ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Ciencias de la Tierra

“DISEÑO PARAMÉTRICO DE LA ESTRUCTURA DE UNA NAVE INDUSTRIAL”

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero Civil

Presentado por:

Caicedo Cárdenas Kevin Rafael

GUAYAQUIL - ECUADOR

Año: 2021

DEDICATORIA

Dedico el presente proyecto a mis padres Rafael y Elena y a mis hermanos.

AGRADECIMIENTOS

A la Escuela Superior Politécnica del Litoral y todos los docentes de la Facultad de Ingeniería en Ciencias de la Tierra.

Agradezco a mis padres por su constante apoyo durante mis estudios.

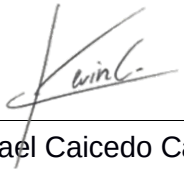
Mi más sincero agradecimiento a mi hermano Jeyko por despejar mis dudas sobre programación.

Un agradecimiento a la empresa Microsoft por distribuir de forma gratuita su entorno de desarrollo integrado Microsoft Visual Studio 2019 el cual facilitó el desarrollo del proyecto.

También agradezco a la empresa Tekla por proveer licencias estudiantiles para su programa Tekla Structures.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; *Kevin Rafael Caicedo Cárdenas* y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Kevin Rafael Caicedo Cárdenas

EVALUADORES

Firmado electrónicamente por:
MIGUEL ANGEL CHAVEZ MONCAYO

Ph.D. Miguel Ángel Chávez Moncayo

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
CARLOS PAUL QUISHPE OTACOMA

M.Sc. Carlos Paul Quispe Otacoma

PROFESOR TUTOR

RESUMEN

En base a la necesidad de garantizar la seguridad estructural de las construcciones informales en Ecuador, el objetivo de la presente Tesis es desarrollar una tecnología que permita a los maestros de obra construir galpones seguros mediante la automatización del proceso de diseño. Se espera reducir drásticamente el tiempo que toma diseñar la estructura de una nave industrial. Para lograrlo se ha utilizado la Interfaz de Programación de Aplicaciones (API) de los programas Sap2000 y Tekla Structures. La primera se utilizó para automatizar el proceso iterativo del diseño estructural. El diseño se realizó conforme a la NEC 2015 y las normas ACI 318, AISC 360-10, AISI LRFD 96 y AWS D1.1. También se programó la modelación automática y generación de planos de la estructura en Tekla Structures con el uso de su API. La aplicación desarrollada redujo el tiempo del diseño estructural a menos de 20 minutos. El programa automatizó con éxito la generación de planos y del presupuesto referencial del proyecto con su respectivo análisis de precios unitarios. Los resultados obtenidos mediante la aplicación son aproximados. A pesar de que el usuario objetivo del programa es un maestro de obra, siempre es mejor que un profesional revise el diseño. Dicho esto, los ingenieros civiles pueden ver la aplicación como una herramienta que genera un prediseño aproximado de una nave industrial.

Palabras Clave: nave industrial, automatización, API, Sap2000, Tekla Structures

ABSTRACT

Based on the need to guarantee the structural safety of informal constructions in Ecuador, the objective of this thesis is to develop a technology that allows construction foremen to build safe sheds by automating the design process. It is expected to drastically reduce the time it takes to design the structure of an industrial building. To achieve this goal, it has been used the Application Programing Interface of the programs Sap2000 and Tekla Structures. The first was used to automate the iterative process of structural design. The design was made in accordance with the NEC 2015 and the ACI 318, AISC 360-10, AISI LRFD 96 and AWS D1.1 standards. It was also programmed the automatic modelling and generation of drawings in Tekla Structures using its API. The developed application reduced the structural design time to less than 20 minutes. The program successfully automated the generation of plans and the project's referential budget with their respective unit price analysis. The results obtained through the application are approximate. Even though the target user of the program is a construction foreman, it is always better to have a professional review the design. That said, civil engineers can view the application as a tool that generates an approximate predesign of an industrial shed.

Keywords: *industrial shed, automation, API, Sap2000, Tekla Structures*

ÍNDICE GENERAL

ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
ASTM	American Society for Testing and Materials
ASCE	American Society of Civil Engineers
AWS	American Welding Society
AISC	American Institute of Steel Construction
AISI	American Iron and Steel Institute
API	Application Programming Interface
NEC	Norma Ecuatoriana de la Construcción
IDE	Entorno de Desarrollo Integrado

SIMBOLOGÍA

mm	Milimetro
mg	Miligramo
m	Metro
kg	Kilogramo
N	Newton
t	Tonelada
l	Litro
lb	Libra
psi	Libra por pulgada cuadrada
hr	Hora
m2	Metro cuadrado
m3	Metro cúbico
s	Segundo
cm	centímetro
km	kilómetro

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE PLANOS

CAPÍTULO 1

1. INTRODUCCIÓN

En el año 2019, la Senplades estimó que en el Ecuador existía una demanda de 61.000 industrias (Diario Expreso, 2019). Por la pandemia de Covid-19, se ha incrementado el comercio electrónico y a su vez la necesidad de espacios de almacenamientos (Coba, 2021).

Tras el comienzo de la Revolución Industrial, surgió la necesidad de trabajar en grandes espacios productivos debido al tamaño de la maquinaria que complementaba la mano de obra. Aunque las primeras naves industriales no satisfacían por completo estas necesidades, el avance de la siderurgia permitió utilizar el acero como material estructural y así construir naves industriales con mayor espacio.

Actualmente, gran número de naves industriales son construidas en parques industriales. Estos parques constituyen una zona favorable para realizar sus procesos laborales debido a que se facilita el transporte, obtención de mano de obra y servicios públicos.

En el Ecuador existe una gran demanda de naves industriales, ya sea destinado a un fin industrial o comercial. Según Germán Carvajal, gerente de división inmobiliaria de la empresa MarketWatch, las bodegas corporativas se han convertido en un tipo de inmueble cada vez más atractivo en el sector industrial de Ecuador.

Antes del avance de la tecnología, diseñar estructuras suponía realizar gran cantidad de cálculos y planos manualmente. En la actualidad, la ingeniería civil posee una gran variedad de softwares que ayudan a los ingenieros civiles en las distintas fases de un proyecto como diseñar, redactar, documentar, construir, etc.

En la presente tesis se profundizará en el uso de dos softwares mediante la ingeniería computacional. El primero es Sap2000, programa desarrollado por la empresa Computers and Structures, Inc. utilizado para analizar y diseñar estructuras. Y el segundo programa es Tekla Structures, desarrollado por la empresa TEKLA, que sirve para modelar estructuras, realizar planos de despiece y de fabricación, y generar una lista de materiales y piezas.

Ambos programas poseen una herramienta que le permite a un desarrollador de aplicaciones utilizar estos programas desde una aplicación externa o incluso

añadir funciones en el programa. Esta herramienta es la Interfaz de Programación de Aplicaciones o API.

Con el trabajo a realizar, se mostrará las ventajas que ofrece la API de Sap2000 y Tekla Structures a los ingenieros civiles utilizando el lenguaje de programación C# y el editor de código Microsoft Visual Studio.

1.1 Antecedentes

La informalidad en la construcción representa un gran riesgo frente a sismos, lo cual es preocupante en Ecuador donde la construcción informal llega casi al 70%. Estas construcciones no pasan por la revisión de un profesional y son construidas por maestros de obra sin experiencia alguna en diseño estructural.

Si se toma en cuenta este hecho, que la demanda de industrias en el país para el año 2019 alcanzaban las 60.000 unidades y, que para el 2021 se ha incrementado por auge del comercio electrónico, es necesario mejorar la seguridad de estas construcciones informales.

Aunque se pueden fortalecer las estructuras luego de ser construidas, es preferible que la construcción informal corresponda desde un inicio con un buen diseño estructural. Sin embargo, el tiempo que toma diseñar cualquier tipo de estructura es largo, esto incluye la duración del proceso de cálculo, detallamiento, elaboración de presupuesto, etc. Incluso con práctica, el diseño completo de un galpón puede tomarle semanas a un profesional.

Por estos motivos se necesita facilitarles a los maestros de obras una herramienta que les ayude a construir estructuras seguras, pero a su vez que reduzca el tiempo necesario para realizar el diseño. Esta herramienta se puede desarrollar mediante la ingeniería computacional.

Mediante la programación se puede automatizar estos procesos de diseño, reduciendo significativamente el tiempo de trabajo. Canchari Edmundo, 2009 desarrolló un programa de análisis de sólidos con simetría axial con el método de los elementos finitos, utilizando la interfaz de programación de aplicaciones de Sap2000. (Canchari, 2009)

En el presente trabajo, se realizará un proceso similar respecto al uso de la API de los programas SAP2000 y Tekla Structures. Sin embargo, el objetivo de la

aplicación es de diseñar la estructura de una nave industrial con ciertos requerimientos.

Se desea una aplicación de escritorio que permita al usuario introducir las medidas necesarias para definir la geometría de la nave industrial, como altura de las columnas, diferencia entre la altura máxima de la estructura y la columna, ancho de columna (criterio arquitectónico), longitud y profundidad de la nave industrial. También podrá introducir las cargas muertas, vivas, y de sismo que se aplicarán a la estructura. En el caso de las cargas de sismo. el usuario introducirá los coeficientes C y k que definen un sismo estático. Además de realizar el diseño, el programa generará los planos estructurales.

1.2 Información básica

La aplicación deberá diseñar una nave industrial utilizando los perfiles comerciales locales. Dicho diseño será realizado con la documentación de la API del software Sap2000. Esta documentación es una biblioteca de funciones, de las cuales se detallarán, en el marco teórico, las utilizadas en este proyecto. De igual manera se realizará para la API del software Tekla Structures.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar una aplicación de escritorio que diseñe la estructura de una nave industrial con dimensiones paramétricas mediante el uso de la Interfaz de Programación de Aplicaciones de los softwares Sap2000 y Tekla Structures.

1.3.2 Objetivos Específicos

1. Diseñar una interfaz atractiva mediante el editor de código Visual Studio para la facilitación de manejo del usuario.
2. Automatizar el diseño estructural de una nave industrial con el uso de la API del programa Sap2000.
3. Generar planos estructurales de forma automática mediante la API del programa Tekla Structures.

1.4 Justificación

Debido al incremento en la demanda de lugares de almacenamiento en Ecuador, es necesario facilitar el diseño de nave industriales de pequeño a mediano tamaño sin necesidad de contratar a un profesional con un alto grado de experticia.

Además, Sergio Torassa, profesor del IDE y socio de Diagnóstico & Soluciones, consultora en reestructuraciones empresariales, con sede en Barcelona, España, dice a Diario Expreso que los parques industriales atraen fuertes inversiones y estimulan el desarrollo económico de la zona en la que se encuentran instalados (Diario Expreso, 2019).

CAPÍTULO 2

2. DESARROLLO DEL PROYECTO

2.1 Metodología

Primero se investigó las funciones de la documentación API de los softwares Sap2000 y Tekla Structures necesarias para la realización del proyecto. Se desarrolló la interfaz gráfica utilizando la tecnología Windows Presentation Foundation (WPF). El objetivo de esta interfaz es de trabajar como programa principal que permite controlar los demás programas mediante código. Esta es la clave del diseño automático.

Se desarrolló un procedimiento de prediseño válido para galpones comunes semejantes a la alternativa de solución seleccionada. Para previsualizar la geometría final del galpón, se creó un apartado dentro de la aplicación que calcula la geometría completa de la estructura, la muestra en un espacio 3D y permite visualizar diferentes vistas.

Se codificó el ingreso de datos al Sap2000 que definen los materiales, secciones, elementos tipo frame, cargas, etc. Con el modelo de prediseño generado por código, se verificó manualmente el cálculo de las secciones de prediseño. Además, se desarrolló el algoritmo que genera todos los elementos de la estructura utilizando las secciones calculadas anteriormente.

Luego se añadió el proceso automático de diseño, en el cual el programa le ordena al SAP2000 que realice el chequeo de la estructura. Sin embargo, esto por sí solo no diseña el galpón. Para esto se desarrolló un algoritmo iterativo que calcula las secciones óptimas que cumplan con las solicitaciones.

Finalmente, se programó la recuperación de la geometría y resultados del análisis estructural, para automatizar el diseño de la cimentación. Con el diseño terminado, se automatizó la modelación de la estructura y la elaboración de los planos en Tekla Structures.

2.2 Marco teórico

2.2.1 Acero estructural

El acero presenta un conjunto de beneficios al momento de usarse en estructuras como: gran capacidad de resistir esfuerzos, proceso constructivo relativamente fácil, capacidad de cubrir luces largas manteniendo un peso ligero en la estructura (Córdova Reyes, 2014).

2.2.2 Acero laminado

Este acero proviene de un proceso de laminado, en el cual se deforman los lingotes de acero fundido a altas temperaturas alargándolos cada vez a través de un tren de laminación. Los aceros laminados poseen gran capacidad de elasticidad y dilatación, además de presentar valores isotrópicos en sus propiedades debido a la homogeneidad del material.

2.2.3 Nave industrial

Son edificios de uso industrial o comercial, caracterizados por capacidad de almacenamiento. Su origen se remonta a la Revolución Industrial cuando se asentó la producción en serie, y la mano de obra asistida por maquinaria requería espacios grandes. Pueden ser construidos en hormigón, madera o acero, siendo este último material el más usado.

2.2.4 Tipos de nave industrial según su geometría

Existen diferentes tipos de galpones que presentan ventajas y desventajas entre sí. Pero una característica principal que los diferencia es la geometría de su cubierta. De acuerdo con este criterio, existen los galpones curvos, a un agua, a dos aguas, y a diente sierra.

2.2.5 Galpón curvo

Figura 2.1 Galpón curvo

Fuente: Padilla & Portugal, 2013

Este tipo de galpón con forma curva es ideal para instalaciones comerciales, deportivas, auditorios, etc. La carga de viento sobre la estructura se ve reducida. El diseño de la nave industrial se puede realizar con estructura tubular o reticular (Padilla & Portugal, 2013).

2.2.6 Galpón a un agua

Este tipo de galpón presenta solo una pendiente, lo que lo hace ideal para zonas con gran intensidad de lluvia o granizo. Su diseño puede realizarse en estructura de alma llena, tubular o reticulado (Padilla & Portugal, 2013).

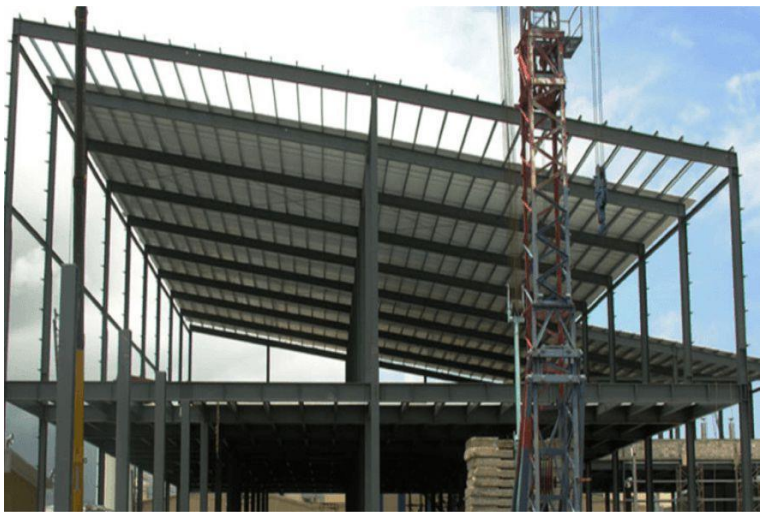


Figura 2.2 Galpón a un agua

Fuente: Padilla & Portugal, 2013

2.2.7 Galpón a dos aguas con viga de amarre

El galpón a dos aguas en cambio presenta dos pendientes, y también puede ser diseñado en estructura de alma llena, tubular o reticulado. Sin embargo, tanto los sistemas a un agua como a dos aguas se pueden rigidizar de igual manera. La figura 2.3 muestra un galpón rigidizado mediante una viga de amarre.

