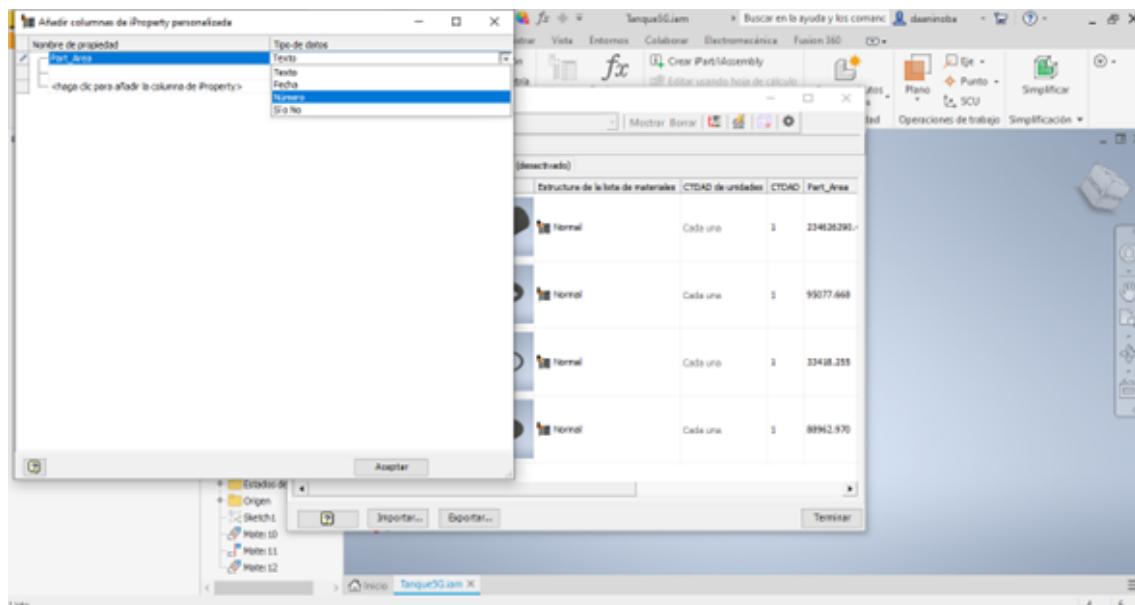


Figura 46: Propiedad física añadida.

- Extracción de modelo 3D en pdf

Una de las características más interesantes que tiene Inventor es lograr crear un archivo pdf en el cual aparezca el modelo de una pieza o un ensamblaje de interés, el modelo que aparece dentro del pdf es interactivo, este puede ser rotado, puesto “invisible” para ver la configuración interna de la pieza, entre otras características. El siguiente código hecho en visual basic dentro de Inventor permite exportar de manera automática este formato:

```

1 Public Sub Exportar3D()
2     Dim oPDFAddIn As ApplicationAddIn
3     Dim oAddin As ApplicationAddIn
4     For Each oAddin In ThisApplication.ApplicationAddIns
5         If oAddin.ClassIdString = "{3EE52B28-D6E0-4EA4-8AA6-C2A266DEBB88}" Then

```

```
6      Set oPDFAddIn = oAddin
7      Exit For
8  End If
9 Next
10
11 If oPDFAddIn Is Nothing Then
12   MsgBox "Inventor 3D PDF Addin not loaded."
13   Exit Sub
14 End If
15
16 Dim oPDFConvertor3D
17 Set oPDFConvertor3D = oPDFAddIn.Automation
18
19 Dim oDocument As Document
20 Set oDocument = ThisApplication.ActiveDocument
21
22 Dim oOptions As NameValueMap
23 Set oOptions = ThisApplication.TransientObjects.CreateNameValueMap
24
25 oOptions.Value("FileOutputLocation") = ("C:\Users\Personal\Desktop\Modelo3D.
26   pdf")
26 oOptions.Value("ExportAnnotations") = 0
27 oOptions.Value("ExportWokFeatures") = 0
28 oOptions.Value("GenerateAndAttachSTEPFile") = False
29 oOptions.Value("LimitToEntitiesInDVRs") = True
30 oOptions.Value("VisualizationQuality") = AccuracyEnum.kVeryHigh
31
32 Dim sProps(0) As String
33 sProps(0) = "{F29F85E0-4FF9-1068-AB91-08002B27B3D9}:Title" ' Title
34
35 oOptions.Value("ExportAllProperties") = False
36 oOptions.Value("ExportProperties") = sProps
37
38 Dim sDesignViews(0) As String
39 sDesignViews(0) = oDocument.ComponentDefinition.RepresentationsManager.
40   ActiveDesignViewRepresentation.Name
41
42 oOptions.Value("ExportDesignViewRepresentations") = sDesignViews
43
44 Call oPDFConvertor3D.Publish(oDocument, oOptions)
End Sub
```

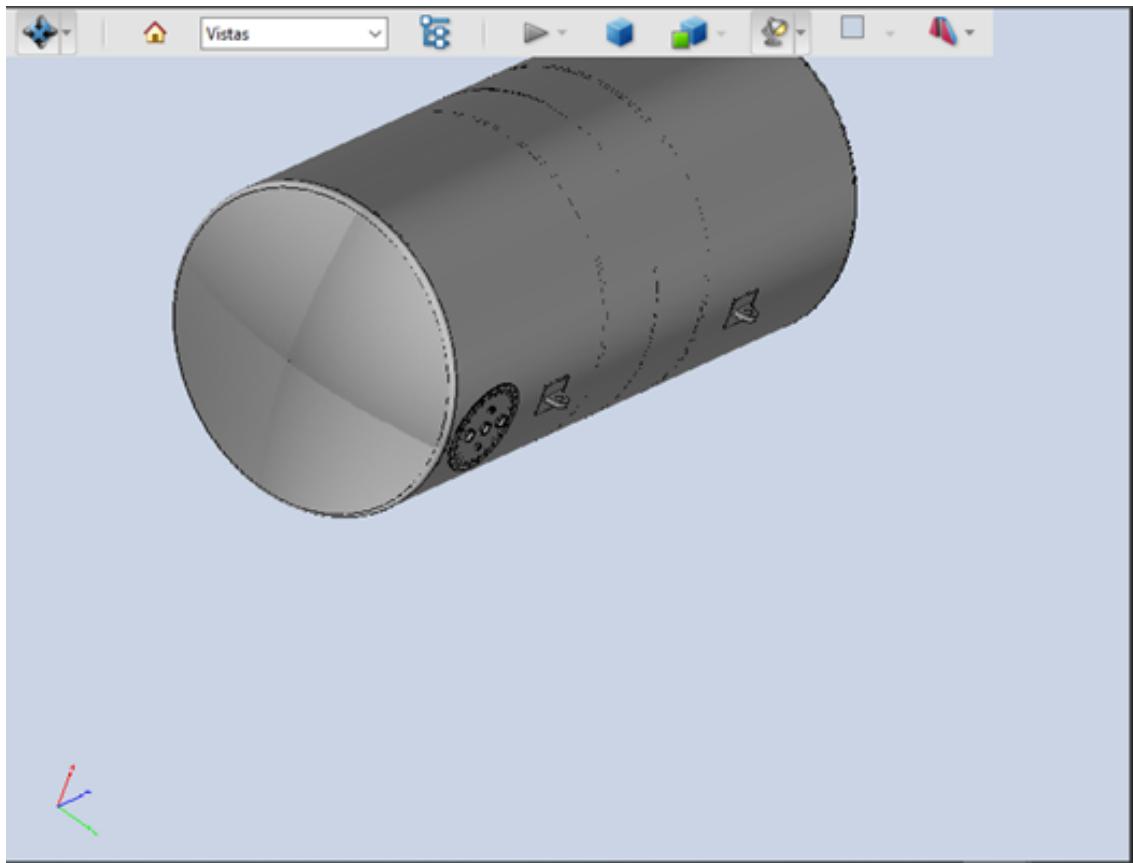


Figura 47: Modelo 3D.

3.1.2. Conclusiones y recomendaciones

- En cuanto al modelado en inventor para una ejecución real del proyecto lo mas recomendable es realizar un diseño de las piezas como chapa y posteriormente realizar un ensamble soldado, esto podría simplificar mucho desarrollos de laminas, mejorar planos, calculos de material y soldadura requerida; pero en este caso no se realizo debido a que parametrizar chapas y este tipo de ensambles conlleva mucha más complejidad y tiempo que no se tuvo.
- El aprendizaje del lenguaje iLogic puede ser de gran utilidad para programar procesos de generación de modelos y planos en Inventor de forma automática. Sin embargo, es importante tener en cuenta que la información acerca de este lenguaje, el cuál es una capa del lenguaje Visual Basic para Aplicaciones enfocada hacia Inventor, no se encuentra clasificada y ordenada como la información para el manejo del lenguaje LISP, por lo cuál los foros de Autodesk son la mejor opción para comenzar a entender la lógica y los comandos requeridos para poder realizar estas ejecuciones como las aplicadas a este proyecto.

3.1.3. Contribuciones

- Juan Sebastián Sánchez M: Realización del modelado CAD en inventor completamente parametrizado y ayuda en conjunto para desarrollo de la hoja de excel parametrizada del modelado de tanques junto con funciones que permitían la creación de tanques de tamaños intermedios.
- David Aguirre Rocha: Optimización y parametrización del modelado final de tanques con la hoja de excel parametrizada, Desarrollo de la hoja de excel parametrizada del modelado de tanques junto con funciones que permitían la creación de tanques de tamaños intermedios, rea-

lización completa de los planos de manufactura del tanque teniendo en cuenta los procesos de manufactura ideales considerados y participación en el intento de automatización de los planos en inventor con reglas de iLogic y vinculación con demás partes del proyecto.

- Raúl Alejandro Gil Zapata: Realización del proceso de automatización de la generación de planos en términos de su actualización y guardado de todos los planos en un solo archivo PDF con el trigger de abrir el archivo, utilizando reglas con el lenguaje de programación iLogic en el archivo .DWG, e intento de alineación de vistas por medio de bordes de los dibujos para uno de los planos. Redacción del apartado de parametrización de planos y de parte de la sección de modelado en el informe final.
- David Alejandro Niño: Automatización de algunos archivos generados que son incluidos en la respuesta para un cliente, a través, de códigos realizados en visual basic dentro de inventor. También realice el proceso de añadir nuevas columnas en el BOM, ya sean propiedades de pieza o propiedades físicas.

3.2. Modelado Tableros

Integrantes:

- Fredy Alexander Castañeda Pereira
- Juan David Diaz Garcia

3.2.1. Descripción

Un gabinete eléctrico o tablero eléctrico es un elemento que proporciona protección de los equipos eléctricos alojados en su interior, como lo son fusibles, protecciones magnetotérmicas y diferenciales. Para el dimensionamiento de un tablero se deben tener en cuenta el número de cuentas, estas varían si son monofásicas, bifásicas o trifásicas, estas dimensiones se ven en el cuadro 1

CÓDIGO	DIMENSIONES (mm) ± 2%		
	ALTO (A)	ANCHO (B)	PROFUNDIDAD (C)
9 – CMON	1750	700	350
12 – CMON	1600	800	400
12 – CBIF	1750	1000	400
12 – CTRI	1750	1000	400
15 – CMON	1600	950	400
15 – CBIF	1750	1200	400
15 – CTRI	1750	1200	400

Cuadro 1: Dimensiones de la caja según el número de acometidas

Como proyecto se pretende entregar un plano con las dimensiones necesarias al cliente según sus requerimientos, entiéndase como requerimientos el número de cuentas que el desee instalar, además de una cotización general, así como el desarrollo de las láminas.

3.2.2. Metodología

Para la creación del modelado se partió de la base dada por el cliente, la cual se puede ver en la figura 48.

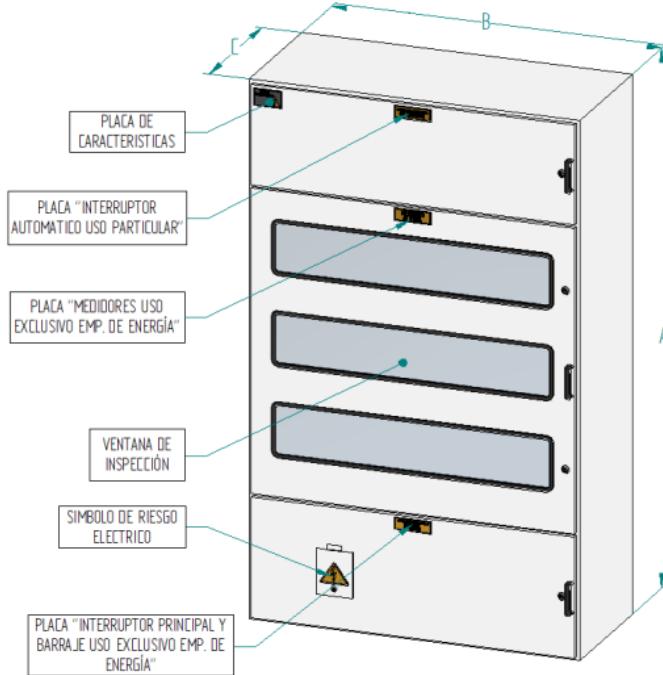


Figura 48: Plano base para el modelado.