Figura 2.3 Galpón a dos aguas con viga de amarre

Fuente: Padilla & Portugal, 2013

2.2.8 Galpón a dos aguas con tensores laterales en X

Otra forma de rigidizar los galpones es con tensores laterales en X, también llamadas crucetas. Sin embargo, no suelen ser muy usadas en galpones que requieren permitir la movilidad a través de los laterales.

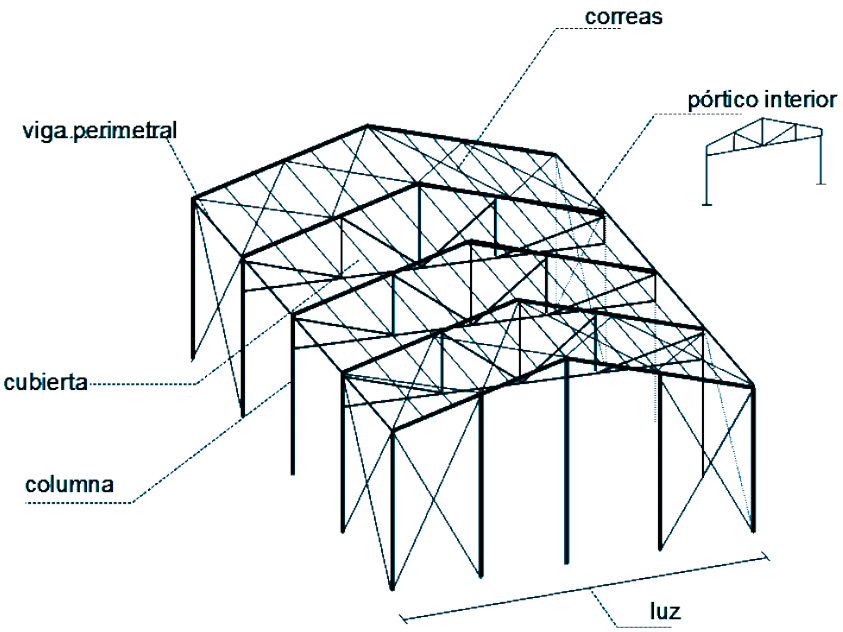


Figura 2.4 Galpón a dos aguas con tensores laterales en X

Fuente: Padilla & Portugal, 2013

2.2.9 Galpón a dos aguas con mampostería

Finalmente, aunque la mampostería no es usada con el fin de rigidizar la estructura, un buen diseño estructural toma en cuenta el aporte de rigidez de estos elementos a la estructura.

Figura 2.5 Galpón a dos aguas con mampostería

Fuente: Padilla & Portugal, 2013

2.2.10 Cargas

2.2.10.1 Carga muerta

También se denominan cargas permanentes porque se mantienen aplicadas indefinidamente en las estructuras. Se incluye el peso de elementos estructurales como muros, paredes, recubrimientos, instalaciones, etc. (Ministerio de Desarrollo Urbano y Vivienda, 2014)

2.2.10.2 Carga viva

(Ministerio de Desarrollo Urbano y Vivienda, 2014): La carga viva, también llamada sobrecargas de uso, que se utilizara en el cálculo depende de la ocupación a la que está destinada la edificación y están conformadas por los pesos de personas, muebles, equipos y accesorios móviles o temporales, mercadería en transición, y otras.

2.2.11 Cimentación

De forma general, se designa cimentación de una estructura a su parte más baja. Esta cumple una función de intermediario entre la estructura y el suelo para la transferencia de carga. Si la cimentación no se diseña de manera correcta, el suelo sobre el que se asienta estará sobre esforzado, lo que conlleva a que presente posibles asentamientos excesivos o fallas de corte. (Das & González, 2015)

2.2.11.1 Tipos de cimentaciones

Las cimentaciones se clasifican en dos grandes categorías, las cimentaciones superficiales o de poca profundidad y las cimentaciones profundas. Las primeras funcionan trasmitiendo las cargas al suelo justo debajo de ellas. Entre las más usadas se encuentra la zapata aislada, que mantiene los esfuerzos en el suelo por debajo del límite permitido al aplicar la carga sobre un área de tamaño adecuado.

Por otro lado, las cimentaciones profundas trasmiten gran parte de la carga, o en ocasiones la carga completa, a suelos más profundos. Entre su propia clasificación se encuentran los pilotes, son elementos estructurales esbeltos que pueden ser elaborados en sitio o prefabricados e hincados en el suelo. (Coduto, 2001)

2.2.11.2 Cimentaciones profundas

Las cimentaciones profundas son utilizadas en algunas situaciones, por lo general cuando el suelo cercano a la superficie no presenta una capacidad de carga suficiente para soportar las solicitaciones de la estructura. También se utilizan cuando los asentamientos esperados al utilizar cimentaciones superficiales exceden los límites permisibles. Otra utilidad que tienen es de disminuir los asentamientos diferenciales debido a la variación excesiva del suelo o a las cargas de la estructura. (Budhu, 2000)

2.2.11.3 Cimentaciones superficiales

Como se mencionó al inicio, las cimentaciones superficiales transfieren las cargas estructurales al suelo en profundidades relativamente pequeñas. Estas son fáciles de construir y requieren de poca o nada de mano de obra ni equipos especializados. Esta es una gran ventaja frente a las cimentaciones profundas, puesto que, si el suelo cercano a la superficie es competente, la cimentación superficial es más económica. (Salgado, 2008)

Otra ventaja de las cimentaciones superficiales es que permiten realizar una inspección momentos antes de la colocación del hormigón, esto implica que se puede comprobar si los supuestos sobre el suelo en la etapa de diseño son correspondientes.

Sin embargo, este tipo de cimentación presenta desventajas al trabajar sobre suelos muy compresibles. La interacción con estos suelos presenta asentamientos elevados, o en el caso de suelos expansivos, se dan asentamientos diferenciales debido a la gran diferencia de densidad de suelo entre cimientos.

2.2.11.4 Tipos de cimentaciones superficiales

La figura 1.1 muestra la disposición de los 4 tipos de cimentaciones superficiales. Primero se encuentra la zapata aislada, que corresponde a una cimentación que recibe la carga de solo una columna. Cuando la cimentación soporta dos columnas, se denomina zapata combinada. Para una línea de columnas o si la cimentación soporta un muro portante, se denomina zapata continua. Luego se presenta un caso especial de zapata combinada, el “Strap footing” corresponde a dos zapatas aisladas unidas por una viga de amarre. Y finalmente, la losa de cimentación que soporta muchas columnas en diferentes ejes estructurales.

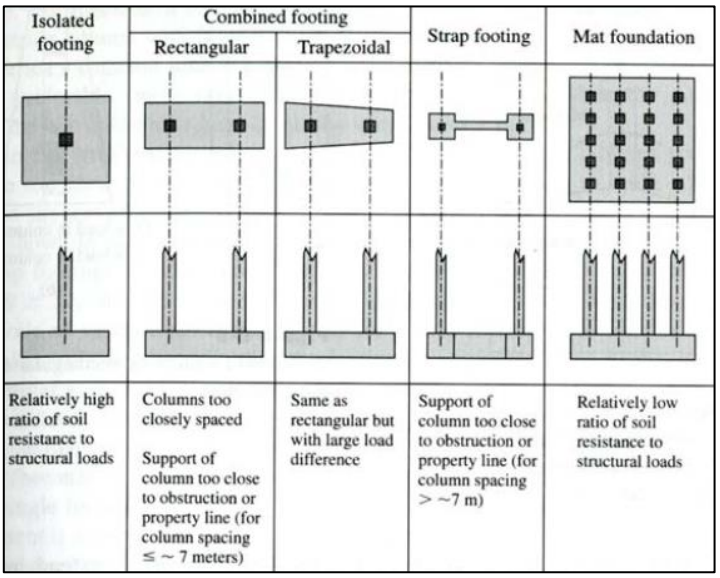


Figura 2.6 Tipos de cimentaciones superficiales

Fuente: Salgado, 2008

2.2.11.5 Cimentaciones rígidas y flexibles

Se puede clasificar las cimentaciones superficiales de acuerdo con su comportamiento flexible. Dependiendo de si la cimentación es flexible o rígida, la distribución de presión por debajo de esta, así como los asentamientos, varían como se muestra en la figura 1.2.

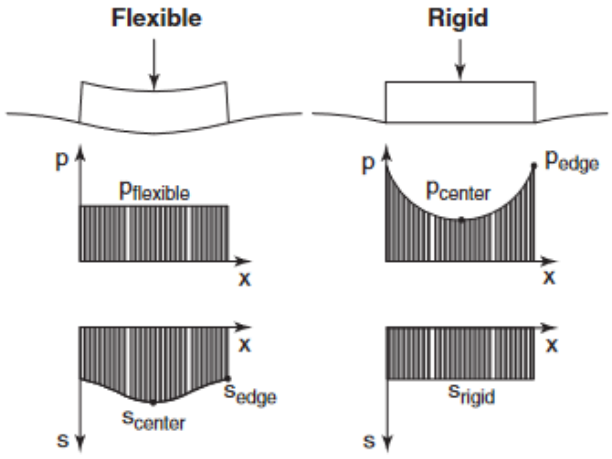


Figura 2.7 Distribución de presión y asentamientos debajo de una cimentación superficial

Fuente: Briaud, 2013

A la izquierda de la figura 1.2 se puede observar que las cimentaciones flexibles presentan una distribución de presión constante y asentamientos variables con un máximo en el centro. Por otro lado, la cimentación rígida de la derecha presenta asentamientos constantes y una distribución de presión variable con máximos a los bordes (Briaud, 2013).

2.2.11.6 Capacidad ultima de carga de cimentaciones superficiales

Para cimentaciones poco profundas, la capacidad última de carga corresponde al límite de carga que puede soportar un suelo antes de sufrir una falla de corte. Si a esta se le aplica un factor de seguridad, se obtiene la carga neta máxima que resistirá la cimentación. (Das & González, 2015)

2.2.12 Programación en C#

En esta sección se presenta los fundamentos del lenguaje de programación Visual C# necesarios para comprender la aplicación de la API de Sap2000 y Tekla Structures con este lenguaje.

2.2.12.1 Microsoft Visual Studio

El entorno de desarrollo integrado (IDE) de Visual Studio 2019 es un sistema de software para el diseño de aplicaciones que integra varias herramientas comunes del desarrollador en una misma interfaz gráfica de usuario. Este permite escribir código con menos errores de manera rápida y precisa, encontrar información o cambios realizados en el código, abrir su proprio repositorio para el control de versiones y mucho más.

2.2.12.2 Microsoft .NET Framework

La tecnología .NET Framework es un soporte para la construcción y ejecución de aplicaciones de Windows o servicios web. Sus principales características son generación de un ambiente de programación orientada a objetos de forma consistente, así como promover la ejecución de códigos seguros. Esta tecnología permite integrar cualquier código basado en .NET Framework con algún otro código (Wagner et al., 2020).

2.2.12.3 Clase

Es fundamental de C#, las clases son estructuras conformadas por campos y métodos. Los campos son estados en forma de variables y los métodos son un

conjunto de acciones que dirigen el comportamiento. Las clases definen a los objetos, de hecho, los objetos se crean a forma de plantilla mediante las clases (Wagner, 2020).

2.2.12.4 Espacio de nombre

Los espacios de nombres se utilizan de dos formas en la programación en C#. El objetivo de usarlos en .NET Framework es para organizar proyectos de códigos largos. Estos permiten diferenciar clases con el mismo nombre de diferentes proyectos. En otras palabras, son la manera en cómo están organizados los programas C# (Wagner, 2017).

2.2.12.5 Método

Es un bloque de código que contiene una serie de acciones. Los programas en C# ejecutan dichas acciones llamando a los métodos. Toda acción se realiza dentro del contexto de un método, por ejemplo, el inicio de cualquier aplicación se realiza llamando al método Main o principal.

2.2.13 API de Sap2000

La Interfaz de Programación de Aplicaciones de CSi (Computers & Structures Inc.) es una excelente herramienta que permite al usuario automatizar los procesos de modelación, análisis y diseño de estructuras. La API crea un puente entre el Sap2000 y los softwares de terceros que da lugar al intercambio de información entre programas.

2.2.13.1 Funciones por utilizar

La tabla 2.1 describe las funciones por utilizar de la API de Sap2000. Todas son necesarias para llegar a generar de forma automática el diseño correcto de los elementos de la nave industrial de geometría paramétrica.

Tabla 2.1 Funciones por utilizar de la API de Sap2000 Parte 1

Fuente: El autor

Categoría	Subcategoría	Función	Descripción
General Functions		ApplicationStart	Inicia la aplicación Sap2000.
		InitializeNewModel	Limpia el modelo actual e Inicia el programa para un nuevo modelo.
		SetPresentUnits	Define las unidades actuales.

File		NewBlank	Solo debe ser usada para crear un modelo nuevo, preferiblemente luego llamar a las funciones InitializeNewModel o ApplicationStart.
		Save	Si no se especifica un nombre, guarda el archivo con el nombre actual.

Tabla 2.2 Funciones por utilizar de la API de Sap2000 Parte 2

Fuente: El autor

Categoría	Subcategoría	Función	Descripción
Definition	Load Patterns	Add	Añade un nuevo patrón de carga.
	Load Patterns/Auto Seismic Load	SetUserCoeficient	Asigna valores de carga auto sísmica para tipos de carga de coeficiente de usuario.
	Properties/Frame	SetChannel	Inicia o modifica una sección tipo C.
		SetCircle	Inicia o modifica una sección circular.
		SetColdC	Inicia o modifica una sección tipo G.
		SetDblAngle	Inicia o modifica una sección tipo doble ángulo.
		SetDblChannel	Inicia o modifica una sección tipo doble C.
	Properties/Material	AddMaterial	Añade una nueva propiedad de material basado en propiedades de material predefinidas en el archivo instalado "CSiMaterialLibrary*.xml".
		SetMPIsotropic	Define el tipo de simetría direccional del material a isotrópico y asigna valores de propiedades mecánicas isotrópicas.
		SetMPUniaxial	Define el tipo de simetría direccional del material a uniaxial y asigna valores de propiedades mecánicas uniaxiales.
		SetOColdFormed	Asigna valores de propiedades mecánicas de diseño (FY, Fu, etc.) para materiales laminados en frío.
		SetOSteel_1	Asigna valores de propiedades mecánicas de diseño (FY, Fu, etc.) para materiales de acero.
Object Model	Frame Object	AddByCoord	Añade un nuevo objeto frame al modelo con coordenadas específicas.
		SetEndLengthOffset	Asigna los desfases de longitud en dirección del eje local 1.
		SetInsertionPoint	Asigna información de punto de inserción a un objeto frame.

		SetReleases	Realiza asignación de liberación de reacciones y rigidez parcial.
		SetSection	Asigna una propiedad de sección a un objeto frame.
		SetSelected	Define el estado de selección de objetos frame.
		SetTCLimits	Asigna límites de fuerza de tensión o compresión en objetos frame.
	Point Object	SetRestraint	Asigna restricciones de nodo a los objetos punto.

Tabla 2.3 Funciones por utilizar de la API de Sap2000 Parte 3

Fuente: El autor

Categoría	Subcategoría	Función	Descripción
Analyse	Analysis Results Setup	DeselectAllCasesAndCombosForOutput	Deselecciona todos los casos de carga y combinaciones para la salida de resultados.
		SetCaseSelectedForOutput	Establece un estado caso de carga para la salida de resultados.
		SetComboSelectedForOutput	Establece una combinación de carga para la salida de resultados.
	Results	FrameForce	Reporta las fuerzas en los objetos frame.
		ModalParticipatingMassRatios	Reporta las tazas de participación de masa de cada modo de vibración para cada caso modal de análisis.
		ModalPeriod	Reporta el periodo modal, frecuencia cíclica y eigenvalor para cada caso modal de carga seleccionado.
Design	Steel	SetComboStrength	Selecciona o deselecciona combinaciones de carga para el diseño de acero.
		StartDesign	Inicia el diseño de los frame de acero.
		VerifyPassed	Reporta los nombres de los objetos frame que no pasaron el chequeo.
	Steel/AISC360-10	SetOverwrite	Establece los valores de diseño de acero por sobrescribir.
	ColdFormed	SetComboStrength	Selecciona o deselecciona combinaciones de carga para el diseño de laminado en frío.

		StartDesign	Inicia el diseño de laminado en frío.
		VerifyPassed	Reporta los nombres de los objetos frame que no pasaron el chequeo.
	ColdFormed/AISI LRFD96	SetOverwrite	Establece los valores de diseño de laminado en frío por sobrescribir.
Select		PropertyFrame	Selecciona o deselecciona todos los objetos frame con sección especificada.
View		RefreshView	Refresca la vista de una ventana.

2.3 Trabajo de campo, laboratorio y gabinete

2.3.1 Desarrollo del programa

En esta sección se desarrolla la interfaz gráfica del programa con el lenguaje de programación Visual C#. Se utiliza las funciones de la tabla 2.1 para definir la geometría de la nave industrial, secciones, materiales, patrones de cargas y combinaciones de carga de acuerdo con la NEC-SE-DS. Aunque el programa solo trabaja con un tipo de geometría de nave industrial, es posible diseñar cualquier tipo de nave industrial tras agregar el algoritmo que genere su geometría.

2.3.2 Organización de la aplicación

El programa inicia ejecutando la ventana principal, en la cual el usuario modifica los parámetros necesarios para definir la geometría de la nave industrial y las cargas por aplicar sobre la estructura. La figura 2.3 muestra lo parámetros dentro de la ventana principal, que incluye una imagen que guía al usuario a definir correctamente los parámetros.

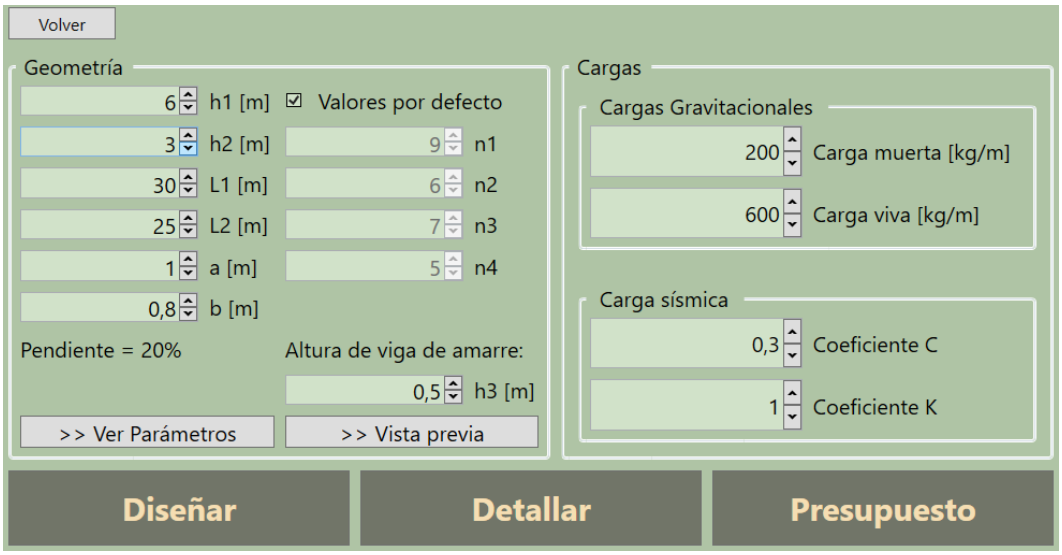


Figura 2.8 Ventana principal de la interfaz gráfica

Fuente: El autor

El proceso general del programa queda ilustrado en el diagrama de flujo de la figura 2.4. La subrutina “Análisis y diseño en Sap2000” incluye el prediseño de la estructura, el cálculo de la geometría, la generación del modelo, el análisis estructural y diseño de los elementos de acero y laminados en frío.

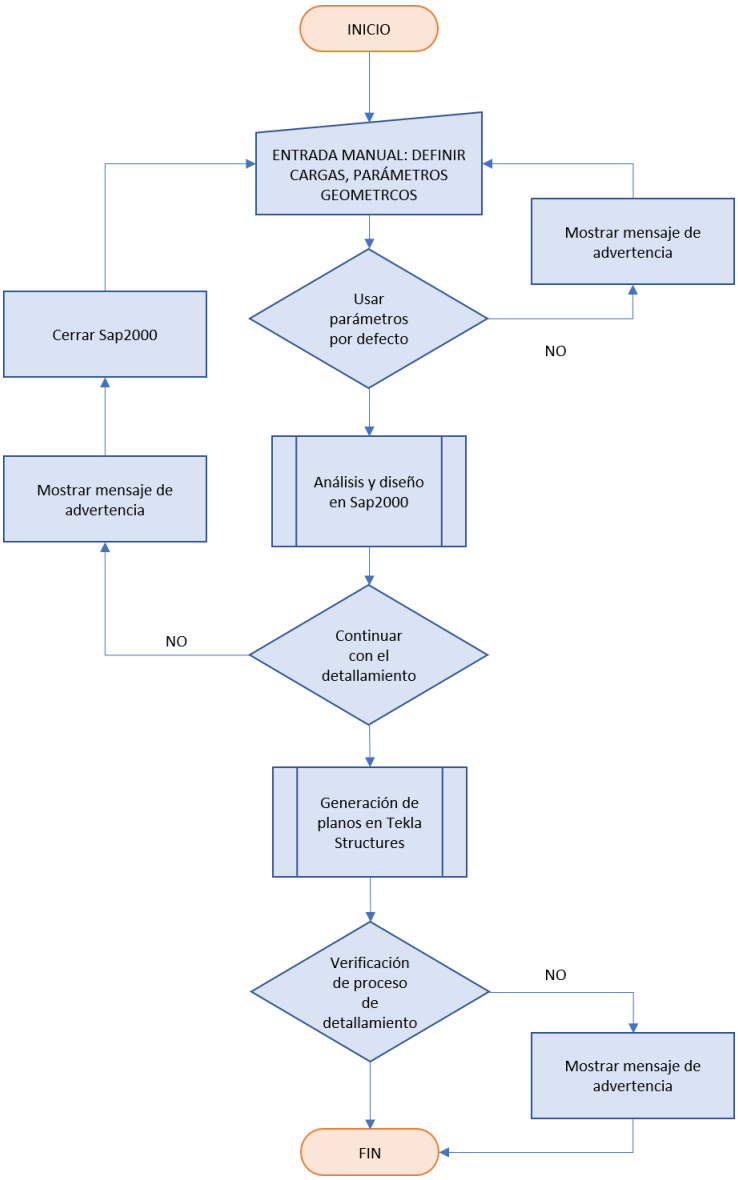


Figura 2.9 Diagrama de flujo general

Fuente: El autor

2.4 Planteamiento de alternativas

Las siguientes alternativas corresponden a los galpones más comunes construidos en Ecuador. La alternativa A es el diseño de un galpón curvo y las alternativas B, C y D son el diseño de un galpón a dos aguas. La diferencia entre estas tres últimas alternativas es el sistema de arriostramiento lateral que poseen. La alternativa B es un galpón a dos aguas con viga de amarre, la alternativa C posee tensores laterales y la alternativa D presenta mampostería confinada.

2.5 Análisis de alternativas

Para definir el tipo de nave industrial a diseñar, se calificó las cuatro alternativas en cinco criterios: estética, costo, tiempo, mano de obra y cantidad construidas al año.

Con estética se califica la presentación arquitectónica del galpón. En el criterio costo, tiene un peso mayor el galpón menos costoso. Respecto al tiempo, la opción que supone un mayo tiempo de construcción presenta una calificación menor. La mano de obra califica la calidad o grado de experticia necesaria para montar la estructura, a menor grado de experticia mayor calificación. Por último, la cantidad construidas al año refleja la diferencia numérica de cada tipo de galpón en Ecuador, a mayor cantidad por año mayor calificación.

La tabla 2.4 muestra el puntaje total de cada alternativa, con un máximo de 95 para el galpón a dos aguas con viga de amarre y un mínimo de 47 puntos para el galpón curvo. Aunque el galpón curvo presenta una buena estética, la cantidad construida al año de este tipo es muy baja, por lo que se descarta como solución.

Los galpones con tensores laterales generan los menores costos; sin embargo, no se construyen muchos de estos porque el uso de esos tensores limita el paso. Usar mampostería confinada presenta buena estética, pero necesita un poco más de experticia en la mano de obra para su construcción.

Tabla 2.4 Matriz de selección de alternativas

Fuente: El autor

Criterio	**Puntaje Ideal**	**Alternativas**			
			Galpón a dos aguas		
		Galpón curvo	**Con viga de amarre**	**Con tensores laterales**	**Con mampostería confinada**
Estética	25	24	20	10	22
Costo	25	10	21	23	18
Tiempo	15	5	21	22	17
Mano de obra	15	5	14	14	11
Cantidad construidas al año	20	3	17	10	13
Puntaje total	100	47	**95**	79	80

2.6 Selección de alternativa

Finalmente, se selecciona como solución diseñar el galpón a dos aguas con viga de amarre que presenta una buena estética, un costo relativamente bajo, sin necesidad de mayor grado de experticia en el montaje y con gran acogimiento en la industria de la construcción.

CAPÍTULO 3

3. DISEÑOS Y ESPECIFICACIONES

3.1 Diseños

A continuación, se presenta un ejemplo de nave industrial diseñada por el programa. Los parámetros del galpón se muestran en la figura 3.1, este tiene una altura máxima de 9 metros, luz de 30 metros y profundidad de 25 metros.

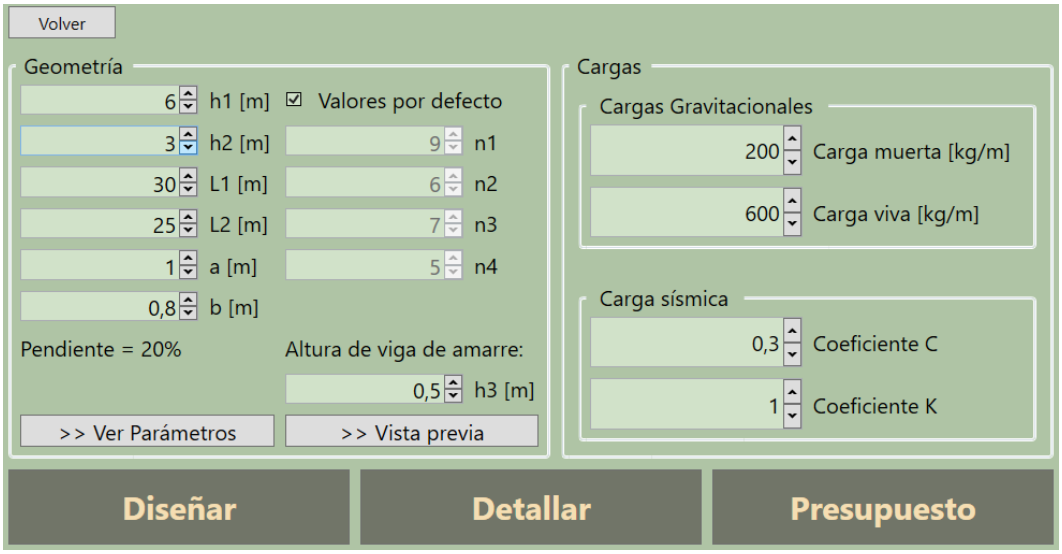


Figura 3.1 Parámetros del galpón a diseñar

Fuente: El autor

Al programa permite mostrar una vista previa del galpón como elementos tipo línea. La figura 3.2 presenta la ventana de vista previa donde se puede escoger distintas vistas y así darse una idea de cómo está constituido la estructura metálica.

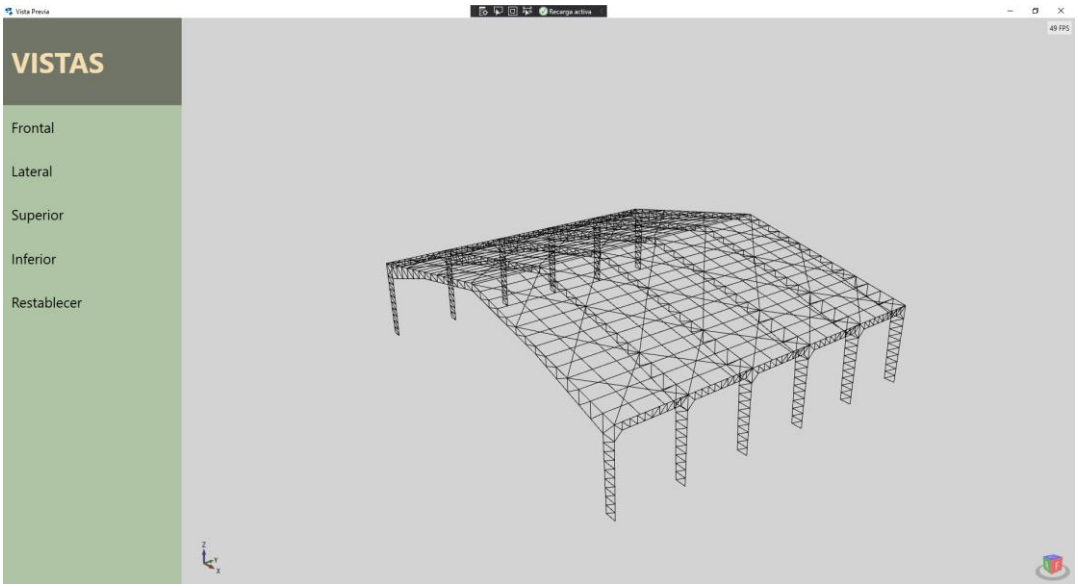


Figura 3.2 Ventana de vista previa

Fuente: El autor

Una vez se está seguro de que la geometría está bien insertada en el programa, se da clic en el botón “Diseñar”. Así empieza el proceso de diseño dividido en dos partes, la primera consiste en un cálculo dentro del propio programa a modo de prediseño, y la segunda corresponde al diseño y chequeo de la estructura en SAP2000. Todo este proceso se encuentra enlistado dentro del método Design en la figura 3.3.

Luego de hacer el prediseño, se ejecuta el método ExportSectionsFromExcel. Este genera secciones en SAP2000 de los perfiles metálicos encontrados en una base de datos local. El siguiente paso es generar el modelo completo de la nave industrial con los puntos de inserción correctos, esto se realiza con los métodos CalculateFullGeometry y AsignCorrectInsertionPoints.

Con la estructura correctamente modelada, se procede a definir los patrones de carga, combinaciones de carga, se añaden las cargas al modelo y se itera el diseño dentro del método IterativeDesign. Luego para concluir el diseño, se utilizan métodos que obtienen la geometría real que tendrá el galpón al momento de detallar y se diseña la cimentación.

```
1 referencia
public async Task Design()
{
    await Predesign();
    await ExportSectionsFromExcel();
    await CalculateFullGeometry();
    await AsignCorrectInsertionPoints();
    await esaSap2000.RefreshView();
    await IterativeDesign();
    numberofTensors -= 1;
    await GetElementInfoForDetailing();
    await SetElementInfo();
    CalculateRealGeometryForCFrames();
    CalculateRealGeometryForLFrames();
    CalculateRealGeometryForGFrames();
    await RetrieveJointForcesForDesignFoundation();
    DesignFoundation();
}
```

Figura 3.3 Método principal para diseñar la estructura

Fuente: El autor

3.1.1 Prediseño

Antes de crear el modelo de prediseño, primero se definen los materiales a utilizar. Se utilizó acero A36, que para los aceros estructurales se los pudo obtener directamente desde una base de datos del SAP2000. Sin embargo, como los perfiles locales utilizan el acero A36 como material para laminados en frío, se tuvo que definir este material manualmente como A36CF para poder diseñar las correas. Las propiedades mecánicas de cada material se muestran en la figura 3.4.