Para el dimensionamiento se toman los datos del cliente, los cuales son extraídos de un archivo .xlsx el cual proviene del formulario web, los datos se pueden ver en la figura 49.

	A	B
1		15 Monofasicas
2		10 Bifasicas
3		8 Trifasicas
4		5 Bandejas

Figura 49: Datos obtenidos de la pagina web.

Un código de Matlab realiza la labor de que en base al número de cuentas escogidas por el cliente, se calcula el alto, ancho y largo de la caja y estos datos se guardan en otro archivo .xlsx llamado datos, estos datos se pueden ver en la figura 50.

	A	B	C
1	Bandejas	5 ul	
2	Alto	2400 mm	
3	Ancho	1900 mm	
4	Space1	250 mm	
5	Space2	350 mm	
6	Space3	350 mm	
7	Space4	350 mm	
8	Space5	350 mm	
9	Space6	0 mm	
10	Space7	0 mm	

Figura 50: Datos generados por el código de Matlab.

Ya con el dimensionamiento de la caja se empezó con el proceso de modelado, para ello se parametrizó todas las piezas en función de las medidas obtenidas por el código de Matlab, tal y como se ve en la figura 51.

Bandejas		ul	5 ul	5,000000
Alto	d1	mm	2400 mm	2400,000000
Ancho	d0	mm	1900 mm	1900,000000
Space1		mm	250 mm	250,000000
Space2		mm	350 mm	350,000000
Space3		mm	350 mm	350,000000
Space4		mm	350 mm	350,000000
Space5		mm	350 mm	350,000000
Space6		mm	0 mm	0,000000
Space7		mm	0 mm	0,000000

Figura 51: Parámetros colocados en inventor.

Después de parametrizar todas las piezas de procedió a hacer el ensamblaje, el cual se puede ver en la figura 58

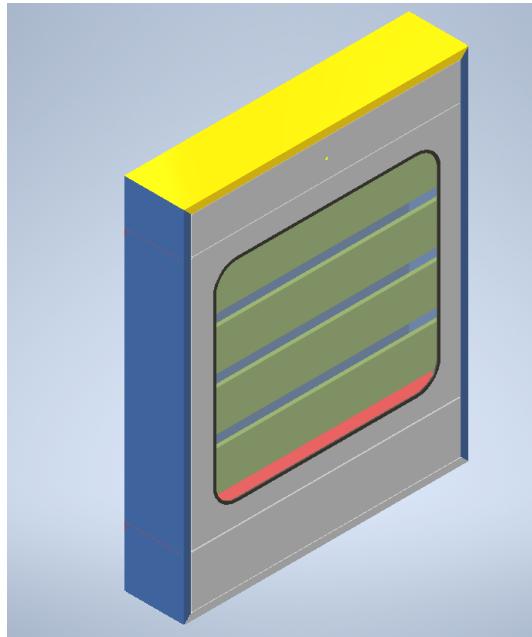


Figura 52: Tablero inventor.

Después de ello se procede a hacer los planos, lo cuales se pueden ver en la sección de planos. Para la parte de exportar el modelado de 3D se procedió a utilizar VB, en nuestro caso tocó instalar el plugin para que funcionara, como se muestra en las figuras 53, 54 y 55.

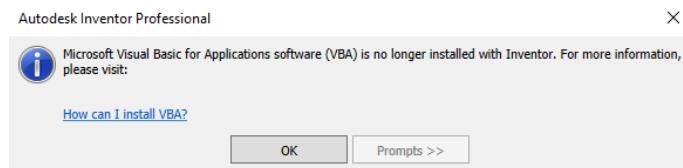
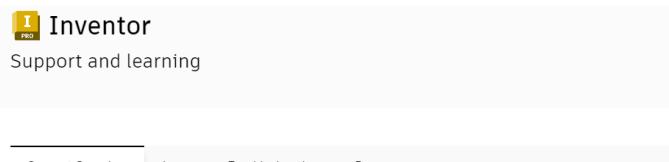


Figura 53: Instalación VB.



Download the Microsoft VBA module for Inventor

Products and versions covered ▾

Figura 54: Instalación VB desde la pagina de Inventor.
39



Figura 55: Instalador.

Ya pudiendo correr VB se desarrollo el código para exportar en 3D, el cual se puede ver en la sección de códigos.

Después con el iLogig se crearon eventos después de abrir el documento, para automatizar todo, como se ve en la figura 56

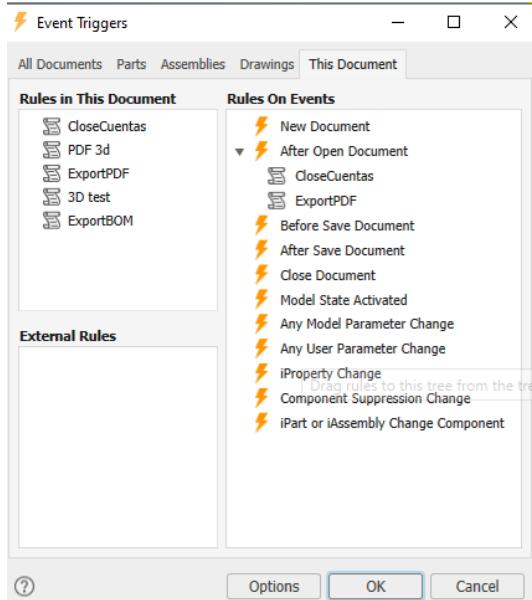


Figura 56: Configuración iLogic.

En la sección de códigos se muestran los códigos creados para crear el pdf 3D, exportar el BOM y crear el pdf, aunque este último no funcionó del todo bien.

3.2.3. Referencias

Como referencia se tomó la ET911 de CODENSA, la cual dice las especificaciones técnicas que deben tener los tableros. Esta a su vez está basada en las siguientes normas:

NORMA	DESCRIPCIÓN
NTC 3444	Armarios para instalación de medidores de energía eléctrica.
NTC 1156	Productos metálicos y recubrimientos. Ensayo de cámara salina.
NTC 3279	Grado de protección dado a los encerramientos. (Código IP).
NTC 912	Método para determinar la dureza de recubrimientos orgánicos con un balancín tipo sward para dureza.
ASTM D 4541	Standard Test Method for Pull-Off Strength of Coatings Using Portable Adhesion Testers

Tabla 2: Normas de fabricación y pruebas
40

3.2.4. Código

A continuación se enseña el código que se utilizo en *Matlab*

```

1 %function []=Parametros_Tableros()
2 clc; clear all
3
4 [num,txt,raw] = xlsread('C:\Users\David\Documents\unal\Computacion grafica\
    Projecto cajas\Modelado\FormularioWeb.xlsx');
5
6 Monofasicas=num(1);
7 bifasicas=num(2);
8 trifasicas=num(3);
9 bandejas=num(4);
10
11 Total=Monofasicas+bifasicas+trifasicas;
12 LongT=Monofasicas*15+(bifasicas+trifasicas)*30;
13 %% Vector de cuentas
14 %cuenta cualquiera [longitud, altura]
15 %selection=menu('Que opinas de este menu', 'Con esta opcion', ' y esta otra opcion
    ');
16 V=[];
17 for i=1:Monofasicas
18     V=[V;[15,25]];
19 end
20 for i=1:bifasicas
21     V=[V;[30,35]];
22 end
23 for i=1:trifasicas
24     V=[V;[30,35]];
25 end
26 %% Calculo Longitud total por bandeja
27 Ncuentas=length(V(:,1));
28 j=1; %Contador de bandejas
29 Long_bandeja=zeros(1,bandejas);
30 Altura_bandeja=zeros(1,bandejas);
31 AlturaMax=0;
32 SumaParcial=0;
33 for i=1:Ncuentas
34     SumaParcial=SumaParcial+V(i,1);
35     if SumaParcial>=(j-1)*LongT/bandejas && SumaParcial<=j*LongT/bandejas
36         Long_bandeja(j)=Long_bandeja(j)+V(i,1);
37         if AlturaMax<V(i,2)
38             AlturaMax=V(i,2);
39             Altura_bandeja(j)=AlturaMax;
40         end
41     else
42         j=j+1;
43         AlturaMax=0;
44         Long_bandeja(j)=Long_bandeja(j)+V(i,1);
45     end
46 end
47
48 end
49 LongT
50 Long_bandeja
51 Altura_bandeja
52 LongTolerance=100;
```

```

53 AltuTolerance=50;
54 CompartimentoAcometidas=300;
55 CompartimentoBarraje=400;
56 LongMax=max( Long_bandeja)*10+LongTolerance;      %max( Long_bandeja ) *10 toma la
   mayero distacion y la pasa a mm
57 AlturaTotal=sum( Altura_bandeja)*10+AltuTolerance;      % Suma las alturas y
   las pasa a mm cajon central
58 AlturaTotal=AlturaTotal+CompartimentoAcometidas+CompartimentoBarraje; % Altura
   Total del tablero
59 filename = 'C:\Users\David\Documents\unal\Computacion grafica\Proyecto cajas\
   Modelado\datos.xlsx';
60 AB=zeros(7,1);          % Establecemos el numero maximo de bandejas
61 AB(1:bandejas,1)=transpose(10*Altura_bandeja);
62 A = { bandejas ; AlturaTotal ; LongMax ;AB(1) ;AB(2) ;AB(3) ;AB(4) ;AB(5) ;AB(6) ;AB(7) };
63 sheet = 1;
64 xlRange = 'B1';
65 xlswrite(filename ,A, sheet ,xlRange)

66
67 AlturaTotal
68 LongMax
69 A

```

Código para exportar el 3D

```

1 Public Sub Exportar3D()
2     Dim oPDFAddIn As ApplicationAddIn
3     Dim oAddin As ApplicationAddIn
4     For Each oAddin In ThisApplication.ApplicationAddIns
5         If oAddin.ClassIdString = "{3EE52B28-D6E0-4EA4-8AA6-C2A266DEBB88}" Then
6             Set oPDFAddIn = oAddin
7             Exit For
8         End If
9     Next
10
11     If oPDFAddIn Is Nothing Then
12         MsgBox "Inventor 3D PDF Addin not loaded."
13         Exit Sub
14     End If
15
16     Dim oPDFConvertor3D
17     Set oPDFConvertor3D = oPDFAddIn.Automation
18
19     Dim oDocument As Document
20     Set oDocument = ThisApplication.ActiveDocument
21
22     Dim oOptions As NameValueMap
23     Set oOptions = ThisApplication.TransientObjects.CreateNameValueMap
24
25     oOptions.Value("FileOutputLocation") = ("C:\Users\David\Documents\unal\
   Computacion grafica\Proyecto cajas\Modelado\Modelo3D.pdf")
26     oOptions.Value("ExportAnnotations") = 0
27     oOptions.Value("ExportWokFeatures") = 0
28     oOptions.Value("GenerateAndAttachSTEPFile") = False
29     oOptions.Value("LimitToEntitiesInDVRs") = True
30     oOptions.Value("VisualizationQuality") = AccuracyEnum.kVeryHigh
31
32     Dim sProps(0) As String
33     sProps(0) = "{F29F85E0-4FF9-1068-AB91-08002B27B3D9}:Title" ' Title

```

```

34
35     oOptions . Value("ExportAllProperties") = False
36     oOptions . Value("ExportProperties") = sProps
37
38     Dim sDesignViews(0) As String
39     sDesignViews(0) = oDocument . ComponentDefinition . RepresentationsManager .
        ActiveDesignViewRepresentation . Name
40
41     oOptions . Value("ExportDesignViewRepresentations") = sDesignViews
42
43     Call oPDFConvertor3D . Publish(oDocument , oOptions)
44 End Sub
45 Public Sub ExportarBOM()
46     Dim oDoc As AssemblyDocument
47     Set oDoc = ThisApplication . ActiveDocument
48
49     Dim oBOM As BOM
50     Set oBOM = oDoc . ComponentDefinition . BOM
51     oBOM . StructuredViewFirstLevelOnly = False
52     oBOM . StructuredViewEnabled = True
53
54     Dim oStructuredBOMView As BOMView
55     Set oStructuredBOMView = oBOM . BOMViews . Item("Structured")
56     oStructuredBOMView . Export "C:\Users\David\Documents\unal\Computacion grafica
        \Proyecto cajas\Modelado\BOM.xlsx" , kMicrosoftExcelFormat
57 End Sub
58
59 Public Sub Exportar3D()
60     Dim oPDFAddIn As ApplicationAddIn
61     Dim oAddin As ApplicationAddIn
62     For Each oAddin In ThisApplication . ApplicationAddIns
63         If oAddin . ClassIdString = "{3EE52B28-D6E0-4EA4-8AA6-C2A266DEBB88}" Then
64             Set oPDFAddIn = oAddin
65             Exit For
66         End If
67     Next
68
69     If oPDFAddIn Is Nothing Then
70         MsgBox "Inventor 3D PDF Addin not loaded."
71         Exit Sub
72     End If
73
74     Dim oPDFConvertor3D
75     Set oPDFConvertor3D = oPDFAddIn . Automation
76
77     Dim oDocument As Document
78     Set oDocument = ThisApplication . ActiveDocument
79
80     Dim oOptions As NameValueMap
81     Set oOptions = ThisApplication . TransientObjects . CreateNameValueMap
82
83     oOptions . Value("FileOutputLocation") = ("C:\Users\David\Documents\unal\
        Computacion grafica\Proyecto cajas\Modelado\Modelo3D.pdf")
84     oOptions . Value("ExportAnnotations") = 0
85     oOptions . Value("ExportWokFeatures") = 0
86     oOptions . Value("GenerateAndAttachSTEPFile") = False
87     oOptions . Value("LimitToEntitiesInDVRs") = True
88     oOptions . Value("VisualizationQuality") = AccuracyEnum . kVeryHigh