La figura 3.6 detalla cómo se introducen los elementos tipo frame al modelo. Estos cuatro elementos corresponden a un modelo sencillo de un pórtico tipo cercha de la figura 3.5.

```
public async Task SetPredesignModel()
{
    await esaSap2000.InitializeNewModel(eUnits.kip_in_F);
    await esaSap2000.CreateNewBlank();

    //Set new material
    MaterialName1 = "A36";
    await esaSap2000.AddMaterial(MaterialName1, eMatType.Steel, "United States", "ASTM A36", "Grade 36");
    //Set new material
    ColdFormedMaterial = "A36CF";
    await esaSap2000.SetMaterial(ColdFormedMaterial, eMatType.ColdFormed);
    await esaSap2000.SetMPIsotropic(ColdFormedMaterial, 29000, 0.3, 0.0000065);
    await esaSap2000.SetOColdFormed(ColdFormedMaterial, 36, 58, 1);
    //Set new material
    RebarMaterial = "";
    await esaSap2000.SetMaterial(RebarMaterial, eMatType.Steel);
    await esaSap2000.SetMPUniaxial(RebarMaterial, 29000, 0.0000065);
    await esaSap2000.SetOSteel(RebarMaterial, 60, 80, 66, 88, 1, 1, 0.011, 0.11, 0.17);
```

Figura 3.4 Definición de materiales

Fuente: El autor

En este ejemplo, se genera un pórtico y una carga muerta distribuida uniformemente de 200Kg/m y una carga viva de 600Kg/m.

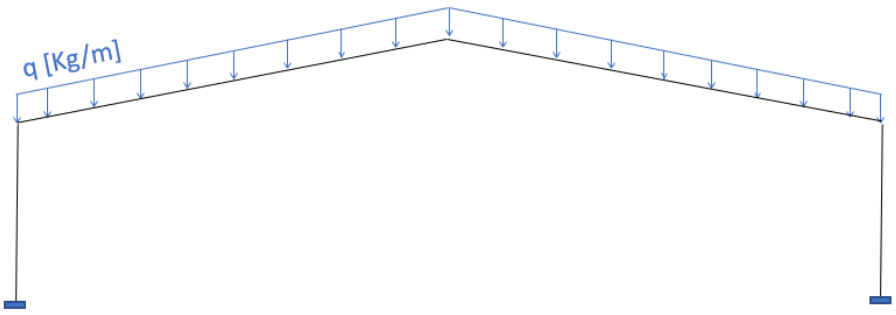


Figura 3.5 Modelo prediseño

Fuente: El autor

Cabe destacar que el ingreso de los elementos se de acuerdo con los parámetros geométricos definidos al iniciar el programa. Se puede observar en la figura 3.6 que las coordenadas de los elementos están en función de la luz y alturas definidas por el usuario.

```
//Add frames to the model
string FrameName1 = " ";
string FrameName2 = " ";
string FrameName3 = " ";
string FrameName4 = " ";
FrameName1 = await esaSap2000.FrameObjAddByCoord(0, 0, 0, 0, 0, h1value, FrameName1,
    FrameSectionName1, "1", "Global");
FrameName2 = await esaSap2000.FrameObjAddByCoord(0, 0, h1value, L1value / 2, 0, h1value + h2value,
    FrameName2, FrameSectionName1, "2", "Global");
FrameName3 = await esaSap2000.FrameObjAddByCoord(L1value / 2, 0, h1value + h2value, L1value, 0, h1value,
    FrameName3, FrameSectionName1, "3", "Global");
FrameName4 = await esaSap2000.FrameObjAddByCoord(L1value, 0, h1value, L1value, 0, 0,
    FrameName4, FrameSectionName1, "4", "Global");
```

Figura 3.6 Añadir objetos tipo frame al modelo

Fuente: El autor

Una vez estén generados los elementos, se asigna las restricciones a los nodos de la base, de acuerdo con la figura 3.7.

```
//Set joints restrictions
string PointName1 = " ";
string PointName2 = " ";
bool[] Restraint = new bool[6] { true, true, true, true, true, true };
var tuple =  await esaSap2000.FrameObjGetPoints(FrameName1, PointName1, PointName2);
await esaSap2000.PointObjStRestraint(tuple.Item1, Restraint);
tuple = await esaSap2000.FrameObjGetPoints(FrameName4, PointName1, PointName2);
await esaSap2000.PointObjStRestraint(tuple.Item2, Restraint);

await esaSap2000.RefreshView();
```

Figura 3.7 Asignar restricciones a nodos

Fuente: El autor

Se definen los patrones de carga, casos de carga, y para este caso de prediseño, se define una combinación de carga de servicio (Figura 3.8). Luego se asignan las cargas a los elementos y se ejecuta el análisis estructural. La figura 3.9 muestra cómo se selecciona los casos de carga para la salida de resultados del análisis y la generación de una lista de resultados mediante la clase FrameForce y su método RetrieveForces que se detalla en el anexo 1.

```
//Add load patterns
await esaSap2000.LoadPatternsAdd("Live", eLoadPatternType.Live, 0, true);
await esaSap2000.LoadPatternsChangeName("Dead", "CM");
await esaSap2000.LoadPatternsChangeName("Live", "CV");
await esaSap2000.LoadPatternsSetSelfWTMultiplier("CM", 0);

//Add load cases
await esaSap2000.LoadCasesChangeName("Dead", "CM");
await esaSap2000.LoadCasesChangeName("Live", "CV");

//Add load combination
await esaSap2000.RespComboAdd("CM + CV", 0);
var NameType = eCNameType.LoadCase;
await esaSap2000.RespComboSetCaseList("CM + CV", NameType, "CM", 1);
await esaSap2000.RespComboSetCaseList("CM + CV", NameType, "CV", 1);
```

Figura 3.8 Definir patrones, casos y combinaciones de carga

Fuente: El autor

Respecto a la figura 3.9, antes de recuperar los resultados del análisis se utiliza la función SetPresentUnits para cambiar las unidades actuales del SAP2000 a Ton_m_C.

```
//Save model
await esaSap2000.FileSave();

//Run analysis
await esaSap2000.AnalyzeRunAnalysis();

await esaSap2000.ResultsDeselectAllCasesAndCombosForOutput();
await esaSap2000.ResultsSetCaseSelectedForOutput("CM");
await esaSap2000.ResultsSetCaseSelectedForOutput("CM");
await esaSap2000.ResultsSetComboSelectedForOutput("CM + CV");

await esaSap2000.SetPresentUnits(eUnits.Ton_m_C);

foreach (var FrameName in FrameList)
{
    List<FrameForce> frameForces = await FrameForce.RetrieveForces(esaSap2000, FrameName, eItemTypeElm.ObjectElm);
    FrameForceResultsList.Add(frameForces);
}
```

Figura 3.9 Correr el análisis estructural y recuperar los resultados

Fuente: El autor

La figura 3.10 muestra el diagrama de momentos, del cual se determina un momento máximo de 41 T-m.

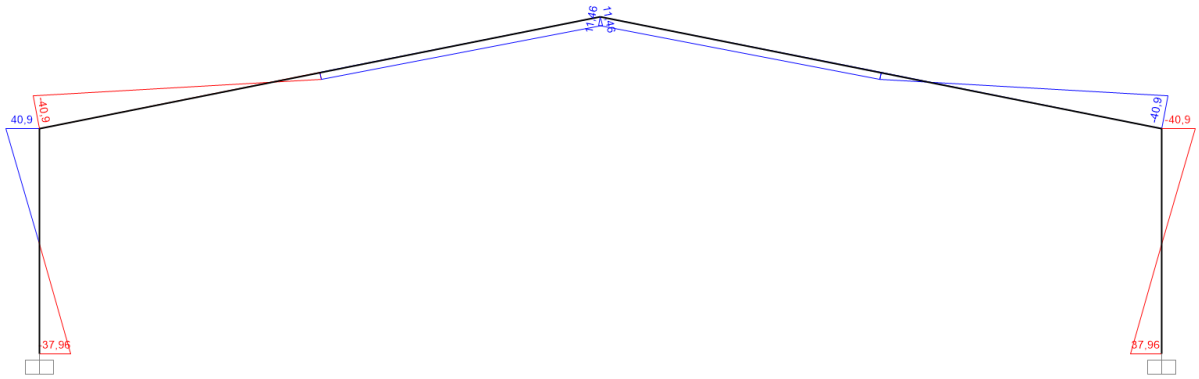


Figura 3.10 Diagrama de momentos del modelo de prediseño

Fuente: El autor

Para la selección de los perfiles el programa primero abre un archivo de Excel que contiene un catálogo de perfiles locales (Figura 3.11).

```
//Open Excel database
string path = "D:\\DOCUMENTOS\\Kevin\\TESIS\\Visual Studio\\Nave Industrial Sap\\bin\\Debug\\CATALOGO DE DIPAC.xlsx";
var file = File.Open(path, FileMode.Open, FileAccess.Read);
var reader = ExcelReaderFactory.CreateReader(file);
result = reader.AsDataSet(new ExcelDataSetConfiguration()
{
    ConfigureDataTable = (_) => new ExcelDataTableConfiguration()
    {
        UseHeaderRow = true
    }
});
```

Figura 3.11 Leer un archivo de Excel con el catálogo de perfiles

Fuente: El autor

El perfil tipo C utilizado como cordones de la cercha se seleccionan obteniendo el momento máximo de la lista de resultados generada FrameForceResultList (Figura 3.13). Luego se calcula el área necesaria del perfil para resistir una fuerza proporcional al parámetro a1 que corresponde al ancho mayor de la columna.

En este ejemplo, la fuerza de tensión en el perfil se calcula tomando un brazo de a1 igual a 1m, como lo muestra la figura 3.12. Para la tensión se tomará aproximadamente el 0.6 de fy, es decir: 1518Kg/cm2 entonces el área necesaria para una fuerza de 41T sería:

$$A = \frac{F}{fy} = \frac{M}{a1} * \frac{1000}{1518}$$

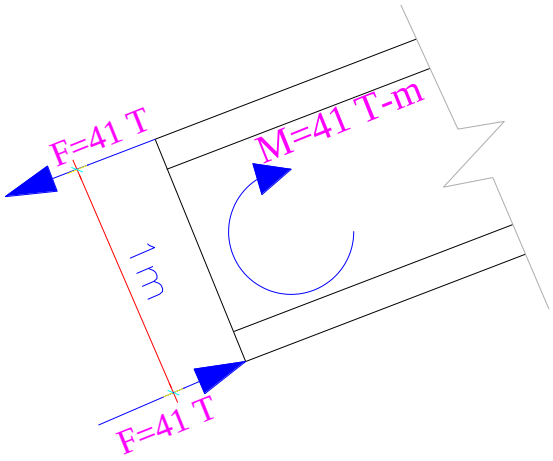


Figura 3.12 Fuerza de tensión en los cordones del prediseño

Fuente: El autor

Luego se utiliza el método FindCorrectSection que busca un perfil en la base de datos que tenga un área mayor o igual al área necesaria.

```
//Selection of C section
momentoMaximo = 0;

foreach (var tuple in FrameForceResultsList)
{
    foreach (var c in tuple)
    {
        if (Math.Abs(c.M3) > momentoMaximo)
        {
            momentoMaximo = Math.Abs(c.M3);
        }
    }
}

double AreaSectionC = (momentoMaximo / a1) * 1000 / 1518;
Trace.WriteLine(AreaSectionC);
Tuple<string, int> SectionCtuple = FindCorrectSection(result.Tables, 1, AreaSectionC, "SECCION");
SectionCName = SectionCtuple.Item1;
SectionCindex = SectionCtuple.Item2;
Trace.WriteLine(SectionCindex);
PredesignSectionCname = SectionCName;
PredesignSectionCindex = SectionCindex;
```

Figura 3.13 Prediseño de los cordones

Fuente: El autor

Un proceso similar se utiliza para los demás perfiles, primero se calcula el área necesaria y luego se busca un área mayor o igual en la base de datos. Los ángulos se diseñan para resistir el cortante de la columna. El diagrama de cortante se muestra en la figura 3.14.

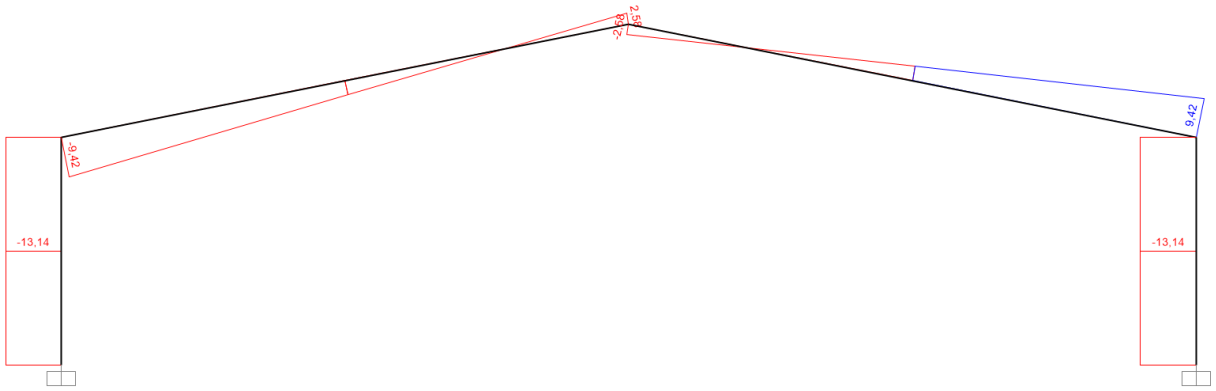


Figura 3.14 Diagrama de cortante del modelo de prediseño

Fuente: El autor

Del diagrama se obtiene un cortante de 13T, si se asume que la diagonal esta inclinada en 45 grados como en la figura 3.15, entonces se tendría una fuerza actuante de 18.5T.

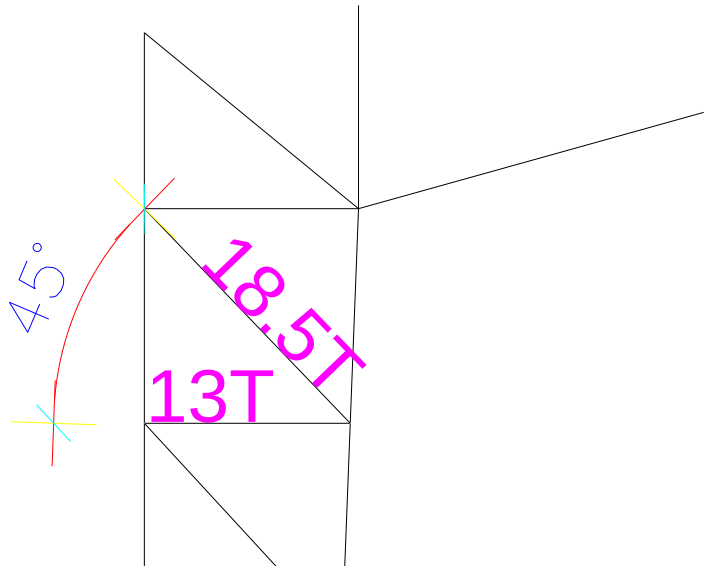


Figura 3.15 Fuerza máxima en los ángulos del prediseño

Fuente: El autor

La figura 3.16 presenta el código para el prediseño de los ángulos. Primero se determina el cortante máximo en la columna dentro de la lista de resultados, y luego se calcula el área necesaria para resistir ese cortante.

```
//Selection of angle section
double cortanteColumna = 0;

foreach (var tuple in FrameForceResultsList)
{
    foreach (var c in tuple)
    {
        if (Math.Abs(c.V2) > cortanteColumna)
        {
            cortanteColumna = Math.Abs(c.V2);
        }
    }
}

double cortanteAngulo = cortanteColumna / Math.Sin(45 * Math.PI / 180);
double AreaSectionL = (cortanteAngulo / 1.5) / 2;
Tuple<string, int> SectionLtuple = FindCorrectSection(result.Tables, 4, AreaSectionL, "SECCION");
SectionLName = SectionLtuple.Item1;
SectionLindex = SectionLtuple.Item2;
PredesignSectionLname = SectionLName;
PredesignSectionLindex = SectionLindex;
```

Figura 3.16 Prediseño de los ángulos

Fuente: El autor

El prediseño del perfil doble C, que se utiliza como diagonal al final de la columna, se realiza calculando primero la resultante actuante en el elemento, como lo muestra la figura 3.17.

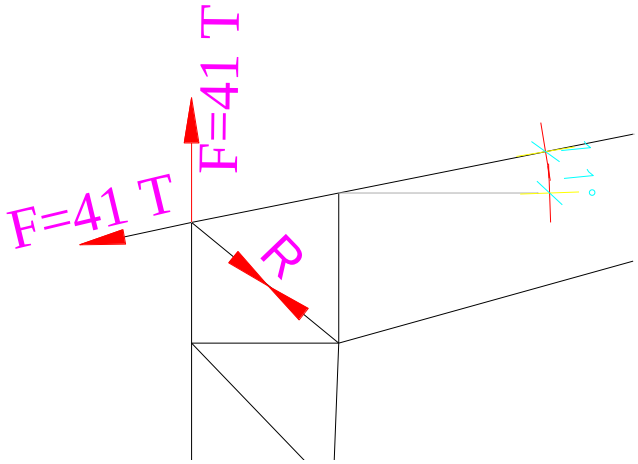


Figura 3.17 Fuerza resultante en el perfil doble C

Fuente: El autor

La figura 3.18 muestra el cálculo de esta fuerza resultante en función del momento máximo, el parámetro a1 y el ángulo de la pendiente de la cubierta. Luego se sigue el mismo procedimiento para determinar el perfil necesario.

```
//Selection of DoubleC section
angulo = Math.Atan(h2value / (L1value / 2));
double resultant = Math.Sqrt(Math.Pow(momentoMaximo / a1, 2)
    + Math.Pow((1 - Math.Sin(angulo)) * momentoMaximo / a1, 2));
double AreaSectionDoubleC = (resultant / 1.5) / 2;
Tuple<string, int> SectionDoubleCtuple = FindCorrectSection2(result.Tables, 1, AreaSectionDoubleC,
    "SECCION", result.Tables[1].Rows[SectionCindex].Field<double>("A"));
SectionDoubleCName = "2" + SectionDoubleCtuple.Item1;
SectionDoubleCindex = SectionDoubleCtuple.Item2;
PredesignSectionDoubleCname = SectionDoubleCName;
PredesignSectionDoubleCindex = SectionDoubleCindex;
```

Figura 3.18 Prediseño del perfil doble C

Fuente: El autor

Para finalizar el prediseño, para la selección de las correas, primero se calcula las cargas aplicadas sobre una correa con un ancho tributario de 1.2m. Luego, se recorre la base de datos de perfiles G, y se calculan los momentos resistentes y actuantes. Luego se comprueba que la proporción de momentos del perfil elegido sea menor a uno, como validación de la flexión biaxial. Este proceso se muestra en la figura 3.19.

```
//Selection of G section
int numberofSpaces = (int)Math.Ceiling((decimal)(L2value / 6));
double Yseparation = L2value / numberofSpaces;
cargaMuertaG = cargaMuerta * 1.2 / Yseparation;
cargaVivaG = cargaViva * 1.2 / Yseparation;
double cargaG = cargaMuertaG + cargaVivaG;

SectionGName = "";
SectionGindex = 0;
double area = Double.PositiveInfinity;

GSectionTable = result.Tables[0].Rows;
for (int i = 1; i <= GSectionTable.Count - 1; i++)
{
    string SectionName = GSectionTable[i].Field<string>("PERFIL");
    double Sx = GSectionTable[i].Field<double>("Wx");
    double Sy = GSectionTable[i].Field<double>("Wy");
    double Mrx = cargaG * Math.Cos(angulo) * Math.Pow(Yseparation, 2) * 100 / 8;
    double Mry = cargaG * Math.Sin(angulo) * Math.Pow(Yseparation, 2) * 100 / 8;
    double Mcx = 1500 * Sx;
    double Mcy = 1500 * Sy;
    double ratio = Mrx / Mcx + Mry / Mcy;
    double A = GSectionTable[i].Field<double>("SECCION");

    if (ratio <= 1 && A <= area)
    {
        SectionGName = SectionName;
        area = A;
        SectionGindex = i;
    }
}
```

Figura 3.19 Prediseño de las correas

Fuente: El autor

3.1.2 Exportación de secciones

Se generan las secciones en SAP2000 utilizando una base de datos en Excel. La figura 3.20 muestra cómo se usa la función PropFrameSetChannel para generar una sección en SAP2000.

```
// Exporting sections from EXCEL
await esaSap2000.SetPresentUnits(eUnits.kgf_mm_C);

//Exporting C sections
CSectionTable = result.Tables[1].Rows;
for (int i = 1; i <= CSectionTable.Count - 1; i++)
{
    string SectionName = CSectionTable[i].Field<string>("PERFIL");
    double t3 = CSectionTable[i].Field<double>("A");
    double t2 = CSectionTable[i].Field<double>("B");
    double tf = CSectionTable[i].Field<double>("e");
    double tw = CSectionTable[i].Field<double>("e");

    await esaSap2000.PropFrameSetChannel(SectionName, MaterialName1, t3, t2, tf, tw);

    //Set section properties modifiers
    await esaSap2000.PropFrameSetModifiers(SectionName, ModValue);
}
```

Figura 3.20 Exportar secciones tipo C de Excel a SAP2000

Fuente: El autor

Para los ángulos se debe agregar un paso más antes de exportar las secciones. SAP2000 permite definir las secciones doble ángulo espalda contra espalda, por lo tanto, se necesita calcular la distancia entre espaldas que presenten una inercia equivalente a la posición real de los ángulos. La figura 3.21 muestra un ejemplo de la posición real de los ángulos (arriba) y como los define SAP2000 (abajo).

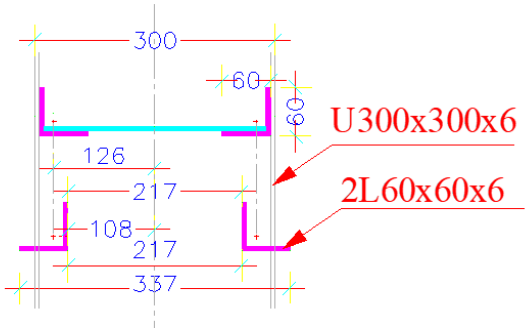


Figura 3.21 Diferencia de posición de ángulos en SAP2000

Fuente: El autor

La figura 3.22 presenta el cálculo de la distancia entre ángulos necesaria y como se utiliza la función PropFrameSetDblAngle para definir una sección tipo doble ángulo.

```
//Exporting L sections
LSectionTable = result.Tables[4].Rows;
double CA = CSectionTable[SectionCindex].Field<double>("A");
double Ce = CSectionTable[SectionCindex].Field<double>("e");
for (int i = 1; i <= LSectionTable.Count - 1; i++)
{
    string SectionName = LSectionTable[i].Field<string>("PERFIL");
    double t3 = LSectionTable[i].Field<double>("A");
    double tf = LSectionTable[i].Field<double>("e");
    double tw = LSectionTable[i].Field<double>("e");
    double Xb = LSectionTable[i].Field<double>("X=Y") * 10;
    double x = CA / 2 - Ce - Xb;
    double x1 = (-Xb + Math.Sqrt(Math.Pow(Xb, 2) - 4 * 0.25 * (Math.Pow(Xb, 2) - Math.Pow(x, 2)))) / (2 * 0.25);
    double x2 = (-Xb - Math.Sqrt(Math.Pow(Xb, 2) - 4 * 0.25 * (Math.Pow(Xb, 2) - Math.Pow(x, 2)))) / (2 * 0.25);
    double dis = Math.Max(x1, x2);
    double t2 = LSectionTable[i].Field<double>("B") * 2 + dis;

    await esaSap2000.PropFrameSetDblAngle(SectionName, MaterialName1, t3, t2, tf, tw, dis);

    //Set section properties modifiers
    await esaSap2000.PropFrameSetModifiers(SectionName, ModValue);
}
```

Figura 3.22 Exportar secciones tipo 2L de Excel a SAP2000

Fuente: El autor

Las secciones de perfil tipo doble C, tipo G y varilla de refuerzo se generan de forma similar al perfil tipo C. La única diferencia es la función utilizada, para la sección doble C se utiliza la función PropFrameSetDblChannel, para la sección G se utiliza la función PropFrameSetColdC y para las varillas de refuerzo se utiliza PropFrameSetCircle.

3.1.3 Generación de la geometría completa de la estructura

El método principal para la generación de la geometría se muestra en la figura 3.23. En este se calcula varios valores como, la separación entre pórticos, la cantidad de crucetas en la cubierta de cada tipo. Como solución a la disposición de las crucetas, se definieron crucetas de 3, 4, 5 y 6 espacios, es decir, que abarcan 4, 5, 6 o 7 correas.

Por cada pórtico se calcula la geometría del propio pórtico, la geometría de las correas y tensores conectados a él y la viga de amarre correspondiente entre

pórticos. Los anexos 2 al 5 contienen los métodos para calcular cada una de estas geometrías.

```
public async Task CalculateFullGeometry()
{
    double Ycoor = 0;
    int numberofSpaces = nParts4;
    int numberofTrusses = numberofSpaces + 1;
    double Yseparation = L2value / numberofSpaces;
    int nPartsCerchaAmarre = calculateNParts(Yseparation - 2 * h3value, 0.5) / 2;
    int nOfspaces = nParts2 + nParts3 + 1;
    Tuple<int, int, int, int> tupleCrosses = calculateNumberOfTrusses(nOfspaces);
    int nOf6Crosses = tupleCrosses.Item1;
    int nOf5Crosses = tupleCrosses.Item2;
    int nOf4Crosses = tupleCrosses.Item3;
    int nOf3Crosses = tupleCrosses.Item4;
    int nOf6Crossesdif = 0;
    int nOf5Crossesdif = 0;
    int nOf4Crossesdif = 0;
    int nOf3Crossesdif = 0;
    TotalNumberOfGFrames = 2 * (nParts2 + nParts3 + 1);

    for (int i = 1; i <= numberofTrusses; i++)
    {
        await CalculateTrussGeometry(i, Ycoor);
        await CalculateGFramesGeometry(i, Ycoor);
        await CalculateTensorsGeometry(i, Ycoor, Yseparation, nOf6Crosses, nOf5Crosses, nOf4Crosses,
            nOf3Crosses, nOf6Crossesdif, nOf5Crossesdif, nOf4Crossesdif, nOf3Crossesdif);
        await esaSap2000.RefreshView();
        Ycoor += Yseparation;
    }

    Ycoor = 0;
    for (int i = 1; i <= numberofTrusses; i++)
    {
        await CalculateTieBeamsGeometry(i, Ycoor, Yseparation, nPartsCerchaAmarre);
        await esaSap2000.RefreshView();
        Ycoor += Yseparation;
    }
}
```

Figura 3.23 Cálculo de la geometría completa de la estructura

Fuente: El autor

3.1.4 Diseño iterativo

Tras modelar la estructura completa, se ejecuta el método IterativeDesign mostrado en la figura 3.24. Este método ejecuta otras acciones preliminares como definir los patrones de carga sísmica, añadir las combinaciones de carga correspondiente a la norma NEC-SE-CG, añadir las cargas a las correas y seleccionar las combinaciones creadas para el chequeo.

```
1 referencia
public async Task IterativeDesign()
{
    await AddSeismicLoadpatterns();
    await AddLoadCombinations();
    await SelectDesignCombinations();
    await SetDesignOverwrites();
    await AddForcesToGFrames();
    await AnalyseAndDesign();
}
```

Figura 3.24 Método de pasos preliminares al diseño

Fuente: El autor

Los patrones de carga sísmica se definen utilizando las funciones mostradas en la figura 3.25.

```
1 referencia
public async Task AddSeismicLoadpatterns()
{
    //Add sism load pattern
    string[] array = new string[1] { "CM" };
    double[] array2 = new double[1] { 1 };
    await esaSap2000.SourceMassSetMassSource("MSSSRC1", true, false, true, true,
        1, array, array2);
    await esaSap2000.LoadPatternsAdd("Sx", eLoadPatternType.Quake, 0, true);
    await esaSap2000.LoadPatternsAdd("Sy", eLoadPatternType.Quake, 0, true);
    await esaSap2000.LoadPatternsAutoSeismicSetUserCoefficient("Sx", 1, 0.05,
        false, 0, 0, CoeficientC, CoeficientK);
    await esaSap2000.LoadPatternsAutoSeismicSetUserCoefficient("Sy", 2, 0.05,
        false, 0, 0, CoeficientC, CoeficientK);
}
```

Figura 3.25 Añadir patrones de carga sísmica

Fuente: El autor

Las combinaciones de carga se añaden con las funciones RespComboAdd y RespComboSetCaseList. La figura 3.26 muestra un ejemplo de cómo se crea la combinación del 90% de la carga muerta más el sismo en dirección x.

```
1 referencia
public async Task AddLoadCombinations()
{
    //Adding load combinations
    await esaSap2000.RespComboAdd("0.9D+Sx", 0);
    eCNameType NameType = eCNameType.LoadCase;
    await esaSap2000.RespComboSetCaseList("0.9D+Sx", NameType, "CM", 0.9);
    await esaSap2000.RespComboSetCaseList("0.9D+Sx", NameType, "Sx", 1);
```

Figura 3.26 Ejemplo de creación de combinaciones de carga

Fuente: El autor

Luego se seleccionan las combinaciones de carga a ser utilizadas al momento de chequear los elementos. Esto se realiza con la función DesignSteelSetComboStrength y su homóloga para elementos laminados en frío DesignColdFormedSetComboStrength como se muestra en la figura 3.27.

```
1 referencia
public async Task SelectDesignCombinations()
{
    //Select design combinations
    await esaSap2000.DesignSteelSetComboAutoGenerate(false);
    await esaSap2000.DesignSteelSetComboStrength("0.9D+Sx", true);
    await esaSap2000.DesignSteelSetComboStrength("0.9D+Sy", true);
    await esaSap2000.DesignSteelSetComboStrength("0.9D-Sx", true);
    await esaSap2000.DesignSteelSetComboStrength("0.9D-Sy", true);
    await esaSap2000.DesignSteelSetComboStrength("1.2D+1.6L", true);
    await esaSap2000.DesignSteelSetComboStrength("1.2D+L+Sx", true);
    await esaSap2000.DesignSteelSetComboStrength("1.2D+L+Sy", true);
    await esaSap2000.DesignSteelSetComboStrength("1.2D+L-Sx", true);
    await esaSap2000.DesignSteelSetComboStrength("1.2D+L-Sy", true);
    await esaSap2000.DesignSteelSetComboStrength("1.4D", true);

    await esaSap2000.DesignColdFormedSetComboAutoGenerate(false);
    await esaSap2000.DesignColdFormedSetComboStrength("0.9D+Sx", true);
    await esaSap2000.DesignColdFormedSetComboStrength("0.9D+Sy", true);
    await esaSap2000.DesignColdFormedSetComboStrength("0.9D-Sx", true);
    await esaSap2000.DesignColdFormedSetComboStrength("0.9D-Sy", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.2D+1.6L", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.2D+L+Sx", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.2D+L+Sy", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.2D+L-Sx", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.2D+L-Sy", true);
    await esaSap2000.DesignColdFormedSetComboStrength("1.4D", true);
}
```

Figura 3.27 Selección de combinaciones de carga para el chequeo

Fuente: El autor

Para el correcto chequeo de la estructura, se debe sobrescribir las propiedades de diseño de los elementos para que correspondan a un sistema ordinario a momentos (OMF), porque SAP2000 asigna por defecto sistemas resistentes a momentos (SMF).