```

```

89
90     Dim sProps(0) As String
91     sProps(0) = "{F29F85E0-4FF9-1068-AB91-08002B27B3D9}:Title" ' Title
92
93     oOptions.Value("ExportAllProperties") = False
94     oOptions.Value("ExportProperties") = sProps
95
96     Dim sDesignViews(0) As String
97     sDesignViews(0) = oDocument.ComponentDefinition.RepresentationsManager.
98         ActiveDesignViewRepresentation.Name
99
100    oOptions.Value("ExportDesignViewRepresentations") = sDesignViews
101
102    Call oPDFConvertor3D.Publish(oDocument, oOptions)
103 End Sub
104 Public Sub ExportarBOM()
105     Dim oDoc As AssemblyDocument
106     Set oDoc = ThisApplication.ActiveDocument
107
108     Dim oBOM As BOM
109     Set oBOM = oDoc.ComponentDefinition.BOM
110     oBOM.StructuredViewFirstLevelOnly = False
111     oBOM.StructuredViewEnabled = True
112
113     Dim oStructuredBOMView As BOMView
114     Set oStructuredBOMView = oBOM.BOMViews.Item("Structured")
115     oStructuredBOMView.Export "C:\Users\David\Documents\unal\Computacion grafica
          \Proyecto cajas\Modelado\BOM.xlsx", kMicrosoftExcelFormat
116 End Sub

```

Código para exportar el pdf 3d.

```

1 Public Sub Exportar3D()
2     Dim oPDFAddIn As ApplicationAddIn
3     Dim oAddin As ApplicationAddIn
4     For Each oAddin In ThisApplication.ApplicationAddIns
5         If oAddin.ClassIdString = "{3EE52B28-D6E0-4EA4-8AA6-C2A266DEBB88}" Then
6             Set oPDFAddIn = oAddin
7             Exit For
8         End If
9     Next
10
11     If oPDFAddIn Is Nothing Then
12         MsgBox "Inventor 3D PDF Addin not loaded."
13         Exit Sub
14     End If
15
16     Dim oPDFConvertor3D
17     Set oPDFConvertor3D = oPDFAddIn.Automation
18
19     Dim oDocument As Document
20     Set oDocument = ThisApplication.ActiveDocument
21
22     Dim oOptions As NameValueMap
23     Set oOptions = ThisApplication.TransientObjects.CreateNameValueMap
24
25     oOptions.Value("FileOutputLocation") = ("C:\Users\Personal\Desktop\Modelo3D.
          pdf")

```

```

26     oOptions . Value("ExportAnnotations") = 0
27     oOptions . Value("ExportWokFeatures") = 0
28     oOptions . Value("GenerateAndAttachSTEPFile") = False
29     oOptions . Value("LimitToEntitiesInDVRs") = True
30     oOptions . Value("VisualizationQuality") = AccuracyEnum . kVeryHigh
31
32     Dim sProps(0) As String
33     sProps(0) = "{F29F85E0-4FF9-1068-AB91-08002B27B3D9}:Title" ' Title
34
35     oOptions . Value("ExportAllProperties") = False
36     oOptions . Value("ExportProperties") = sProps
37
38     Dim sDesignViews(0) As String
39     sDesignViews(0) = oDocument . ComponentDefinition . RepresentationsManager .
          ActiveDesignViewRepresentation . Name
40
41     oOptions . Value("ExportDesignViewRepresentations") = sDesignViews
42
43     Call oPDFConvertor3D . Publish(oDocument , oOptions)
44 End Sub

```

Código para exportar pdf

```

1 workspacePath = ThisDoc . WorkspacePath()
2 WorkspacePathLength = Len(WorkspacePath)
3
4 PathOnly = ThisDoc . Path
5 DirectoryPath = Strings . Right(PathOnly , PathOnly . Length - WorkspacePathLength)
6 PDFpath = "C:\Users\David\Documents\unal\Computacion grafica\Proyecto cajas\
      Modelado\Planos" & DirectoryPath
7
8 If (Not System . IO . Directory . Exists(PDFpath)) Then
9     System . IO . Directory . CreateDirectory(PDFpath)
10 End If
11 ThisDoc . Document . SaveAs(PDFPath & "\\" & ThisDoc . FileName(False) & ".pdf" , True)

```

Código para exportar el BOM

```

1 Dim oDoc As AssemblyDocument
2     oDoc = ThisApplication . ActiveDocument
3
4 Dim oBOM As BOM
5     oBOM = oDoc . ComponentDefinition . BOM
6     oBOM . StructuredViewFirstLevelOnly = False
7     oBOM . StructuredViewEnabled = True
8
9 Dim oStructuredBOMView As BOMView
10 Set oStructuredBOMView = oBOM . BOMViews . Item("Estructurado")
11     oStructuredBOMView . Export ("C:\Users\David\Documents\unal\Computacion
      grafica\Proyecto cajas\Modelado\Planos\BOM.xlsx",
      kMicrosoftExcelFormat)
12
13 'ThisBOM . Export("Parts Only" , "fileName" , kMicrosoftExcelFormat)
14 'kMicrosoftAccessFormat = Microsoft Access
15 'kMicrosoftExcelFormat = Microsoft Excel
16 'kDBASEIIIFormat = dBASE III
17 'kDBASEIVFormat = dBASE IV
18 'kTextFileTabDelimitedFormat = Text File Tab Delimited

```

```
19 | 'kTextFileCommaDelimitedFormat      = Text File Comma Delimited  
20 | 'kUnicodeTextFileTabDelimitedFormat = Unicode Text File Tab Delimited  
21 | 'kUnicodeTextFileCommaDelimitedFormat = Unicode Text File Comma Delimited
```

3.2.5. Planos de piezas

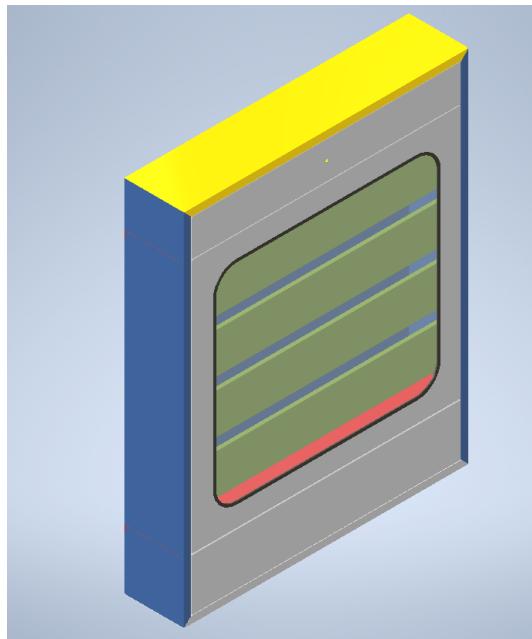


Figura 57: Tablero inventor.

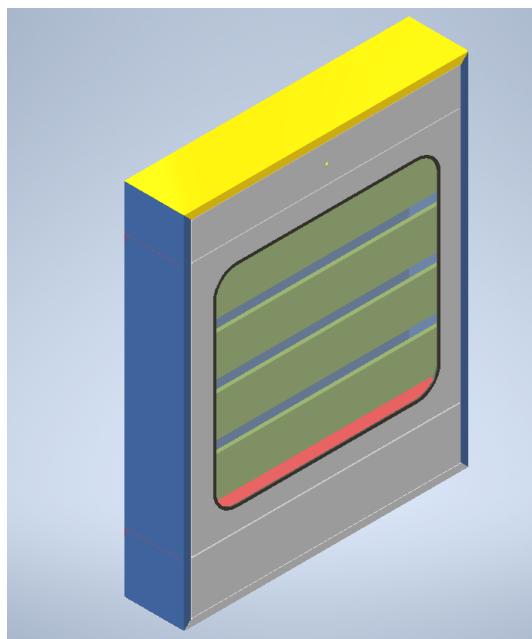


Figura 58: Tablero inventor.