Esto se realiza utilizando las funciones DesignSteelAISC360_10SetOverwrite en el caso de perfiles estructurales, y DesignColdFormedAISI_LRFD96SetOverwrite para los perfiles laminados en frío. Más información de cómo se utilizan estas funciones se encuentra en la API de SAP2000.

Como último paso previo al diseño de los elementos, se asignan las cargas usando la función FrameObjSetLoadDistributed como se muestra en la figura 3.28.

```
1 referencia
public async Task AddForcesToGFrames()
{
    //Adding forces to G frames
    for (int i = 1; i <= numberofGframes; i++)
    {
        await esaSap2000.SetPresentUnits(eUnits.kgf_m_C);
        double loadValue = cargaMuertaG;
        await esaSap2000.FrameObjSetLoadDistributed("G-" + Convert.ToString(i), "CM", 1, 10, 0, 1,
            loadValue, loadValue, "Global", true, true, eItemType.Objects);

        loadValue = cargaVivaG;
        await esaSap2000.FrameObjSetLoadDistributed("G-" + Convert.ToString(i), "CV", 1, 10, 0, 1,
            loadValue, loadValue, "Global", true, true, eItemType.Objects);
    }

    numberofCframes -= 1;
    numberofLframes -= 1;
    numberofDoubleCframes -= 1;
    numberofGframes -= 1;
}
```

Figura 3.28 Asignación de cargas a las correas

Fuente: El autor

Al ejecutarse el método AnalyseAndDesign suceden tres procesos, primero se corrige las secciones del prediseño respecto a su tamaño geométrico. Por ejemplo, si del prediseño se obtiene para los cordones un perfil C300x300x6, como máximo se podría tener perfiles L125x125x6 que quepan correctamente dentro del perfil C.

Luego se manda a correr el diseño de los elementos estructurales de acero y se obtiene el número de elementos que no pasaron el chequeo, lo mismo se hace

para los elementos con perfil laminado en frío. La figura 3.29 muestra el método ActualizeNumberOfNotPassed que retorna las cantidades respectivas de los elementos que no pasaron los chequeos.

```
2 referencias
public async Task ActualizedNumberOfNotPassed()
{
    //Star Cold Formed Design
    await esaSap2000.AnalyzeRunAnalysis();
    await esaSap2000.DesignColdFormedStartDesign();
    int numberitems1 = 0;
    int numbernotpassed1 = 0;
    int numbernotchecked1 = 0;
    string[] list1 = new string[0];
    var tuple = await esaSap2000.DesignColdFormedVerifyPassed(numberitems1,
        numbernotpassed1, numbernotchecked1, list1);
    numberitems1 = tuple.Item1;
    numbernotpassed1 = tuple.Item2;
    numbernotchecked1 = tuple.Item3;
    list1 = tuple.Item4;
    await esaSap2000.SetModelIsLocked();
    numberofColdFormedNotPass = numbernotpassed1;

    //Star Steel Design
    await esaSap2000.AnalyzeRunAnalysis();
    await esaSap2000.DesignSteelStartDesign();
    tuple = await esaSap2000.DesignSteelVerifyPassed(numberitems1, numbernotpassed1,
        numbernotchecked1, list1);
    numberitems1 = tuple.Item1;
    numbernotpassed1 = tuple.Item2;
    numbernotchecked1 = tuple.Item3;
    list1 = tuple.Item4;
    await esaSap2000.SetModelIsLocked();
    numberofSteelNotPass = numbernotpassed1;

    Trace.WriteLine("ActualizedNumberOfNotPassed");
    Trace.WriteLine(numberofColdFormedNotPass + " " + numberofSteelNotPass);
}
```

Figura 3.29 Método de verificación de elementos

Fuente: El autor

Finalmente, se ejecuta el método Iterar, mostrado en la figura 3.30 que actualizará las cantidades de elementos que no pasan el chuequeo y en caso de estos valores ser mayores a cero, asignan una sección mayor a los elementos. Esto se itera hasta que todos elementos de acero y de acero laminado en frío pasen el chequeo.

```
1 referencia
public async Task Iterate()
{
    do
    {
        while (numberofColdFormedNotPass > 0)
        {
            numberofColdFormedNotPass = await actualizeColdFormedDesignResults();
        }

        while (numberofSteelNotPass > 0)
        {
            numberofSteelNotPass = await actualizeSteelDesignResults();
        }


        await ActualizedNumberOfNotPassed();
    } while (numberofColdFormedNotPass != 0 || numberofSteelNotPass != 0);
}
```

Figura 3.30 Método que itera el diseño de los elementos

Fuente: El autor

Las secciones finales para este galpón se encuentran en la figura 3.31, para los cordones se determinó un perfil C300x100x12, para los ángulos un perfil L75x75x10, para las correas un perfil G150x75x5 y para los tensores se usaría varilla de 18mm.

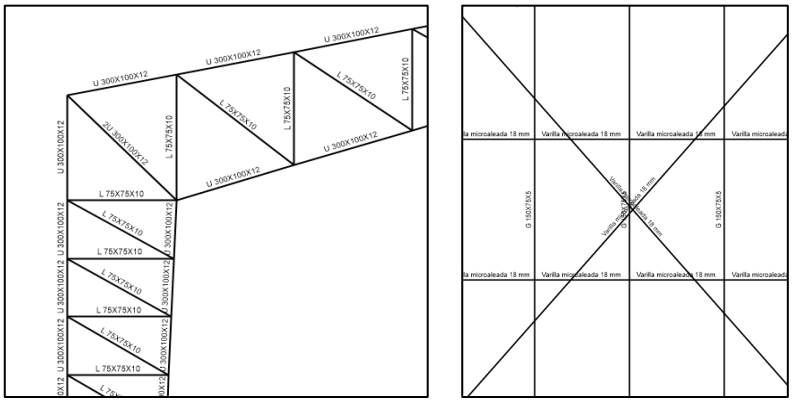


Figura 3.31 Secciones finales del galpón de ejemplo

Fuente: El autor

3.1.5 Diseño de la cimentación

Para diseñar la cimentación primero se debe obtener los resultados del análisis estructural para las combinaciones de carga de servicio y de las cargas mayoradas para el diseño estructural de la cimentación. El método que recupera estos resultados se encuentra en el anexo 6.

Estos resultados corresponden a las reacciones de los nodos en la base de la estructura, la figura 3.32 presenta el cálculo de las cargas de servicio Ps, Msx y Msy para la cimentación, donde F3point1 y F3point2 corresponden a la reacción en el eje Z de los nodos de la base y M1point1 y M1point2 serían momentos en ejes locales correspondientes al eje global Y.

Además, se calculan los valores de cargas mayoradas Pu, Mux y Muy y se asigna las dimensiones mínimas por geometría del pedestal que se apoya sobre la zapata. Al final, se ejecuta el método PredesignPedestal que determina las dimensiones necesarias del pedestal para soportar la carga Pu. Y luego se procede a diseñar la zapata llamando al método Design de la clase FoundationCalculator.

```
var distance = jointforce.distance;
Trace.WriteLine(F3point1 + " " + F3point2 + " " + distance);
FoundationCalculator foundation = new FoundationCalculator();
foundation.Ps = Math.Abs(F3point1 + F3point2);
foundation.Msy = Math.Abs(Math.Abs(F3point1 * distance / 2) + Math.Abs(F3point2 * distance / 2));
foundation.Msx = Math.Abs(M1point1 + M1point2);
foundation.Pu = Math.Abs(Fu3point1 + Fu3point2);
foundation.Muy = Math.Abs(Math.Abs(Fu3point1 * distance / 2) + Math.Abs(Fu3point2 * distance / 2));
foundation.Mux = Math.Abs(Mu1point1 + Mu1point2); ;
foundation.pedestalB = Math.Max(b1 + 0.1, 0.3);
foundation.pedestalA = Math.Max(CA / 1000 + 0.1, 0.3);
foundation.pedestalH = 0.75;
foundation.pedestalXCenterPos = jointforce.CenterPoint;
numberofFoundations += 1;
foundation.cimName = "CIM-" + Convert.ToString(numberofFoundations);
foundation.pedestalName = "Pedestal-" + Convert.ToString(numberofFoundations);
foundation.jointForces1 = jointforce.jointForces1;
foundation.jointForces2 = jointforce.jointForces2;
foundation.distance = distance;
foundation.PredesignPedestal();
foundation.Design();
foundationsList.Add(foundation);
```

Figura 3.32 Definición de parámetros previo al diseño de la zapata.

Fuente: El autor

El diseño de la zapata esta divida en dos partes, en la figura 3.33 se muestra el método principal de la clase, Design. Este método primero realiza el diseño

geotécnico de la cimentación mediante el método DesignFoundationDimensions con el cual se establecen las dimensiones de ancho y largo de la zapata. Y luego realiza el diseño estructural con el método CalculateStructuralDesign.

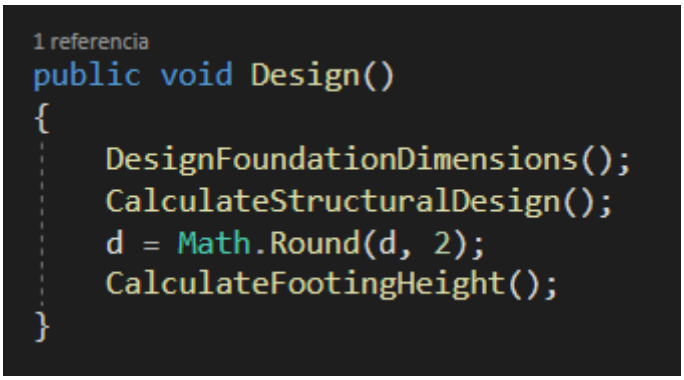


Figura 3.33 Método de diseño de la zapata aislada

Fuente: El autor

3.1.5.1 Dimensionamiento de la zapata

Se determinan las dimensiones de la zapata mediante el método DesignFoundationDimensions. La figura 3.34 muestra los subprocesos que incluye este método. Primero se realiza un prediseño de la zapata. Luego se verifica que la excentricidad de la carga se encuentre dentro del núcleo central para así asegurar que los esfuerzos en el suelo sean siempre de compresión.

Luego se ejecuta el método CalculateEffectiveStress que calcula el esfuerzo efectivo sobre el suelo. Dependiendo del caso, se redimensiona la cimentación con el método RecalculateCimDimensions. Una vez calculadas las dimensiones por capacidad de carga, se verifica los asentamientos y de ser necesario se redimensiona la cimentación.

```
2 referencias
public void DesignFoundationDimensions()
{
    Predesign();
    VerifyExcentricity();
    CalculateEffectiveStress();
    RecalculateCimDimensions();
    SettlementCheck();
    DefineFinalFoundationDim();
}
```

Figura 3.34 Método principal de dimensionamiento de la zapata

Fuente: El autor

Para determinar las dimensiones de la cimentación se divide la carga de servicio Ps para la capacidad de carga admisible qadm. Con el valor del área de la zapata, se calcula las dimensiones del ancho (FootingA) y largo (FootingB) tal que los volados en ambos sentidos tengan longitudes similares.

```
1 referencia
public void Predesign()
{
    PredesignCimArea = (Ps / qadm) * 1.45;

    double p = 4;
    double q = 2 * pedestalA + 2 * pedestalB;
    double r = pedestalA * pedestalB - PredesignCimArea;

    double x = (-q + Math.Sqrt(Math.Pow(q, 2) - 4 * p * r)) / (2 * p);

    FootingB = pedestalB + 2 * x;
    FootingA = FootingB + Math.Abs(pedestalB - pedestalA);

    Trace.WriteLine("init dim" + FootingA + " " + FootingB);

    CimArea = FootingA * FootingB;
}
```

Figura 3.35 Predimensionamiento de la zapata

Fuente: El autor

Ahora se procede a verificar que la carga este dentro del núcleo central de la zapata y así asegurar que los esfuerzos en el suelo sean siempre de compresión. En el caso de una zapata medianera que presenta una excentricidad por defecto, esta se puede ver reducida por la excentricidad de la carga Ps causada por un momento Msy, como se muestra en la figura 3.36.

A pesar de que el momento de servicio Msx es muy pequeño, lo que implica una excentricidad muy pequeña en la otra dirección, la figura 3.37 muestra la comprobación de la excentricidad biaxial. En el caso de no cumplirse, se aumentan la dimensión de la zapata que presente una mayor excentricidad.

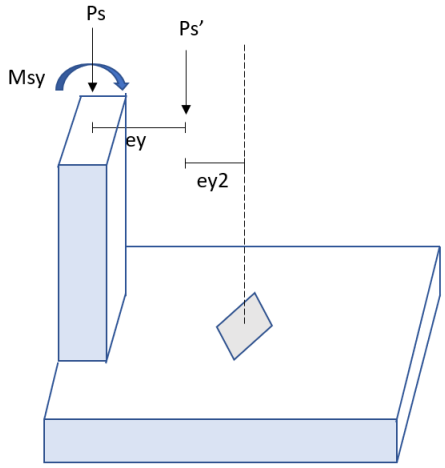


Figura 3.36 Excentricidad de la carga en zapata medianera

Fuente: El autor

```
1 referencia
public void VerifyExcentricity()
{
    double ex = Msx / Ps;
    double ey = Msy / Ps;
    double ey2 = Math.Abs(FootingB / 2 - pedestalB / 2 - Msy / Ps);
    double ratio = (6 * ex / FootingA) + (6 * ey2 / FootingB);

    while (ratio > 1)
    {
        if (ex / FootingA > ey / FootingB)
        {
            FootingA +=  0.05;
        }
        else
        {
            FootingB += 0.05;
            ey2 = Math.Abs(FootingB / 2 - pedestalB / 2 - Msy / Ps);
        }
        ratio = (6 * ex / FootingA) + (6 * ey2 / FootingB);
    }
    CimArea = FootingA * FootingB;
    FootingAefec = FootingA - 2 * ex;
    FootingBefec = FootingB - 2 * ex;
}
```

Figura 3.37 Verificación de la excentricidad biaxial

Fuente: El autor

La siguiente comprobación se realiza con la capacidad de carga admisible del suelo. Para esto se calcular el esfuerzo efectivo tomando en cuenta la carga de servicio Ps y el peso de la cimentación WF. La figura 3.38 muestra el cálculo del esfuerzo efectivo, donde al final se compara con la capacidad de carga admisible y, en caso de no cumplirse, se retorna el valor booleano igual a verdadero. Este valor condiciona el método RecalculateCimDimensions que aumenta las dimensiones de la zapata al no cumplirse la comprobación de capacidad de carga.

```
2 referencias
public void CalculateEffectiveStress()
{
    double ex = Msx / Ps;
    double ey = Msy / Ps;
    double ey2 = Math.Abs(FootingB / 2 - pedestalB / 2 - Msy / Ps);
    double WF = CalculateWF();
    qefec = ((Ps + WF) / CimArea) * (1 + 6 * ex / FootingA + 6 * ey2 / FootingB);
    if (qefec > qadm)
    {
        qadmNotPassed = true;
    }
    else
    {
        qadmNotPassed = false;
    }
}
```

Figura 3.38 Cálculo del esfuerzo efectivo sobre el suelo

Fuente: El autor

Debido a la necesidad de estudios de suelos para el cálculo de los asentamientos por consolidación. Con el fin de mantener el programa suficientemente sencillo para ser utilizado por maestros de obra, se asume un tipo de suelo rígido y solo se calcula los asentamientos elásticos.