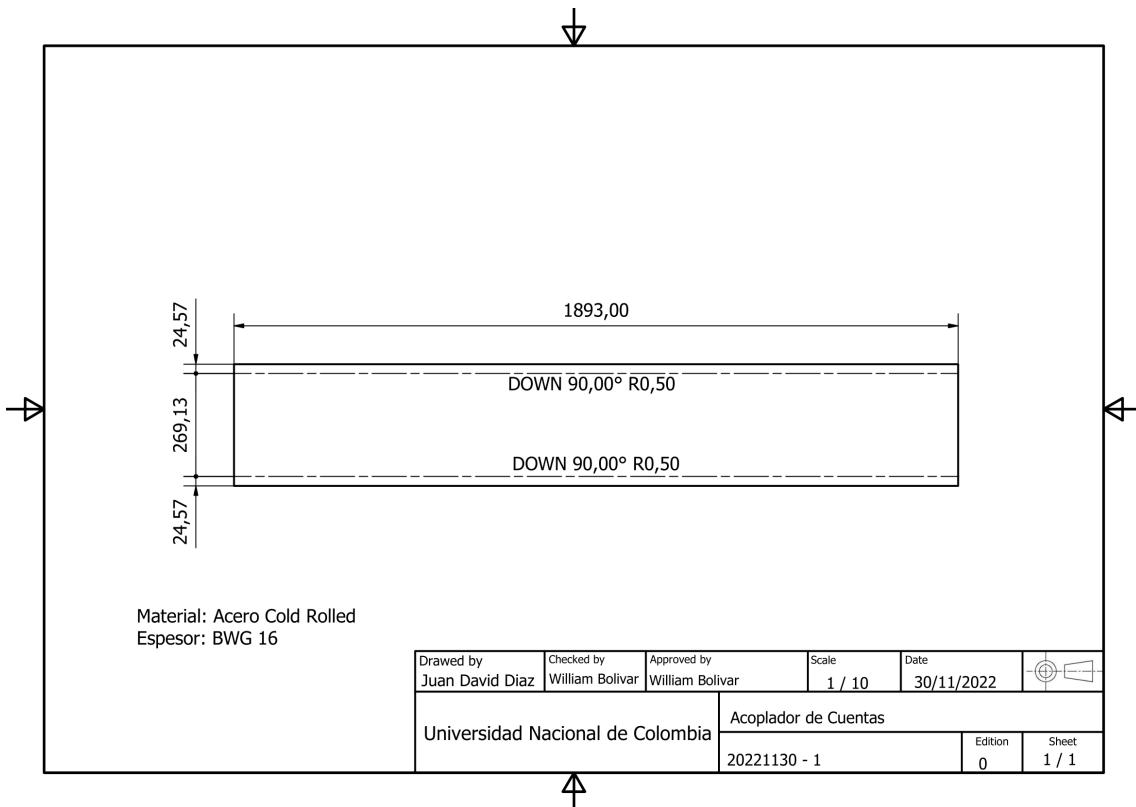


Figura 59: Acoplador de Cuentas

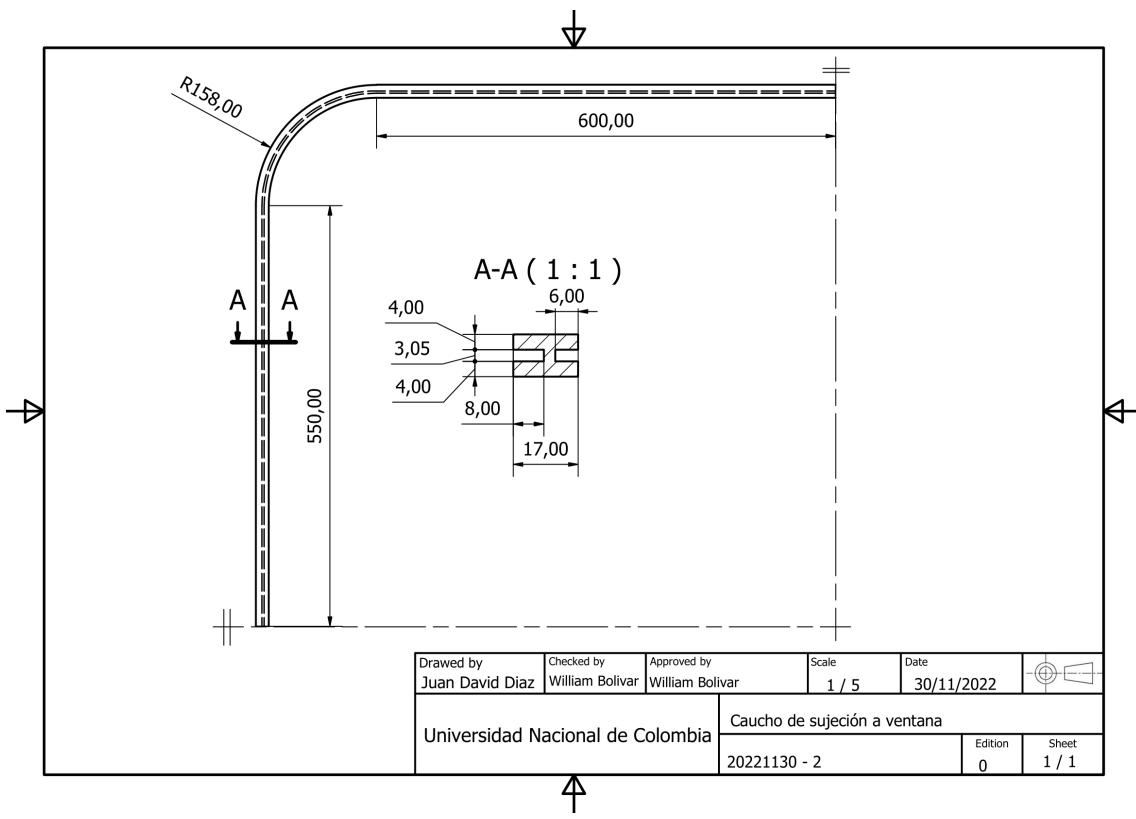


Figura 60: Caucho

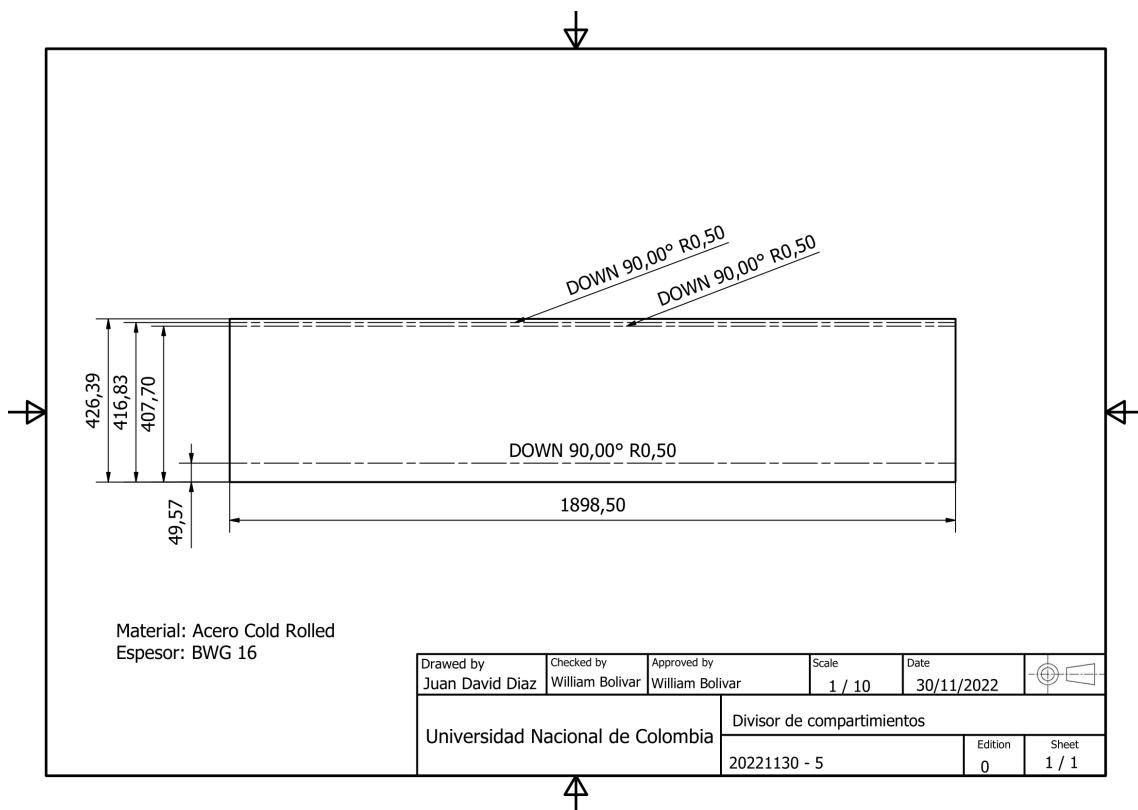


Figura 61: Divisor de compartimientos

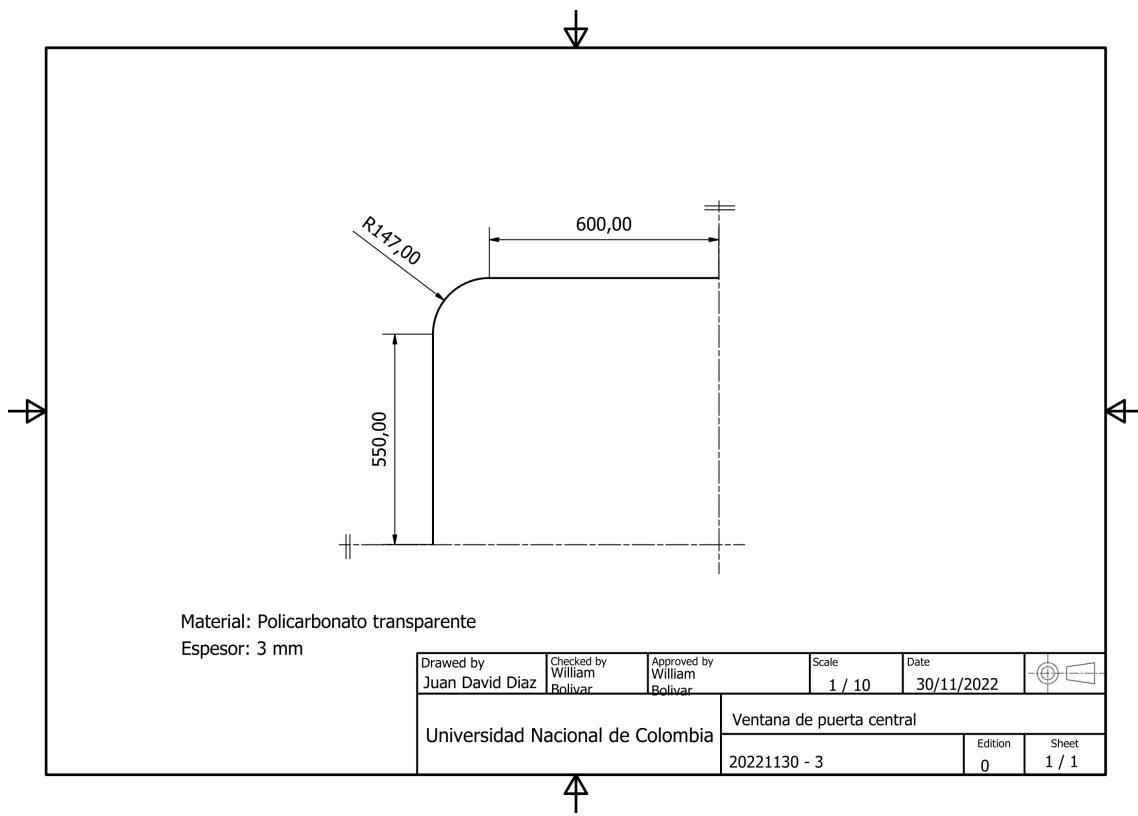


Figura 62: Ventana de visualización

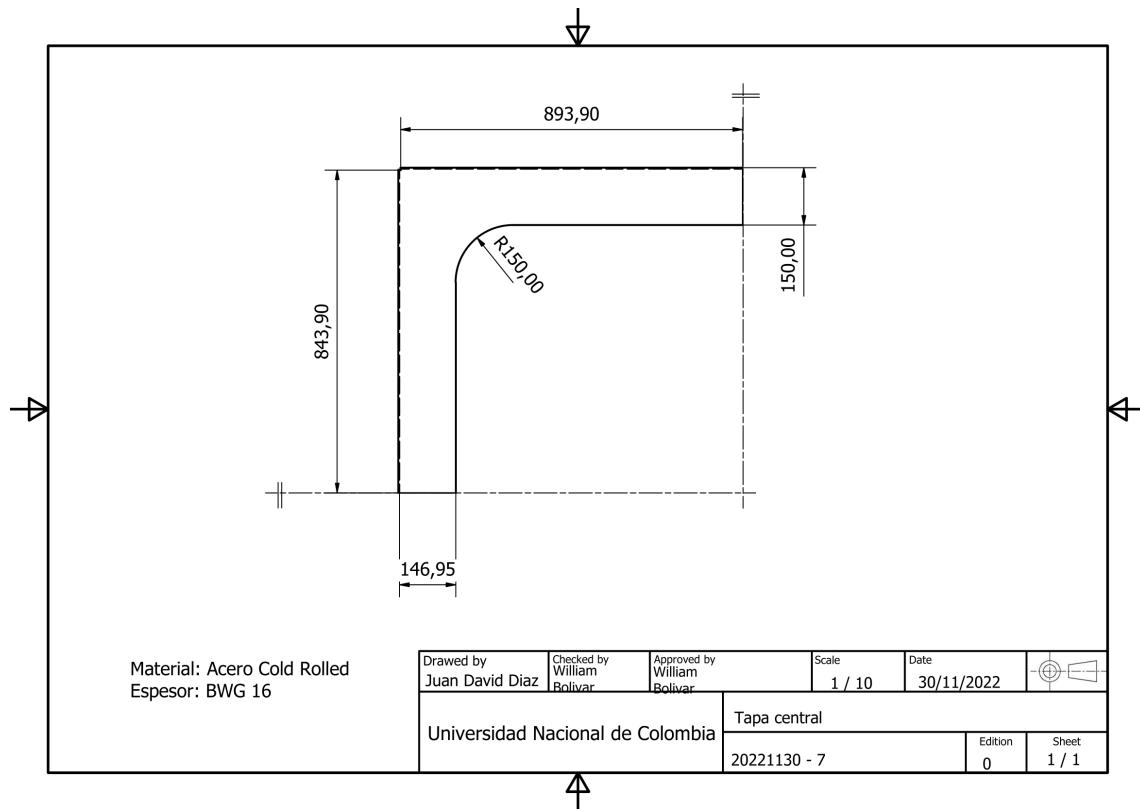


Figura 63: Tapa de compartimiento central

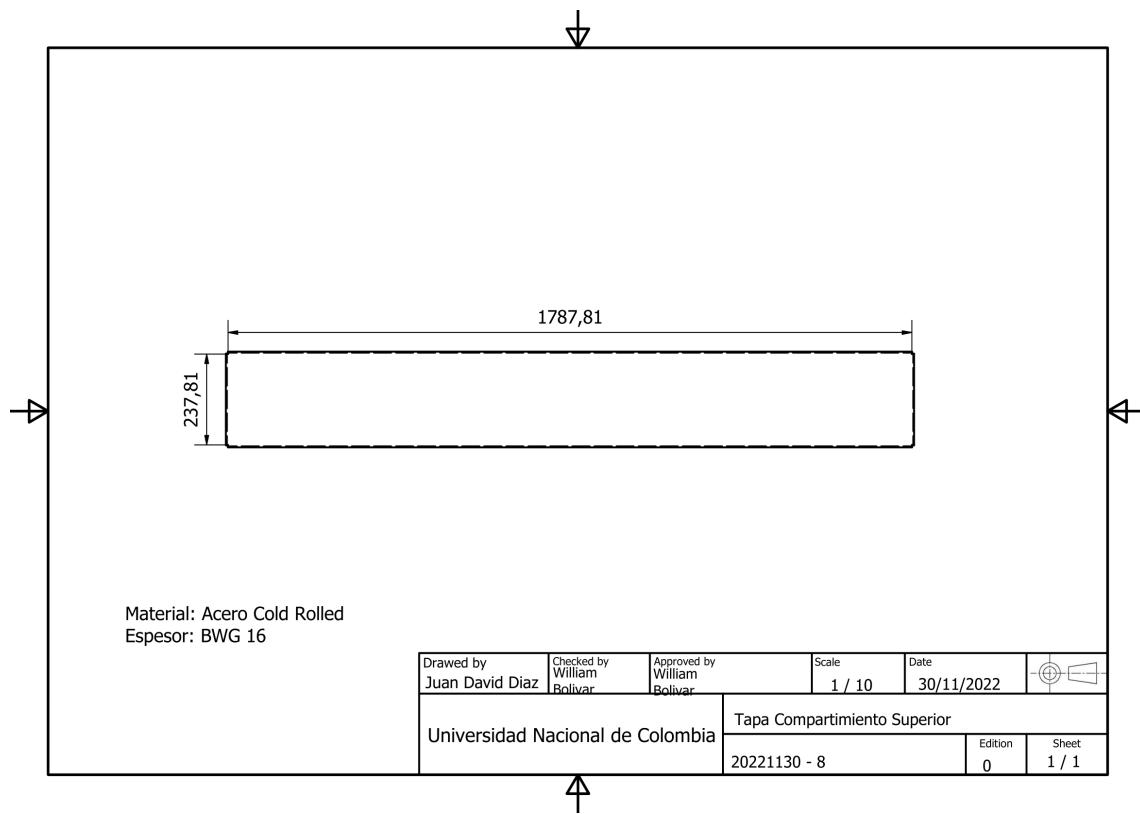


Figura 64: Tapa compartimiento superior

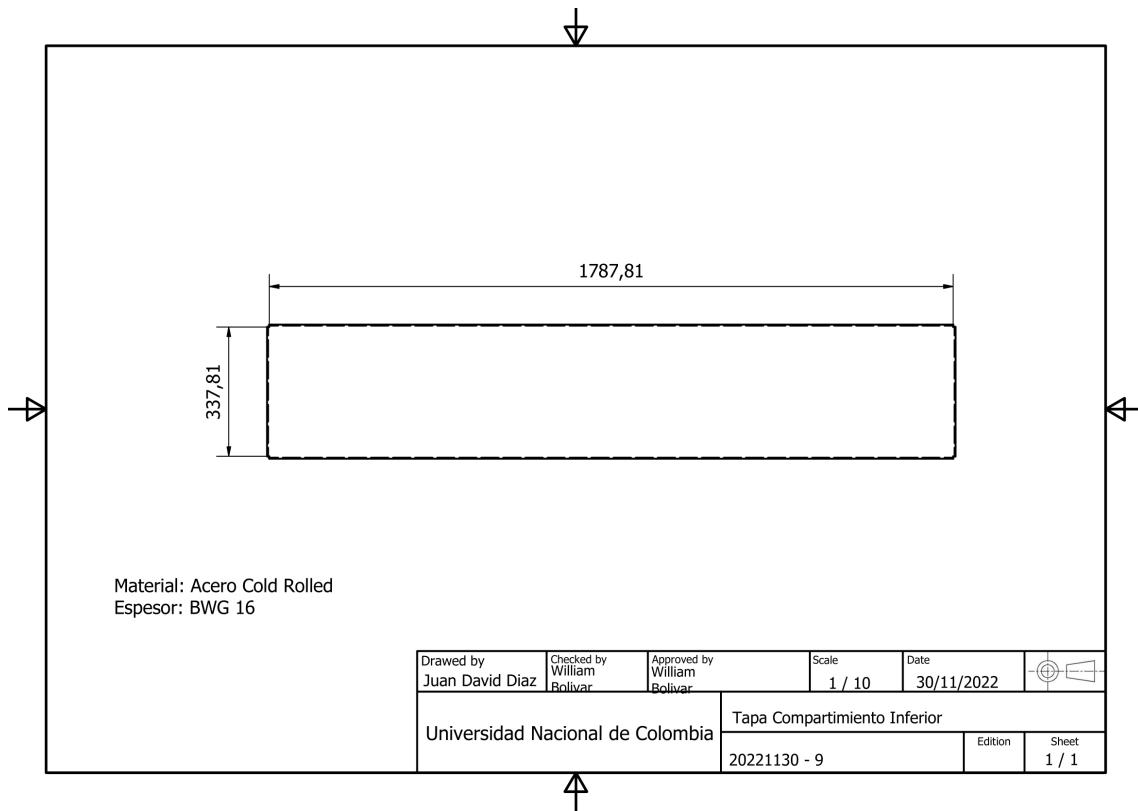


Figura 65: Tapa compartimiento inferior

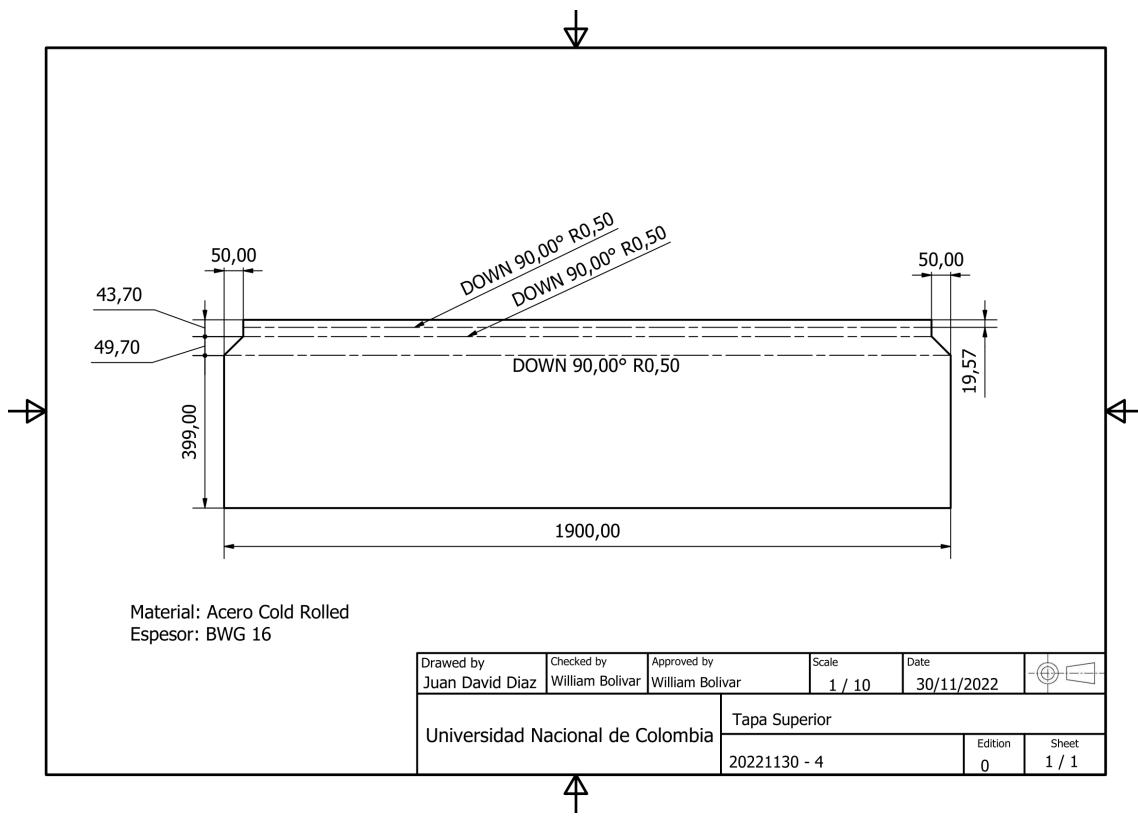


Figura 66: Tapa superior

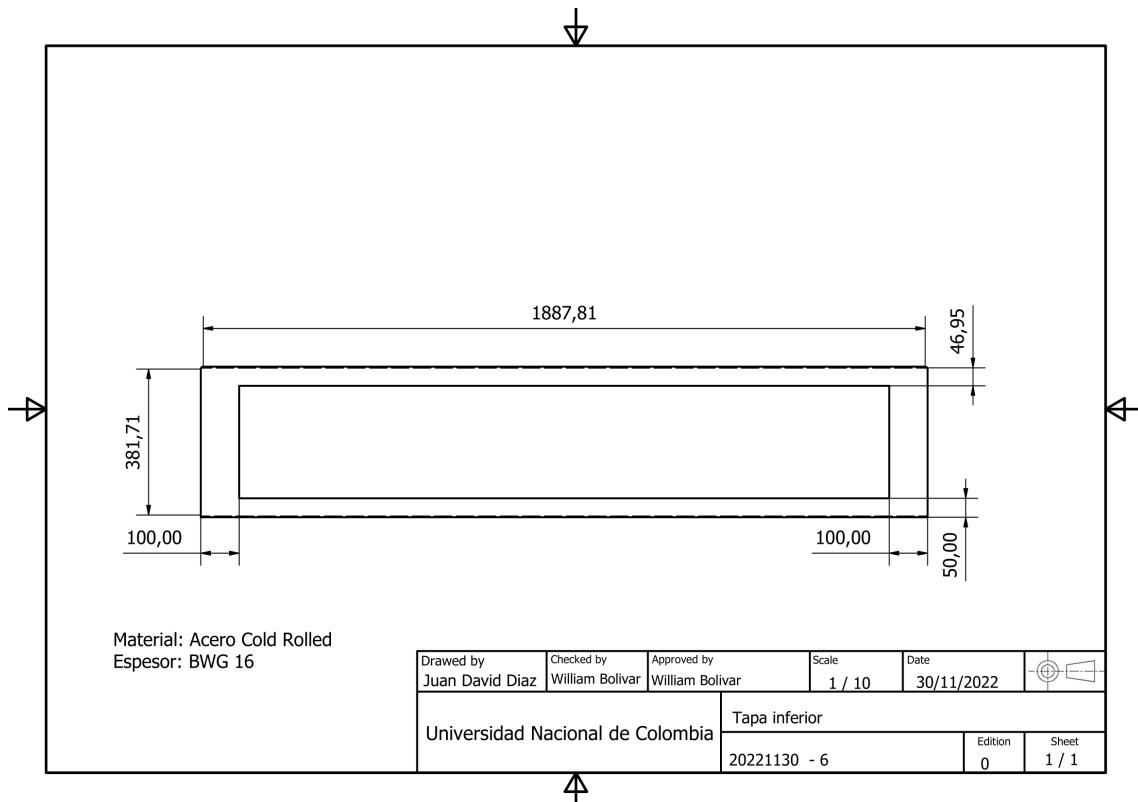


Figura 67: Tapa inferior

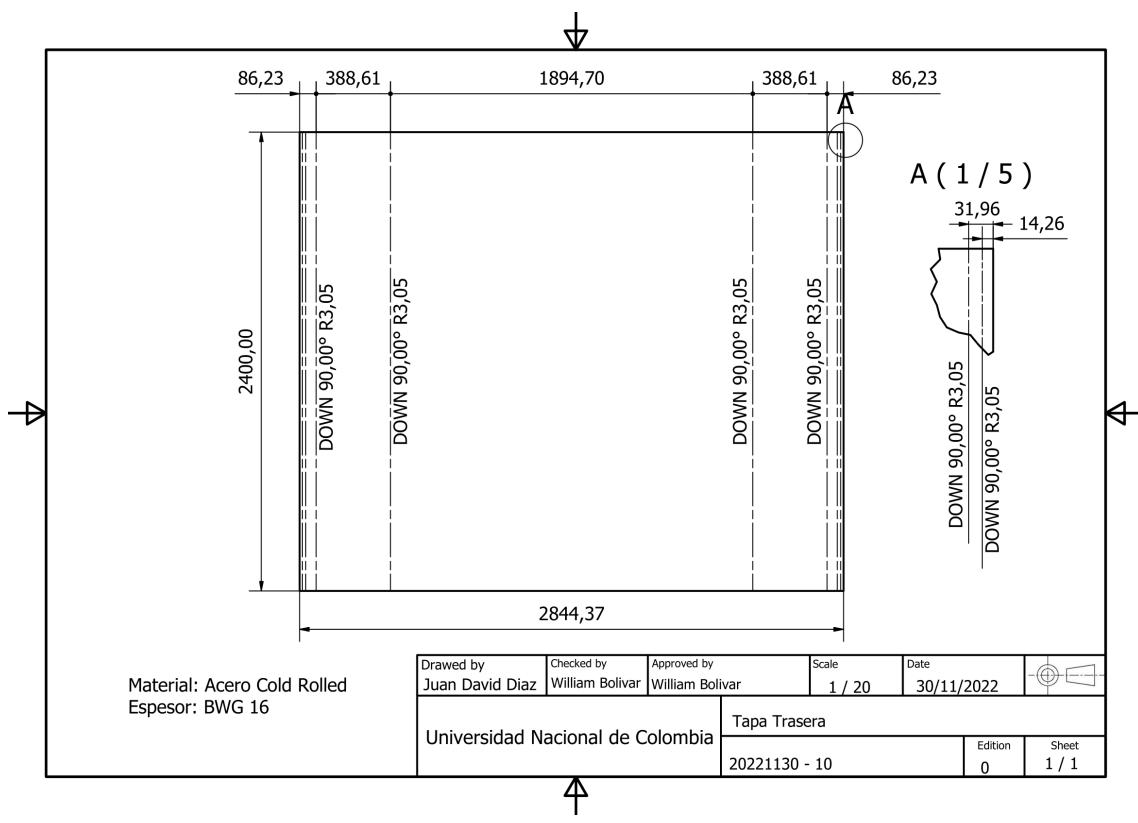


Figura 68: Tapa trasera

3.2.6. Conclusiones y recomendaciones

- Con este trabajo se adquirieron nuevos conocimientos que son de gran ayuda para el ámbito académico y laboral, para automatizar procesos y ahorrar tiempo.
- Con este proyecto se repaso y profundizo en las reglas de dibujo técnico, especialmente en el tema de desarrollo de lamina. Asimismo, se reforzó el uso de software CAD para apoyar la creación de planos de ingeniería.
- Como recomendación sería mejorar la comunicación en proyectos que fue una de las fallas del grupo.
- El uso del lenguaje de VBA permite conectar diversas aplicaciones como excel, inventor, outlook, lo que permitió la relización inicial de esta parte del proyecto de manera automática.
- Se recomienda vincular adecuadamente la programación e ilogic con VBA para evitar errores de compilación. Para ello se recomienda investigar más la documentación de ilogic.
- La principal adversidad presentada en esta parte del proyecto fue la automatización de los planos en inventor lo que retrasó en gran parte la unificación del modelado de tanques con el resto del proyecto.

3.2.7. Contribuciones

- Fredy Alexander Castañeda: Ayuda en la parte de modelado, ayuda en automatizar el proceso, ayuda en tratar de crear el pdf.
- Juan David Diaz: Ayuda en la parte de modelado y creación de planos.