Los parámetros elásticos utilizados corresponden a una arena densa de la figura 3.39. Para una cimentación poco profunda, Das & González, 2015 propone un proceso de cálculo de los asentamientos elásticos asumiendo que la cimentación es perfectamente flexible.

Tabla 17.6 Parámetros elásticos para varios tipos de suelo

Tipo de suelo	Módulo de elasticidad, E_s (MN/m_2)	Coeficiente de Poisson, μ_s
Arena suelta	10–25	0.20–0.40
Arena semi-densa	15–30	0.25–0.40
Arena densa	35–55	0.30–0.45
Arena limosa	10–20	0.20–0.40
Arena y grava	70–170	0.15–0.35
Arcilla blanda	4–20	
Arcilla media	20–40	0.20–0.50
Arcilla dura	40–100	

Figura 3.39 Parámetros elásticos para varios tipos de suelo

Fuente: (Das & González, 2015)

```
public void CalculateElasticSettlement()
{
    //Asumimos un suelo rigido
    //Tipo de suelo: Arena Densa

    double E = 4500; //[ton/m2]
    double u = 0.3;
    double If = CalculateIf();

    double B = FootingA;
    double L = FootingB;
    double alpha = 4;
    double m = L / B;
    H = 5 * B;
    double n = H / (B / 2);
    double m2 = Math.Pow(m,2);
    double n2 = Math.Pow(n, 2);
    double A1 = Math.Log( (m + Math.Sqrt(m2 + 1)) * (Math.Sqrt(1 + n2)) /(m + Math.Sqrt(m2 + n2 + 1)));
    double A2 = m / (n * Math.Sqrt(m2 + n2 + 1));
    double A0 = m * Math.Log( (1 + Math.Sqrt(m2 + 1)) * (Math.Sqrt(m2 + n2)) / (m +  Math.Sqrt(m2 + n2 + 1)) );
    double F2 = (n / (2 * Math.PI)) * Math.Atan(A2);
    double F1 = (1 / Math.PI) * (A0 + A1);
    double Is = F1 + (1 - 2 * u) * F2 / (1 - u);
    double SettlementCtr = 1000 * qefec * (alpha * B / 2) * (1 - Math.Pow(u, 2)) * Is * If / E;
    alpha = 1;
    m = L / B;
    n = H / B;
    double SettlementEsq = 1000 * qefec * (alpha * B) * (1 - Math.Pow(u, 2)) * Is * If / E;
    Settlement = Math.Max(SettlementCtr, SettlementCtr);
}
```

Figura 3.40 Cálculo de asentamientos elásticos

Fuente: El autor

Finalmente, se verifica que los asentamientos no superen el máximo de 10 cm propuesto por la norma NEC-SE-GC. De no cumplirse la verificación, se llama al método ChangeFootingAorBdim que aumenta las dimensiones de la zapata, como se muestra en la figura 3.41.

```
1 referencia
public void VerifySettlement()
{
    if (Settlement > 100)
    {
        SettlementPassed = false;
        ChangeFootingAorBdim();
    }
    else
    {
        SettlementPassed = true;
    }
}
```

Figura 3.41 Verificación de asentamientos

Fuente: El autor

3.1.5.2 Diseño estructural de la zapata

El diseño estructural se lo dividió en tres partes en las que se verifica el cortante, el punzonamiento, el aplastamiento y la flexión. Por último, se calcula las cantidades de varillas del armado a flexión.

La figura 3.42 muestra el método StructuralDesignPart1 en el cual se realiza el cálculo y verificación del cortante y el punzonamiento. Los métodos CalculateShearStresses y CalculatePunching, que detallan este proceso, se encuentran en el anexo 7.

```
3 referencias
public void StructuralDesignPart1()
{
    //Revisar cortante
    bool shearPassed = false;
    while (!shearPassed)
    {
        shearPassed = CalculateShearStresses();
    }

    //Revisar punzonamiento
    bool punchingPassed = false;
    while (!punchingPassed)
    {
        punchingPassed = CalculatePunching();
    }
}
```

Figura 3.42 Diseño estructural de la zapata Parte 1

Fuente: El autor

Al verificar el aplastamiento, como el concreto de la zapata es el mismo del pedestal, por lo general sí se cumple este criterio. En caso de no cumplirse, se puede solucionar añadiendo un dado que aumente el área que resiste este esfuerzo o directamente aumentar el área del pedestal.

```
1 referencia
public bool CalculateFlattening()
{
    double flatteningStress = Pu / (pedestalA * pedestalB);
    double ResistantStress = 0.85 * fc * 0.65 * 10;

    bool flatteningPassed = false;
    if (ResistantStress > flatteningStress)
    {
        flatteningPassed = true;
    }
    else
    {
        pedestalA += 0.05;
        pedestalB += 0.05;
        DesignFoundationDimensions();
        StructuralDesignPart1();
    }
    return flatteningPassed;
}
```

Figura 3.43 Diseño estructural de la zapata Parte 2

Fuente: El autor

La figura 3.43 muestra el cálculo del aplastamiento y su verificación. En caso de no cumplir con el esfuerzo resistente, se aumenta las dimensiones del pedestal y se repite el proceso dimensionamiento de la zapata y diseño estructural que dependían de estos valores.

Como última revisión, se verifica la flexión calculando el peralte efectivo mínimo necesario para aguantar el momento generado por las cargas mayoradas. En caso de no cumplir la revisión, se iguala el peralte actual al mínimo y se recalcula las partes 1 y 2 del diseño estructural (Figura 3.44).

```
1 referencia
public bool CalculateFlexion()
{
    double Ma = Mux;
    double Mb = Muy;
    double q1a = Pu / CimArea + 6 * Ma / (FootingB * Math.Pow(FootingA, 2));
    double q2a = Pu / CimArea - 6 * Ma / (FootingB * Math.Pow(FootingA, 2));
    double q1b = Pu / CimArea + 6 * Mb / (FootingA * Math.Pow(FootingB, 2));
    double q2b = Pu / CimArea - 6 * Mb / (FootingA * Math.Pow(FootingB, 2));
    double volB = FootingB - pedestalB;
    double volA = (FootingA - pedestalA) / 2;
    double q3a = ((q1a - q2a) * (FootingA - volA)) / FootingA + q2a;
    double q3b = ((q1b - q2b) * (FootingB - volB)) / FootingB + q2b;
    double M = (Math.Pow(Math.Max(volA, volB), 2) / 6) * (Math.Max(q3a, q3b) + 2 * Math.Max(q1a, q1b));
    double locald = 0.83 * Math.Sqrt(M / fc);

    bool dPassed = false;
    if (locald <= d)
    {
        dPassed = true;
    }
    else
    {
        d = locald;
        StructuralDesignPart1();
        StructuralDesignPart2();
    }
    return dPassed;
}
```

Figura 3.44 Diseño estructural de la zapata Parte 3

Fuente: El autor

Para finalizar el diseño estructural, se calculan los aceros y su separación. Este proceso se encuentra en el anexo 8. A continuación se presenta un ejemplo de cálculo manual para la cimentación CIM-1 generada por el programa. Esta posee dimensiones de 1.5x4.45m y una altura de 36 cm.

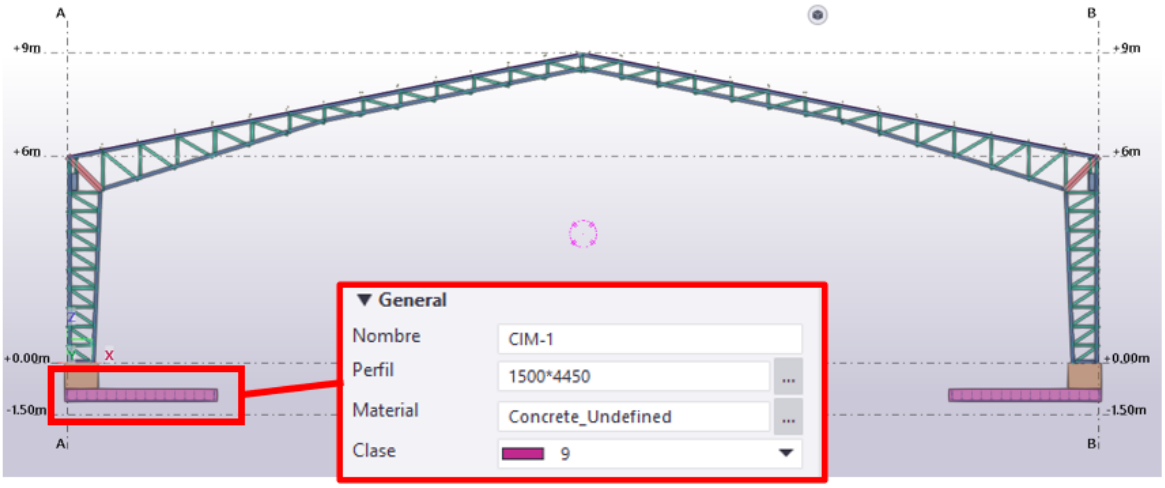


Figura 3.45 Zapata aislada modelada por código

Fuente: El autor

Prediseño de la cimentación:

$Df = 1m$

$Nf = 1m$

$gammaConcr = 2.4ton/m3$

$gammaAgua = 1ton/m3$

$Ps = 6.88\ ton$

$qadm = 20\ ton/m2$

$CimArea = (Ps/qadm) * 1.45\ = 0.5m2$

$p = 4$

$q = 2 * pedestalA + 2 * pedestalB = 2 *\ 0.5 + 2 * 1 = 3$

$r = pedestalA * pedestalB - CimArea = 0.5 * 1 - 0.5 = 0$

$x = (-q + \sqrt{q^2 - 4 * p * r})/(2 * p) = 0$

$B = pedestalB + 2 * x = 1 + 2 * 0 = 1$

$A = B + (pedestalB - pedestalA) = 1 + (1 - 0.5) = 1.5$

$CimArea = 1 * 1.5 = 1.5m2$

Verificación de excentricidad:

$Msx = 0ton * m$

$ex = Msx/Ps = 0m$

$Msy = 16.96ton * m$

$ey = Msy/Ps = 2.46$

Tras varias iteraciones se definió una zapata de 1.5x4.45m, con esto se tiene:

$ey2 = B/2 - pedestalB/2 - ey = 4.45/2 - 1/2 - 2.46 = -0.735$

$ratio = 6 * ex/A + 6 * ey/B = 0 + 6 * 0.735/4.45 = 0.99$

Cálculo del esfuerzo efectivo:

$qefec = ((Ps + Wf)/CimArea) * (1 + 6 * ex/A + 6 * ey/B)$

$Wf = Df * gammaConcr * A * B - (Df - Nf) * gammaAgua = 16.2ton$

$qefec = ((6.88 + 16.2)/6.675) * (1 + 0 + 0.99) = 6.88ton/m2$

$qefec = 6.88ton/m2 < aqdm$

Cálculo del asentamiento elástico:

$Es = 4500ton/m2$

$u = 0.3$

$Df/A = 0.66, \quad A/B = 0.33 \quad \rightarrow \quad If = 0.71$

$alpha = 4$

$m = B/A = 2.96$

$H = 5 * A = 7.5m$

$n = H/(A/2) = 10$

$m2 = m^2 = 8.76$

$n2 = n^2 = 100$

$$A0 = m * ln\frac{\left(1 + \sqrt{m2 + 1}\right) * \sqrt{m2 + n2}}{m * \left(1 + \sqrt{m2 + n2 + 1}\right)} = 0.698$$

$$A1 = ln\frac{\left(m + \sqrt{m2 + 1}\right) * \sqrt{1 + n2}}{m + \sqrt{m2 + n2 + 1}} = 1.515$$

$$A2 = \frac{m}{n * \sqrt{m2 + n2 + 1}} = 0.028$$

$$F1 = \frac{1}{\pi} * (A0 + A1) = 0.704$$

$$F2 = \frac{n}{2\pi} * \tan^{-1} A2 = 0.045$$

$$Is = F1 + \frac{1-2u}{1-u} * F2 = 0.73$$

$$Se = 1000 * qefec * \left(alpha * \frac{A}{2}\right) * \frac{1-u^2}{Es} * Is * If = 2mm$$

3.1.6 Diseño de las placas base

Los métodos utilizados para diseñar la placa base se encuentran en el anexo 9. El método DesignBasePlate comienza calculando la carga máxima sobre la base, luego plantea las dimensiones iniciales de la placa y calcula la resistencia al contacto del concreto. Dependiendo del caso, se incrementan las dimensiones del pedestal o de la placa base, y se concluye calculando el espesor necesario.

3.1.7 Diseño de la soldadura

Los miembros de acero utilizan las soldaduras precalificadas presentes en el manual de la AWS D1.1 capítulo 3. El anexo 10 contiene la especificación de la soldadura para las conexiones entre cordones y para la conexión entre la columna y la placa base. La conexión viga y columna usa la soldadura de junta a tope (Anexo 11) y para la conexión de los ángulos a los cordones de las cerchas se utiliza soldadura de canal bisel doble (Anexo 12).

3.2 Especificaciones técnicas

Preliminares

Rubro 1: Trazado, replanteo y nivelación

Descripción:

Este rubro corresponde al trabajo de topografía requerido al comienzo de la obra, esto incluye verificar las dimensiones del terreno y establecer los niveles de cota para los trabajos subsecuentes. El proceso de trazado y replanteo consiste en establecer las referencias, las cuales debe permanecer fijas durante la ejecución de la actividad. A partir de las cotas de los planos se determinará la profundidad y localización del relleno o excavación dependiendo de la topografía del lugar. Este trabajo se debe realizar en conjunto de aparatos con buena precisión.

Materiales:

Aunque se recomienda utilizar estacas de metal con dimensiones de 20 x 4 cm, es posible utilizar estacas de madera en su lugar.

Equipo por utilizar:

Se necesita equipo topográfico llamado Nivel para realizar la nivelación del terreno, o estación total si se requiere un grado de tolerancia más preciso.

Mano de obra:

Se requiere un profesional en topografía con certificado de la Senecyt con calificación C1 y un cadenero D2 que sirva como ayudante del topógrafo.

Unidad de medida: m2

Preliminares

Rubro 2: Caseta de materiales y guardianía

Descripción:

Es una construcción provisional que brinda seguridad al personal durante la ejecución de la obra. En este rubro se incluye la construcción del campamento con diferentes sitios de trabajo como bodega de materiales, primeros auxilios, etc. Dependiendo de la dimensión de la obra se amoblará más o menos el campamento y la provisión de equipamiento también va acorde al tamaño de la obra. El área de la caseta va en proporción del 1% al área total del terreno.

Materiales:

Se requiere cuartones, tablas, tiras de encofrado, clavos de 2”, planchas de plywood de 9mm, plancha de zinc, bisagras, candado, picaporte y argollas.

Equipo por utilizar:

Se utiliza equipo manual que incluye martillo, regla, taladro y varios más.

Mano de obra:

Se necesita mano de obra mínima, esto incluye un peón P2, carpintero D2 y un maestro de obra C1.

Unidad de medida: m2

Preliminares

Rubro 3: Rótulo de obra y señalética de obra

Descripción:

Este rubro consiste en colocar obligatoriamente una señalética que prevenga a los peatones de los riesgos existentes en el área de trabajo. El rótulo de obra debe proporcionar la información relevante para la ejecución de la obra.

Materiales:

Se utiliza un rótulo en lona de gigantografía en concordancia al formato establecido por el municipio.