4. Respuesta Automática

Integrantes:

- Andrés Holguín Restrepo

4.1. Explicación general

Para la sección de respuesta automática fue necesario realizar diversas operaciones con el fin de poder converger todos los avances realizados en la página web, los dos proyectos CAD, la base de datos, y la implementación del correo electrónico como entrada y salida de información. Todo esto se implementó en un excel principal donde mediante el accionamiento de un botón, empieza a funcionar el programa ya sea perpetuamente o por una cantidad definida de veces. A continuación se ve una imagen de dicha interfaz:

	A	B	C	D	E	F	G	H	I
1									
2									
3		Datos Correo							
4	Destino	biciklofbt@gmail.com							
5	Destinatario	proyectocomputaciongráfica2022@gmail.com							
6	Asunto	Modelado tableros							
7	Fecha	12/11/2022 19:31							
8		Datos cliente							
9	Nombre	Andrés Holguín Restrepo							
10	Correo electrónico	aholguinr@unal.edu.co							
11	Dirección	Cra 45 #2685							
12	Número de contacto	3166999804							
13									
14	CASO	Tableros							
15									
16									
17									
18									
19									
20									

Figura 69: Interfaz excel principal

Este programa fue ejecutado bajo el computador portátil propio de alto rendimiento, teniendo una memoria RAM de 16GB, un disco SSD de más de 300 GB disponibles, una tarjeta gráfica nvidia 1050 ti y un procesador intel i7 de octava generación. Dicho esto, es de esperarse que los procesos puedan ser realizados sin problemas en esta máquina.

Alejándose un poco de esta definición del equipo para la ejecución del proyecto, es necesario tener una mejor claridad y concepción de todos los pasos involucrados en el proceso interno de la máquina que va a ejecutar el programa. Con base a esta observación, se realizó el diagrama de flujo que muestra el funcionamiento general de todo el proceso del proyecto.

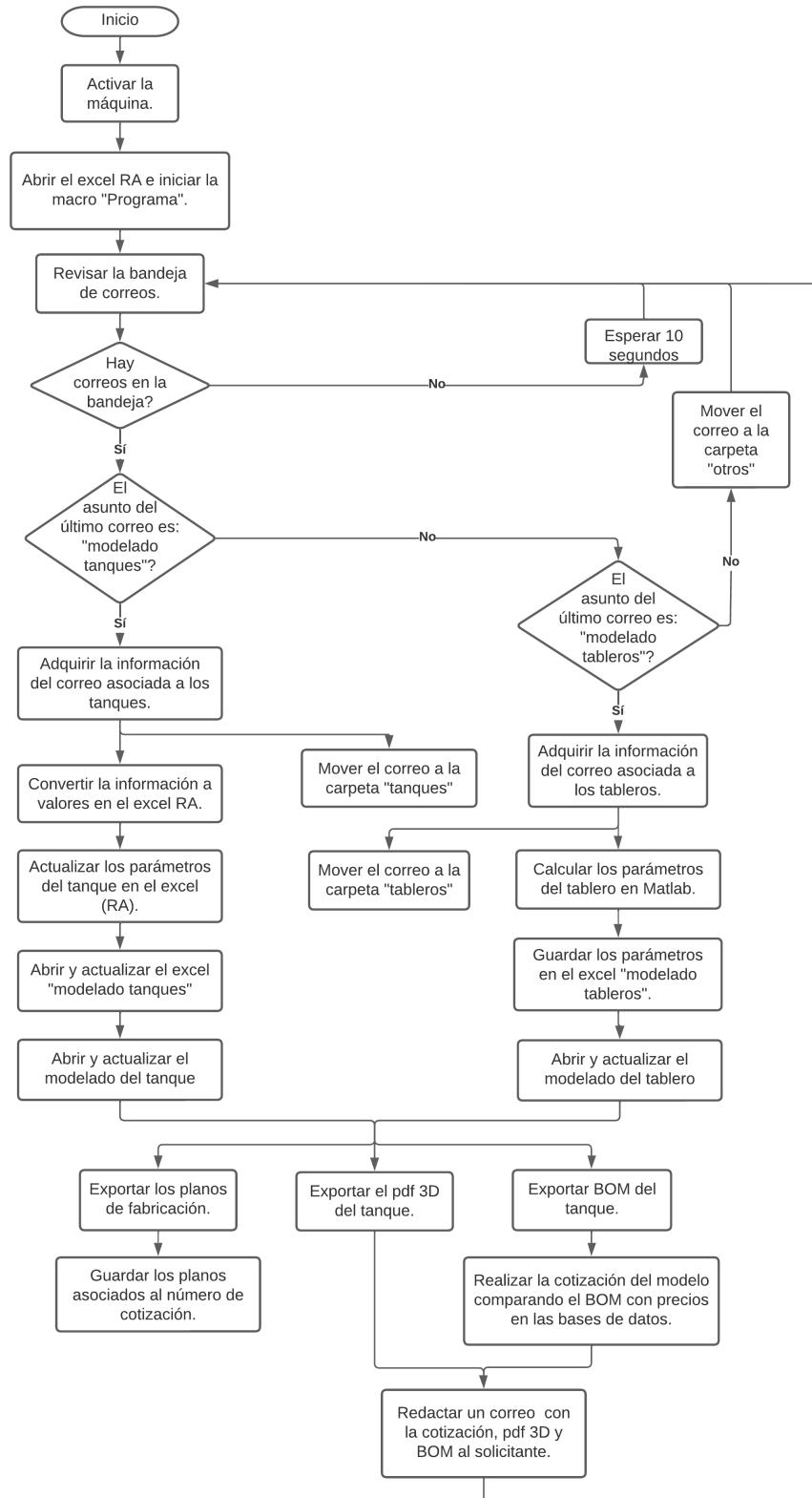


Figura 70: Diagrama de flujo del proyecto

Como puede verse en el diagrama, mediante el uso del correo, en este caso el correo del proyecto: *proyectocomputaciongrafica2022@gmail.com* es posible adquirir los datos que se obtienen de los formularios realizados en la página web. Así, se adquieren los datos como un arreglo de caracteres, con formatos diferentes para los tableros y para los tanques. Es por esto que es necesario ramificar el

proceso de conversión de información dependiendo a cuál de las cotizaciones se realizó.

Al identificar qué proceso debe hacerse, realiza la extracción de variables asociadas a cada modelado. En caso de los tanques, al extraer el galonaje requerido por el cliente, se calculan los parámetros necesarios del tanque como los diámetros externos en un excel diferente al principal, el cual se encuentra conectado a los parámetros de inventario, motivo por el cual al abrir el modelado con variaciones en estos parámetros, las dimensiones del modelo varían acorde a la petición del cliente. Similar con los tableros, aunque en este caso sus parámetros se dan mediante la ejecución de un programa en Matlab que termina escribiendo un nuevo archivo .xlsx que viene conectado con el modelado y así es que se actualizan los parámetros de los tableros.

A partir de estos cambios, se abren los modelados correspondientes, actualizándolos al estar enlazados automáticamente a archivos de Matlab con los parámetros asociados. Debido a una falta de profundización en el modelado, el tablero generaba una advertencia inicial por errores de relaciones entre piezas que tocaba validar manualmente, por lo que este proceso no fue del todo automático, sin embargo al realizar esta operación, sigue el transcurso continuo del programa. Una vez actualizado el modelado, se extrae la vista isométrica, el BOM, y el PDF 3D, el cual es lo más demorado de todo el proceso, ya que puede durar entre 3-5 minutos para el tanque y de 1-2 minutos para el tablero, donde el tanque generó inconvenientes ya que sus altos tiempos de consumo debido al tamaño físico del modelado de tanques a veces generaba advertencias de pausa del programa en excel hasta que no se validara y que no terminara la finalización de la generación del PDF 3D. Ahora bien, ya con estos archivos, se prosigue a guardar cambios y a calcular la cotización mediante modificaciones del BOM. Con un parámetro ya establecido y encontrando el costo final, se procede a enviar el correo al cliente acerca de su consulta y cotización, enviándole estos entregables previamente mencionados.

Por último, se almacenan todos los archivos en sus carpetas correspondientes al número de pedido establecido. empezando por el número 1 en adelante

4.2. Código

Con todo y lo anterior, continuación se muestra el código implementado en la macro principal.

```

1 Sub main()
2     ' Funcion principal donde se ejecuta en bucle o N veces
3     On Error GoTo ErrHandler 'Usar en Error Handler para manejo de errores
4
5     Dim objOutlook As Object ' Set a un objeto de tipo outlook
6     Set objOutlook = CreateObject("Outlook.Application") ' Crear el objeto
7         aplicación Outlook
8
9     Dim objNSpace As Object      ' Crear y asignar un objeto espacio .
10    Set objNSpace = objOutlook.GetNamespace("MAPI")
11
12    Dim myFolder As Object      ' Crear un objeto carpeta .
13    Set myFolder = objNSpace.GetDefaultFolder(olFolderInbox)
14    Dim myDestFolderOtros As Object ' Crear un objeto de carpeta destino .
15    Dim myDestFolderTanques As Object ' Crear un objeto de carpeta destino .
16    Dim myDestFolderTableros As Object ' Crear un objeto de carpeta destino .
17    Set myDestFolderOtros = objNSpace.GetDefaultFolder(olFolderInbox).Folders("Otros") ' Asignar el objeto a la carpeta "otros" del correo
18    Set myDestFolderTanques = objNSpace.GetDefaultFolder(olFolderInbox).Folders("Tanques") ' Asignar el objeto a la carpeta "Tanques" del correo
19    Set myDestFolderTableros = objNSpace.GetDefaultFolder(olFolderInbox).Folders("Tableros") ' Asignar el objeto a la carpeta "Tableros" del correo

```

```
19  Dim objItem As Object ' Crear un objeto tipo item
20
21  ' Crear strings de direcciones locales de los .iam
22  Dim strFileTanques As String
23  Dim strFileTableros As String
24  strFileTanques = "C:\Users\andre\Downloads\Computacion Grafica\main\Tanques\
25    Tanque_Parametrizado\Workspaces\Workspace\Tanque5G.iam"
26  strFileTableros = "C:\Users\andre\Downloads\Computacion Grafica\main\
27    Tableros\Tablero.iam"
28
29  ' Crear strings para almacenar informacion de cliente
30  Dim CorreoCliente As String
31  Dim NombreCliente As String
32
33  ' Usar variables iteradores
34  i = 0 ' usada para numero de ciclos a realizar
35  NumCot = 1 ' usada para tener un ID de cotizacion
36
37  ' Direccion variable donde se almacenan los entregables de cada cotizacion
38  Dim dir As String
39  dir = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\" &
40    NumCot
41
42  ' Crear un FSO para mover objetos de direcciones mas adelante
43  Dim FSO As New FileSystemObject
44  Set FSO = CreateObject("Scripting.FileSystemObject")
45
46  ' Strings de direcciones de archivos que se van a crear
47  Dim strBomOr As String
48  Dim strPdfOr As String
49  Dim strImgOr As String
50  strImgOr = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\
51    Modelado.png"
52  strBomOr = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\
53    BOM.xlsx"
54  strPdfOr = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\
55    Modelo3D.pdf"
56
56  ' Strings de direcciones finales despues de mover los archivos a su carpeta
57  ' correspondiente
58  Dim strBomFi As String
59  Dim strPdfFi As String
60  Dim strImgFi As String
61
62  strImgFi = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\" &
63    NumCot & "\Modelado.png"
64  strBomFi = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\" &
65    NumCot & "\BOM.xlsx"
66  strPdfFi = "C:\Users\andre\Downloads\Computacion Grafica\main\Cotizaciones\" &
67    NumCot & "\Modelo3D.pdf"
68
69  'Do While True 'Siempre va a realizar la rutina de busqueda y extraccion de
70  ' datos
71  Do While i < 1 ' va a realizar la extraccion de correos N veces , 1 vez en
72  ' este caso
73  i = i + 1
74  If myFolder.Items.Count > 0 Then ' Si hay correos en la bandeja de entrada:
```

```

65     Sheets ("DAQ") . Select
66     Range ("B3:B6") . Select
67     Selection . ClearContents
68     Range ("B9:B12") . Select
69     Selection . ClearContents
70     Range ("B40") . Select
71     Selection . ClearContents
72     Range ("H3") . Select
73     Selection . ClearContents
74     Range ("H14:H17") . Select
75     Selection . ClearContents
76     Dim objMail As Outlook . MailItem
77
78     ' Asignar propiedades del ultimo correo de la bandeja a la hoja de
79     ' excel principal "DAQ"
80
81     Set objMail = myFolder . Items . GetLast
82     Range ("B3") = objMail . SenderEmailAddress
83     Range ("B4") = objMail . To
84     Range ("B5") = objMail . Subject
85     Range ("B6") = objMail . ReceivedTime
86
87     ' Si el asunto del correo es "Modelado tanques", empieza toda su
88     ' rutina correspondiente:
89     If objMail . Subject = "Modelado tanques" Then
90         dir = "C:\Users\andre\Downloads\Computacion Grafica\main\
91             Cotizaciones\" & NumCot
92         CrearCarpeta (dir) 'Crea una carpeta donde guardar los datos
93
94         strImgFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
95             Cotizaciones\" & NumCot & "\Modelado.png"
96         strBomFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
97             Cotizaciones\" & NumCot & "\BOM.xlsx"
98         strPdfFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
99             Cotizaciones\" & NumCot & "\Modelo3D.pdf"
100
101     ' Todo el proceso de conversion del body del correo a valores
102     ' individuales de celda
103     Range ("B40") . Value = objMail . Body
104     Range ("B40") . Select
105     Application . CutCopyMode = False
        ActiveSheet . ListObjects . Add (x1SrcRange , Range ("$B$40") , , xlNo) . Name
            =
            "Table115"
        Range ("Table115 [Column1]") . Select
        Application . CutCopyMode = False
        ActiveWorkbook . Queries . Add Name:="Table115" , Formula:=_
            "let" & Chr(13) & "" & Chr(10) & "    Source = Excel .
            CurrentWorkbook () {[Name=""Table115""]}[Content],," & Chr(13) &
            "" & Chr(10) & "#""Changed Type"" = Table .
            TransformColumnTypes (Source ,{{"Column1", type text}}),," &
            Chr(13) & "" & Chr(10) & "#""Split Column by Delimiter"" =
            Table . SplitColumn (#""Changed Type"" , "Column1" , Splitter .
            SplitTextByDelimiter ("#(1f)" , QuoteStyle . Csv) , {"Column1
            .1" , "Column1.2" , "Column1.3" , "Column1.4" , " & -
            ""Column1.5" , "Column1.6" , "Column1.7"}) , & Chr(13) & ""
            & Chr(10) & "#""Changed Type1"" = Table .
            TransformColumnTypes (#""Split Column by Delimiter" ,{""}

```

```

Column1.1"" , Int64.Type} , {""Column1.2"" , type text} , {"""
Column1.3"" , type text} , {""Column1.4"" , type text} , {"""
Column1.5"" , type text} , {""Column1.6"" , type text} , {"""
Column1.7"" , Int64.Type}}) & Chr(13) & "" & Chr(10) & "in" &
Chr(13) & "" & Chr(10) & " #""Changed Type1"""
106 With ActiveSheet.ListObjects.Add(SourceType:=0, Source:=_
107     "OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data Source=$Workbook$;
108     Location=Table115;Extended Properties=""" - 
109     , Destination:=Range("$B$45")).QueryTable
110     .CommandType = xlCmdSql
111     .CommandText = Array("SELECT * FROM [Table115]")
112     .RowNumbers = False
113     .FillAdjacentFormulas = False
114     .PreserveFormatting = True
115     .RefreshOnFileOpen = False
116     .BackgroundQuery = True
117     .RefreshStyle = xlInsertDeleteCells
118     .SavePassword = False
119     .SaveData = True
120     .AdjustColumnWidth = True
121     .RefreshPeriod = 0
122     .PreserveColumnInfo = True
123     .ListObject.DisplayName = "Table115_2"
124     .Refresh BackgroundQuery:=False
125 End With
126     ' Al terminar la conversion de datos , se llevan a las celdas que
127     ' se requieren los datos
128     Range("B9") = Range("D46")
129     Range("B10") = Range("F46")
130     Range("B11") = Range("G46")
131     Range("B12") = Range("H46")
132     Range("H3") = Range("B46")
133     Range("H4") = Range("E46")
134
135     ' Guardar datos del cliente
136     NombreCliente = Range("B9").Text
137     CorreoCliente = Range("B10").Text
138
139     ActiveWorkbook.Questions("Table115").Delete
140     Range("A36:I50").Select
141     Selection.ClearContents
142     Columns("B:B").Select
143     Selection.ColumnWidth = 45
144     Rows("3:12").Select
145     Selection.RowHeight = 12.5
146     Columns("G:G").Select
147     Selection.ColumnWidth = 25
148     Range("C5").Select
149     Range("A1").Select
150     objMail.Move myDestFolderTanques      'Despues de extraer los
151     ' datos del correo , se mueven a la carpeta de Tanques para ser
152     ' almacenada
153     ActiveWorkbook.Save
154     Call CalculosTanque      ' Realiza rutina de valores numericos
155     ' para el tanque
156     Call AbrirTanque(strFileTanques)    ' Realiza todo el manejo de
157     ' inventario para el tanque
158     Dim CostoTanque As String

```

```

153     CostoTanque = GetCostoTanque()      ' Se obtiene el costo final
           del BOM del tanque
154
155     ' Se almacenan los archivos en la carpeta de cotizacion
156     FSO.MoveFile strBomOr, strBomFi
157     FSO.MoveFile strImgOr, strImgFi
158     FSO.MoveFile strPdfOr, strPdfFi
159
160     ' Realiza la rutina de enviar el correo con toda la informacion
           generada
161     Call EnviarCorreo("Cotizacion Tanque", NombreCliente,
           CorreoCliente, CostoTanque, CStr(NumCot), strBomFi, strImgFi,
           strPdfFi)
162
163     NumCot = NumCot + 1 ' Aumenta el numero de cotizacion para el
           proximo caso
164
165
166     ' Si el asunto del correo es "Modelado tanques", empieza toda su rutina
           correspondiente:
167     ElseIf objMail.Subject = "Modelado tableros" Then
168         dir = "C:\Users\andre\Downloads\Computacion Grafica\main\
           Cotizaciones\" & NumCot
169         CrearCarpeta (dir) 'Crea una carpeta donde guardar los datos
170         strImgFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
           Cotizaciones\" & NumCot & "\Modelado.png"
171         strBomFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
           Cotizaciones\" & NumCot & "\BOM.xlsx"
172         strPdfFi = "C:\Users\andre\Downloads\Computacion Grafica\main\
           Cotizaciones\" & NumCot & "\Modelo3D.pdf"
173
174     ' Todo el proceso de conversion del body del correo a valores
           individuales de celda
175     Range("B40").Value = objMail.Body
176     Range("B40").Select
177     Application.CutCopyMode = False
178     ActiveSheet.ListObjects.Add(xlSrcRange, Range("$B$40"), , xlNo).Name
           =
           "Table65"
179     Range("Table65[[#All],[Column1]]").Select
180     Application.CutCopyMode = False
181     ActiveWorkbook.Questions.Add Name:="Table2", Formula:=_
           "let" & Chr(13) & "" & Chr(10) & "    Source = Excel."
182           CurrentWorkbook() {[Name=""Table65""]}[Content], & Chr(13) &
           "" & Chr(10) & "#""Changed Type"" = Table.
183           TransformColumnTypes(Source,{ {"Column1", type text}}), &
           Chr(13) & "" & Chr(10) & "#""Split Column by Delimiter"""
           = Table.SplitColumn(#""Changed Type"" , ""Column1"", Splitter.
           SplitTextByDelimiter("#(1f)" , QuoteStyle.Csv) , {"Column1
           .1" , "Column1.2" , "Column1.3" , "Column1.4" , " & -
           ""Column1.5" , "Column1.6" , "Column1.7" , "Column1.8" , "Column1.9"})
           , & Chr(13) & "" & Chr(10) & "#""Changed
           Type1"" = Table.TransformColumnTypes(#""Split Column by
           Delimiter""" , { {"Column1.1" , Int64.Type} , {"Column1.2" ,
           Int64.Type} , {"Column1.3" , Int64.Type} , {"Column1.4" ,
           Int64.Type} , {"Column1.5" , type text} , {"Column1.6" ,
           type text} , {"Column1.7" , type text} , {"Col" & -

```

```

185     "umn1.8"" , type text} , {""Column1.9"" , Int64.Type})}" & Chr(13)
186     & "" & Chr(10) & "in" & Chr(13) & "" & Chr(10) & "#"
187     Changed Type1"""
188 With ActiveSheet.ListObjects.Add(SourceType:=0, Source:="_
189     "OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data Source=$Workbook$;
190     Location=Table2;Extended Properties="""_
191     , Destination:=Range("$B$45")).QueryTable
192     . CommandType = xlCmdSql
193     . CommandText = Array("SELECT * FROM [Table2]")
194     . RowNumbers = False
195     . FillAdjacentFormulas = False
196     . PreserveFormatting = True
197     . RefreshOnFileOpen = False
198     . BackgroundQuery = True
199     . RefreshStyle = xlInsertDeleteCells
200     . SavePassword = False
201     . SaveData = True
202     . AdjustColumnWidth = True
203     . RefreshPeriod = 0
204     . PreserveColumnInfo = True
205     . ListObject.DisplayName = "Table2"
206     . Refresh BackgroundQuery:=False
207 End With
208 ActiveWorkbook.Questions("Table2").Delete
209 ' Al terminar la conversion de datos , se llevan a las celdas que se
210     requieren los datos
211 Range("B9") = Range("G46")
212 Range("B10") = Range("H46")
213 Range("B11") = Range("I46")
214 Range("B12") = Range("J46")
215 Range("H14") = Range("B46")
216 Range("H15") = Range("C46")
217 Range("H16") = Range("D46")
218 Range("H17") = Range("E46")
219
220 ' Guardar datos del cliente
221 Dim NumBandejas As Integer
222 NumBandejas = Range("H17")
223 NombreCliente = Range("B9").Text
224 CorreoCliente = Range("B10").Text
225
226 Range("A36:J47").Select
227 Selection.ClearContents
228 Columns("B:B").Select
229 Selection.ColumnWidth = 45
230 Range("A1").Select
231
232 objMail.Move myDestFolderTableros
233 ActiveWorkbook.Save
234
235 Call CalculosTablero 'Rutinas de matlab y valores numericos de
236     preparacion para el tablero
237 Call AbrirTablero(strFileTableros, NumBandejas) 'Apertura y
238     actualizacion del tablero
239 Dim CostoTablero As String
240 CostoTablero = GetCostoTablero() ' Se obtiene el costo final del BOM
241     del tanque

```

```
236
237     ' Se almacenan los archivos en la carpeta de cotizacion
238     FSO.MoveFile strBomOr, strBomFi
239     FSO.MoveFile strImgOr, strImgFi
240     FSO.MoveFile strPdfOr, strPdfFi
241
242     ' Realiza la rutina de enviar el correo con toda la informacion
243     ' generada
244     Call EnviarCorreo("Cotizacion Tablero", NombreCliente, CorreoCliente
245             , CostoTablero, CStr(NumCot), strBomFi, strImgFi, strPdfFi)
246     NumCot = NumCot + 1 ' Aumenta el numero de cotizacion para el
247             proximo caso
248
249 Else
250     ' Si tiene otro asunto, se mueve a la carpeta otros
251     objMail.Move myDestFolderOtros
252 End If
253
254 Else
255     ' Si no hay correos en la bandeja, vuelve a revisar en 2 segundos
256     Application.Wait (Now + TimeValue("0:00:2"))
257 End If
258
259 Loop
260
261 ' En caso de que salga del while loop, se liberan los objetos
262 Set objMail = Nothing
263 Set objOutlook = Nothing
264 Set objNSpace = Nothing
265 Set myFolder = Nothing
266
267
268 ErrHandler: ' En caso de que suceda un error, que imprima toda la informacion
269     ' asociada al mismo
270     Debug.Print Err.Description
271     Debug.Print Err.Number
272     Debug.Print Err.HelpContext
273     Debug.Print Err.LastDllError
274     Debug.Print Err.Source
275
276
277 End Sub
278
279 Sub AbrirTanque(strFileTanques As String)
280     'crear una aplicacion inventor
281     Dim InvApp As Inventor.Application
282     Set InvApp = CreateObject("Inventor.Application")
283     InvApp.Visible = True
284     'abrir el archivo .iam de tanques
285     Call InvApp.Documents.Open(strFileTanques, True)
286     'esperar a que la aplicacion termine de abrirse
287     Do While InvApp.Ready = False
288         Application.Wait (Now + TimeValue("0:00:2"))
289     Loop
```

```
290     'llama a las funciones de extraccion de informacion y archivos
291     Call BOMTanque(InvApp)
292     Call Imagen(InvApp)
293     Call PDF(InvApp)
294     'cierra el documento y la app
295     InvApp.Documents.CloseAll
296     InvApp.Quit
297
298 End Sub
299
300 Sub CalculosTanque()
301     'Abre el excel correspondiente al calculo de parametros para el tanque y se
302     'guarda con los nuevos cambios asociados al excel principal
303     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\Tanques\
304         Tanque_Parametrizado\Workspaces\Workspace\Medidas1.xlsx"
305     Workbooks("Medidas1.xlsx").Activate
306     Workbooks("Medidas1.xlsx").Save
307     Workbooks("Medidas1.xlsx").Close
308 End Sub
309
310 Sub CalculosTablero()
311     ' Actualiza el excel enlazado con los valores de entrada , y luego ejecuta el
312     ' programa de matlab que genera el archivo que sera leido en inventor
313     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\Tableros\
314         FormularioWeb.xlsx"
315     Workbooks("FormularioWeb.xlsx").Activate
316     Workbooks("FormularioWeb.xlsx").Save
317     Workbooks("FormularioWeb.xlsx").Close
318     fileToRun = "C:\Users\andre\Downloads\Computacion Grafica\main\Tableros\
319         Parametros_Tableros.m"
320     matlabCommand = "matlab -nodisplay -nosplash -nodesktop -r "" run('' &
321         fileToRun & ''); exit;""
322     Shell (matlabCommand)
323
324 End Sub
325
326 Sub AbrirTablero(strFileTableros As String , NumBandejas As Integer)
327     'crear una aplicacion inventor
328     Dim InvApp As Inventor.Application
329     Set InvApp = CreateObject("Inventor.Application")
330     InvApp.Visible = True
331     'abrir el archivo .iam de tableros
332     Call InvApp.Documents.Open(strFileTableros , True)
333     'esperar a que la aplicacion termine de abrirse
334     Do While InvApp.Ready = False
335         Application.Wait (Now + TimeValue("0:00:2"))
336     Loop
337
338     Call BOMTablero(InvApp , NumBandejas)
339     Call Imagen(InvApp)
340     Call PDF(InvApp)
341     InvApp.Documents.CloseAll
342     InvApp.Quit
343
344 End Sub
345
346 Sub PlanosTanque()
347     Dim InvApp As Inventor.Application
348     Set InvApp = CreateObject("Inventor.Application") 'crear una aplicacion
349         inventor
```

```

341 InvApp.Visible = True
342 Call InvApp.Documents.Open("C:\Users\andre\Downloads\Computacion Grafica\
343     main\Tanques\Tanque_Parametrizado\Workspaces\Workspace\Planostanque.dwg",
344     True)
345     ' esperar a que la aplicacion termine de abrirse
346 Do While InvApp.Ready = False
347     Application.Wait (Now + TimeValue("0:00:2"))
348 Loop
349     ' Guardar el archivo automaticamente y luego cerrarse
350 InvApp.ActiveDocument.Activate
351 InvApp.ActiveDocument.Save
352 InvApp.ActiveDocument.Save2
353 Do While InvApp.Ready = False
354     Application.Wait (Now + TimeValue("0:00:2"))
355 Loop
356 InvApp.ActiveDocument.Close (False)
357 InvApp.Quit
358
359
360
361 End Sub
362
363
364 Sub PDF(InvApp As Inventor.Application)
365     'Codigo de exportacion del PDF 3D
366 Dim oPDFAddIn As ApplicationAddIn
367 Dim oAddin As ApplicationAddIn
368 For Each oAddin In InvApp.ApplicationAddIns
369     If oAddin.ClassIdString = "{3EE52B28-D6E0-4EA4-8AA6-C2A266DEBB88}" Then
370         Set oPDFAddIn = oAddin
371         Exit For
372     End If
373 Next
374 If oPDFAddIn Is Nothing Then
375     MsgBox "Inventor 3D PDF Addin not loaded."
376     Exit Sub
377 End If
378 Dim oPDFConvertor3D
379 Set oPDFConvertor3D = oPDFAddIn.Automation
380 Dim oDocument As Document
381 Set oDocument = InvApp.ActiveDocument
382 Dim oOptions As NameValueMap
383 Set oOptions = InvApp.TransientObjects.CreateNameValueMap
384 oOptions.Value("FileOutputLocation") = ("C:\Users\andre\Downloads\
385     Computacion Grafica\main\Cotizaciones\Modelo3D.pdf")
386 oOptions.Value("ExportAnnotations") = 0
387 oOptions.Value("ExportWokFeatures") = 0
388 oOptions.Value("GenerateAndAttachSTEPFile") = False
389 oOptions.Value("LimitToEntitiesInDVRs") = True
390 oOptions.Value("VisualizationQuality") = AccuracyEnum.kVeryHigh
391 Dim sProps(0) As String
392 sProps(0) = "{F29F85E0-4FF9-1068-AB91-08002B27B3D9}:Title" ' Title
393 oOptions.Value("ExportAllProperties") = False
394 oOptions.Value("ExportProperties") = sProps
395 Dim sDesignViews(0) As String

```

```

395     sDesignViews(0) = oDocument.ComponentDefinition.RepresentationsManager.
396         ActiveDesignViewRepresentation.Name
397     oOptions.Value("ExportDesignViewRepresentations") = sDesignViews
398     Call oPDFConvertor3D.Publish(oDocument, oOptions)
399 End Sub
400
400 Sub BOMTanque(InvApp As Inventor.Application)
401     'Codigo de exportacion del BOM
402     Dim oDoc As AssemblyDocument
403     Set oDoc = InvApp.ActiveDocument
404     Dim oBOM As BOM
405     Set oBOM = oDoc.ComponentDefinition.BOM
406     oBOM.StructuredViewFirstLevelOnly = False
407     oBOM.StructuredViewEnabled = True
408     Dim oStructuredBOMView As BOMView
409     Set oStructuredBOMView = oBOM.BOMViews.Item("Structured")
410
411     ' Genera el archivo BOM en excel
412     oStructuredBOMView.Export "C:\Users\andre\Downloads\Computacion Grafica\main\
413         \Cotizaciones\BOM.xlsx", kMicrosoftExcelFormat
414
415     ' Cotizar sobre este archivo
416     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\
417         \Cotizaciones\BOM.xlsx"
418     Workbooks("BOM.xlsx").Activate
419     ' suma las columnas de costos por su cantidad de unidades
420     ' Costo de mano de obra del 25% del total de material
421     Range("L1").Select
422     ActiveCell.FormulaR1C1 = "Total"
423     Range("L2").Select
424     Application.CutCopyMode = False
425     ActiveCell.FormulaR1C1 = "=+RC[-5]*RC[-4]"
426     Range("L2").Select
427     Selection.AutoFill Destination:=Range("L2:L5"), Type:=xlFillDefault
428     Range("L2:L5").Select
429     Range("K10").Select
430     ActiveCell.FormulaR1C1 = "Materiales"
431     Range("K11").Select
432     ActiveCell.FormulaR1C1 = "Mano de obra"
433     Range("H1").Select
434     ActiveCell.FormulaR1C1 = "Costo area Superficial"
435     Range("H2:H5").Value = 5.9583333333333E-03
436     Range("L10").Select
437     ActiveCell.FormulaR1C1 = "=SUM(R[-8]C:R[-1]C)"
438     Range("L11").Select
439     ActiveCell.FormulaR1C1 = "=R[-1]C*0.25"
440     Range("L12").Select
441     Application.CutCopyMode = False
442     ActiveCell.FormulaR1C1 = "=R[-2]C+R[-1]C"
443     Range("L10:L12").Select
444     Selection.Style = "Currency"
445
446     Workbooks("BOM.xlsx").Save
447     Workbooks("BOM.xlsx").Close
448 End Sub

```

```
450
451
452 Sub BOMTablero(InvApp As Inventor.Application, NumBandejas As Integer)
453     ' Macro de exportacion del BOM del tablero
454     Dim oDoc As AssemblyDocument
455     Set oDoc = InvApp.ActiveDocument
456     Dim oBOM As BOM
457     Set oBOM = oDoc.ComponentDefinition.BOM
458     oBOM.StructuredViewFirstLevelOnly = False
459     oBOM.StructuredViewEnabled = True
460     Dim oStructuredBOMView As BOMView
461     Set oStructuredBOMView = oBOM.BOMViews.Item("Structured")
462     ' Exporta el bom en un excel en una direccion especifica
463     oStructuredBOMView.Export "C:\Users\andre\Downloads\Computacion Grafica\main\
464         \Cotizaciones\BOM.xlsx", kMicrosoftExcelFormat
465
466     ' Cotizar el precio de tablero
467     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\
468         \Cotizaciones\BOM.xlsx"
469     Workbooks("BOM.xlsx").Activate
470     Cells.Replace What:=".", Replacement:="", LookAt:=xlPart, SearchOrder:=_
471         xlByRows, MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False, _
472         FormulaVersion:=xlReplaceFormula2
473     Cells.Replace What:=".00", Replacement:="", LookAt:=xlPart, SearchOrder:=_
474         xlByRows, MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False, _
475         FormulaVersion:=xlReplaceFormula2
476     Cells.Replace What:="$", Replacement:="", LookAt:=xlPart, SearchOrder:=_
477         xlByRows, MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False, _
478         FormulaVersion:=xlReplaceFormula2
479     Range("F5").Select
480     ActiveCell.Value = NumBandejas
481     Range("G12").Select
482     ActiveCell.FormulaR1C1 = "=SUMPRODUCT(R[-10]C[-1]:R[-1]C[-1],R[-10]C:R[-1]C)"
483     Range("F12").Select
484     ActiveCell.FormulaR1C1 = "Materiales:"
485     Range("F13").Select
486     ActiveCell.FormulaR1C1 = "Mano de obra"
487     Range("F14").Select
488     ActiveWindow.SmallScroll ToRight:=1
489     ActiveWindow.SmallScroll Down:=-1
490     Range("G13").Select
491     ActiveCell.FormulaR1C1 = "=R[-1]C*0.25"
492     Range("G14").Select
493     ActiveWindow.SmallScroll Down:=0
494     Range("F14").Select
495     ActiveCell.FormulaR1C1 = "Total:"
496     Range("F15").Select
497     ActiveCell.FormulaR1C1 = ""
498     Range("G14").Select
499     Application.CutCopyMode = False
500     ActiveCell.FormulaR1C1 = "=R[-2]C+R[-1]C"
501
502     Range("G12:G14").Select
503     Selection.NumberFormat = "$ #,##0.00"
504     Workbooks("BOM.xlsx").Save
505     Workbooks("BOM.xlsx").Close
```

```
505 End Sub
506
507 Function GetCostoTablero() As String
508     ' Abre el archivo BOM de tablero y devuelve el valor del costo total en
509     ' formato String
510     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\
511         Cotizaciones\BOM.xlsx"
512     Workbooks("BOM.xlsx").Activate
513     GetCostoTablero = Range("G14").Text
514     Workbooks("BOM.xlsx").Close
515 End Function
516
517 Function GetCostoTanque() As String
518     ' Abre el archivo BOM de tanque y devuelve el valor del costo total en
519     ' formato String
520     Workbooks.Open "C:\Users\andre\Downloads\Computacion Grafica\main\
521         Cotizaciones\BOM.xlsx"
522     Workbooks("BOM.xlsx").Activate
523     GetCostoTanque = Range("L12").Text
524     Workbooks("BOM.xlsx").Close
525 End Function
526
527 Sub Imagen(InvApp As Inventor.Application)
528     ' Exporta una imagen isometrica dentro del Inventor
529     Dim v As View
530     Set v = InvApp.ActiveView
531     Call v.SaveAsBitmap("C:\Users\andre\Downloads\Computacion Grafica\main\
532         Cotizaciones\Modelado.png", 1024, 768)
533 End Sub
534
535
536 Sub EnviarCorreo(asunto As String, nombre As String, correo As String, costo As
537     String, numOrden As String, dirBOM As String, dirIMG As String, dirPDF As
538     String)
539     ' Codigo para la generacion de un correo saludando a la persona, mostrandole
540     ' el costo total, y entregandole el BOM, PDF 3D y una vista isometrica del
541     ' modelo.
542     Dim EmailApp As Outlook.Application
543     Dim Source As String
544     Set EmailApp = New Outlook.Application
545     Dim EmailItem As Outlook.MailItem
546     Set EmailItem = EmailApp.CreateItem(olMailItem)
547     EmailItem.To = correo
548     EmailItem.Subject = asunto
549     Debug.Print costo
550     EmailItem.HTMLBody = "Saludos " & nombre & "." & "<br>" & "<br>" & "Ya se
551     completo el proceso de cotizacion de su pedido." & -
552     "<br>" & -
553     "Numero de orden: " & numOrden & -
554     "<br>" & -
555     "<br>" & -
556     "Al correo se anexa el BOM, La imagen del modelado, y el PDF 3D. " & -
```

```
553 "El costo total del producto es de: " & costo & "COP" &_
554 "<br>" &_
555 "<br>" &_
556 "3D INNOVATION"
557
558 EmailItem.Attachments.Add dirBOM
559 EmailItem.Attachments.Add dirIMG
560 EmailItem.Attachments.Add dirPDF
561
562 EmailItem.Send
563
564 End Sub
```

4.3. Conclusiones

- El lenguaje VBA, a pesar de ser viejo y con una documentación no tan robusta, es bastante útil para todo el enlace de aplicaciones de manera efectiva ya que todos los programas que se usaron tenían módulo VBA.
- El proyecto sirvió bastante para promover el pensamiento colectivo y las acciones de un todo, ya que el resultado final dependió del aporte de muchas secciones separadas que convergían en la parte de respuesta automática.
- Un error fue que se dejó al grupo muy independiente en gran parte del semestre, lo cual implicó retrasos bastante significativos en los avances y que por ende se generó unas limitantes bastante marcadas en los entregables durante la sustentación.
- En términos generales, el curso logra dar a la generación de un proyecto interdisciplinario de alto nivel que involucraba conceptos extraídos de casi todas las ramas, donde se obtuvieron unos resultados bastante limitados, que sin embargo tienen la posibilidad de mejorarse y expandirse al punto de formar un modelo de negocio bastante interesante teniendo en cuenta los procesos de automatización que se aprendieron a lo largo de todo el curso.

4.4. Referencias

- [1] Autodesk Inventor 2022, “InventorVBAMembers Object”, 2021, disponible en: <https://help.autodesk.com/view/INVNTOR/2022/ENU/?guid=InventorVBAMembers>
- [2] Excel & VBA, ”MANDAR UN EMAIL CON VBA”, Disponible en: <https://exceleyvba.com/enviar-email-vba/>
- [3] Microsoft Referencia de VBA para Office”, 2022, disponible en <https://learn.microsoft.com/es-es/office/vba/api/overview/>
- [4] tutorialspoint, ”VBA Tutorial”, disponible en: <https://www.tutorialspoint.com/vba/index.htm#>