Equipo por utilizar:

Se requiere únicamente herramientas manuales.

Unidad de medida: U

Preliminares

Rubro 4: Cerramiento provisional

Descripción:

Con el fin de garantizar la continuidad del trabajo sin interferencias por parte de personas externas a la obra, se coloca un cerramiento en el perímetro de la construcción que permite controlar al personal en el interior y exterior del terreno.

Materiales:

Se debe utilizar cañas, clavos de 2 ½”, lona plástica y alambre recocido #18.

Equipo por utilizar:

La mano de obra solo utilizará herramientas manuales para esta actividad.

Mano de obra:

Maestro de obra C1, peón E2, Carpintero D2.

Unidad de medida: m2

Movimiento de tierra

Rubro 5: Excavación y desalojo a máquina para cimentaciones

Descripción:

Este rubro consiste en excavar a máquina la cantidad necesaria para llegar a la cota determinada en los planos. Queda a disposición del fiscalizador de la obra determinar si el suelo excavado es aceptable como relleno.

Equipo por utilizar:

Volqueta, bomba de 4", retroexcavadora y herramientas manuales.

Mano de obra especializada:

Maestro de obra, chofer profesional, operador de maquinaria, peón.

Unidad de medida: m2

Movimiento de tierra

Rubro 6: Relleno con material importado

Descripción:

Consiste en rellenar lo excavado tras fundir la cimentación hasta llegar a una cota determinada. Queda a disposición del fiscalizador determinar si el suelo es aceptable para ser usado como relleno.

Materiales:

Ensayo Proctor, agua y material importado.

Equipo por utilizar:

Retroexcavadora, compactador semipesado, herramienta manual y una bomba de 4".

Mano de obra:

Maestro de obra, peón, operador de equipo, chofer profesional.

Unidad de medida: m2

Estructura

Rubro 7: Replantillo de hormigón de 140Kg/cm2 e=5cm

Descripción:

Consiste en la colocación de un hormigón de baja resistencia como soporte de los elementos estructurales. La superficie del suelo sobre el cual se coloca el

Replantillo debe estar libre de basura, de agua, y nivelado de acuerdo con las cotas de los planos. Para evitar la disgregación de material granular, el vertido del hormigón debe realizare a una altura máxima de 1.5m.

Materiales:

Cemento tipo I, agua, arena y piedra para la elaboración del hormigón. No se requiere de encofrado, puesto que el propio suelo hace la función de dar forma y sustento al hormigón.

Equipo por utilizar:

Concretera y herramientas manuales.

Mano de obra:

Maestro de obra, peón y albañil.

Unidad de medida: m2

Estructura

Rubro 8: Plintos de hormigón armado f'c=210kg/cm2

Descripción:

Los plintos se elaborarán con hormigón de resistencia 210 kg/cm2 y acero de refuerzo de Fy=4200 kg/cm2. Para la correcta colocación de los plintos, se debe verificar que el encofrado se encuentre estable, limpio y húmedo.

Materiales:

Cemento tipo I, agua, arena y piedra para la elaboración del hormigón. También se usará acero de refuerzo.

Equipo por utilizar:

Concretera y herramientas manuales.

Mano de obra:

Maestro de obra, peón y albañil.

Unidad de medida: m3

Estructura

Rubro 9: Acero de refuerzo Fy=4200kg/cm2

Descripción:

Este rubro incluye el acero que es colocado en los plintos y en los pedestales que soportan las placas base. Para proteger el acero contra la oxidación, se debe mantener el material sobre alguna plataforma evitando el contacto con el suelo. Se debe asegurar que el acero se encuentra limpio, libre de cualquier sustancia o suciedad. El doblado de las varillas debe ir de acuerdo con las especificaciones en los planos estructurales. Antes de la colocación del hormigón, se debe asegurar la correcta posición del refuerzo de acuerdo con los planos ayudándose de alambres en las intersecciones.

Materiales:

Acero de refuerzo, alambre recocido #18.

Equipo por utilizar:

Cortadora, dobladora y herramientas manuales.

Mano de obra:

Fierrero, maestro de obra y peón.

Unidad de medida: Kg

Estructura

Rubro 10: Contrapiso de hormigón f'c=180kg/cm2 e=10cm

Descripción:

Consiste en la colocación del hormigón de resistencia 180 kg/cm2 como contrapiso. Previo a colocar el hormigón se debe asegurar que el suelo se encuentra correctamente compactado y con una superficie limpia. Además, tras colocar el hormigón se deben asegurar las juntas para controlar la dilatación del hormigón.

Materiales:

Cemento tipo Portland, piedra 3/4", arena, agua, tira de encofrado, clavo 2"x8.

Equipo por utilizar:

Concretera, vibrador y herramientas manuales.

Mano de obra:

Maestro de obra, peón, albañil, fierrero y carpintero.

Unidad de medida: m2

Estructura

Rubro 11: Placas de anclaje

Descripción:

Este rubro comprende el suministro e instalación de las placas de anclaje. El contratista deberá fabricar e instalar las placas de acero, esto incluye los trabajos necesarios para la instalación completa de las placas en la posición donde irán las columnas.

Materiales:

Acero estructural A36, acero de refuerzo en barra, electrodo 6011, electrodo 7011 y tanque de CO2.

Equipo por utilizar:

Herramienta manual, cortadora-dobladora y soldadoras eléctricas de 110V y 220V.

Mano de obra:

Fierrero, peón E2, soldador D2 y maestro de obra C1.

Unidad de medida: kg

Estructura

Rubro 12: Suministro de acero estructural

Descripción:

Consiste en la provisión del acero estructural A36 por parte del contratista.

Materiales:

Acero en perfil.

Equipo por utilizar:

Herramienta manual y carro grúa

Mano de obra:

Operador carro grúa, peón, albañil y maestro de obra.

Unidad de medida: kg

Estructura

Rubro 13: Fabricación de cerchas

Descripción:

Incluye la elaboración en taller de las cerchas de la estructura. La mano de obra será la mejor calificada de acuerdo con las prácticas usadas en los talleres de estructuras de acero. Los cortes deberán ser realizados con soplete de manera precisa al igual que el cizallamiento y el martilleo. Se debe redondear cualquier filo producto de un corte en la elaboración utilizando esmeril.

Materiales:

Electrodo revestido E6011.

Equipo por utilizar:

Herramienta manual y soldadora eléctrica 300 A.

Mano de obra:

Maestro soldador especializado, peón e inspector de soldadura.

Unidad de medida: kg

Estructura

Rubro 14: Montaje de la estructura metálica

Descripción:

Consiste en el transporte de las cerchas fabricadas en taller y su montaje en obra. El montaje se realiza con grúas o andamios dependiendo del tamaño del galpón, cualquier diferencia se toma en cuenta como costo indirecto. El inspector de soldadura deberá asegurar que las soldaduras realizadas en el montaje cumplen con las normas establecidas en los planos.

Materiales:

Electrodo revestido E6010.

Equipo por utilizar:

Herramienta manual y grúas/andamios.

Mano de obra:

Soldador E2, peón E2, inspector de soldadura y maestro de obra.

Unidad de medida: kg

Estructura

Rubro 15: Pintado de estructura con sintético automotriz

Descripción:

Este rubro comprende pintar la estructura de sintética automotriz que protegerá al acero de agentes agresivos. Se alquilará por día una cesta elevadora de brazo articulado, aunque también se puede utilizar andamios siempre y cuando el rendimiento sea equivalente para un cierto tamaño de estructura. La aplicación de la pintura deberá respetar el tiempo de secado especificado por el fabricante.

Materiales:

Imprimación de secado rápido y pintura sintética automotriz.

Equipo por utilizar:

Herramienta manual y cesta elevadora de brazo articulado.

Mano de obra:

Pintor y ayudante de pintor.

Unidad de medida: kg

Estructura

Rubro 16: Provisión e instalación de la cubierta

Descripción:

Este rubro comprende el transporte de la cubierta galvalume e=0.3mm y su respectiva instalación que incluye las actividades de corte de cubierta y soldar los pernos autoperforantes.

Materiales:

Estil panel/techos galvalume AR-2000 e=0.3mm, electrodo revestido E6010, pernos autoperforantes.

Equipo por utilizar:

Herramienta manual, andamios, cortadora de disco y motosoldadora.

Mano de obra:

Peón E2, soldador en construcción y maestro de obra C1.

Unidad de medida: M2

Estructura

Rubro 16: Provisión e instalación del cumbrero

Descripción:

Este rubro comprende el transporte del cumbrero tipo estilpanel y su respectiva instalación. La instalación se realizará en los sitios especificados en los planos con el fin de cubrir y proteger la edificación de los cambios del tiempo.

Materiales:

Cumbrero tipo Estilpanel, perno punta de broca 10x3.

Equipo por utilizar:

Herramienta manual.

Mano de obra:

Peón E2, albañil y maestro de obra C1.

Unidad de medida: M

Varios

Rubro 21: Limpieza y desalojo progresivo de escombros en obra

Descripción:

Este rubro comprende la recogida de los escombros generados en la obra a lo largo del proyecto. Para evitar acumulación, la limpieza será progresiva durante toda la duración de la obra y se los transportará a una escombrera autorizada.

Materiales:

Sacos de yute.

Equipo por utilizar:

Herramienta manual y camión.

Mano de obra:

Peón y chofer.

Unidad de medida: U

CAPÍTULO 4

4. PRESUPUESTO

4.1 Descripción de rubros

A continuación, se presenta la lista de los rubros relacionados a las actividades y materiales requeridos para la construcción de un galpón.

#	DETALLES DE RUBROS
A	**PRELIMINARES**
1	TRAZADO, REPLANTEO Y NIVELACION
2	CASETA DE MATERIALES Y GUARDIANIA
3	ROTULO DE OBRA Y SEÑALETICA DE OBRA
4	CERRAMIENTO PROVISIONAL h=2.40
	SUBTOTAL
B	**MOVIMIENTO DE TIERRAS**
5	EXCAVACION Y DESALOJO A MAQUINA PARA CIMENTACIONES
6	RELLENO CON MATERIAL IMPORTADO
	SUBTOTAL
C	**ESTRUCTURA**
7	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM
8	HORMIGÓN ARMADO PARA PLINTOS (f´c=210 kg/cm²)
9	ACERO DE REFUERZO f'y=4200kg/cm2
10	CONTRAPISO DE HORMIGON 180 kg/cm2 E=10cm
11	PLACAS DE ANCLAJE
12	SUMINISTRO DE ACERO ESTRUCTURAL
13	FABRICACIÓN DE CERCHAS
14	MONTAJE DE LA ESTRUCTURA DE ACERO
15	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ
16	PROVISIÓN E INSTALACIÓN DE LA CUBIERTA
17	PROVISIÓN E INSTALACIÓN DEL CUMBRERO
	SUBTOTAL
D	**ALBAÑILERIA**
18	MAMPOSTERIA DE BLOQUE
19	ENLUCIDO DE PAREDES
	SUBTOTAL
E	**PINTURA EN MAMPOSTERIA**
20	EMPASTE, PINTURA BLANCA INTERIOR
	SUBTOTAL
F	**VARIOS**
21	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA.

Figura 4.1 Listado de rubros

Fuente: El autor

4.2 Análisis de costos unitarios

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	1			UNIDAD	M2
DETALLE	TRAZADO,REPLANTEO Y NIVELACION				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,02
NIVEL	1,00	1,25	1,25	0,010	0,01
ESTACION TOTAL	1,00	6,25	6,25	0,010	0,06
SUBTOTAL M					0,09
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
TOPOGRAFO 2 C1	1,00	4,04	4,04	0,010	0,04
CADENERO D2	2,00	3,65	7,30	0,010	0,07
SUBTOTAL N					0,11
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
ESTACAS		U	2,00	0,10	0,20
PINTURA		LT	0,04	3,60	0,14
CAL		KG	0,20	0,10	0,02
SUBTOTAL O					0,36
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					0,56
INDIRECTOS Y UTILIDADES %25					0,14
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					0,70
VALOR OFERTADO					0,70

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M2
DETALLE	CASETA DE MATERIALES Y GUARDIANIA		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,84
SUBTOTAL M					0,84

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Maestro de Obra C1	1,00	4,06	4,06	0,50	2,03
Carpintero D2	3,00	3,66	10,98	0,50	5,49
Peon E2	5,00	3,62	18,10	0,50	9,05
SUBTOTAL N					16,57

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Caña picada	U	1,67	1,75	2,92
Caña rolliza	U	1,35	1,50	2,03
Clavos de 2 1/2"	Kg.	0,25	2,25	0,56
Tabla de encofrado semidura	U	0,68	3,50	2,38
Cuartón de encofrado semiduro	U	0,25	2,00	0,50
Tira de encofrado semidura	U	0,16	1,50	0,24
Soga	U	0,08	0,15	0,01
Planchas de zinc	U	0,45	6,00	2,70
Bisagra cromada de 3,5" x 3,5" in	U	0,17	0,45	0,08
Candado	U	0,10	5,00	0,50
Inodoro	U	0,04	28,00	1,12
Argollas	U	0,17	0,45	0,08
SUBTOTAL O				13,12

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	30,53
INDIRECTOS Y UTILIDADES %25	7,63
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	38,16
VALOR OFERTADO	38,16

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	U
DETALLE	ROTULO DE OBRA Y SEÑALETICA DE OBRA				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,01
SUBTOTAL M					0,01
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
PEON E2	1,00	3,62	3,62	0,037	0,13
SUBTOTAL N					0,13
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B		COSTO C=A*B
ROTULO PUBLICITARIO	U	1,00	48,00		48,00
SUBTOTAL O					48,00
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B		COSTO C=A*B
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					48,14
INDIRECTOS Y UTILIDADES %25					12,03
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					60,17
VALOR OFERTADO					60,17

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M
DETALLE	CERRAMIENTO PROVISIONAL h=2.40		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,10
SUBTOTAL M					0,10

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Maestro de Obra C1	0,25	4,06	1,02	0,110	0,11
Carpintero D2	1,00	3,67	3,67	0,110	0,40
Peon E2	3,00	3,62	10,86	0,110	1,19
SUBTOTAL N					1,70

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Cañas	U	0,30	2,00	0,60
Alambre recocido # 18	KG	0,01	1,67	0,02
Clavos de 2 1/2"	KG	0,02	2,00	0,04
Lona plastica	M	1,10	3,00	3,30
SUBTOTAL O				3,96

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	5,76
INDIRECTOS Y UTILIDADES %25	1,44
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	7,20
VALOR OFERTADO	7,20

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2		UNIDAD	M2	
DETALLE	EXCAVACION Y DESALOJO A MAQUINA PARA AREA TOTAL DEL TERRENO H = 50				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,14
Retroexcavadora	1,00	30,00	30,00	0,150	4,50
Volqueta	1,00	20,00	20,00	0,150	3,00
Bomba de 4"	0,50	3,75	1,88	0,150	0,28
SUBTOTAL M					7,92
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Maestro de obra C1	0,25	4,06	1,02	0,15	0,15
Chofer Profesional, licencia tipo E C1	1,00	5,31	5,31	0,15	0,80
Operador de equipo C1	1,00	4,06	4,06	0,15	0,61
Peon E2	2,00	3,62	7,24	0,15	1,09
SUBTOTAL N					2,65
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
SUBTOTAL O					0,00
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					10,57
INDIRECTOS Y UTILIDADES %25					2,64
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					13,21
VALOR OFERTADO					13,21

ANALISIS DE PRECIOS UNITARIOS						
OBRA:	Diseño parametrico la estructura de una nave industrial					
RUBRO:	2			UNIDAD	M2	
DETALLE	RELLENO CON MATERIAL IMPORTADO					
EQUIPOS						
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC	
HERRAM. MANUAL		0,05				0,05
Retroexcavadora	1,00	30,00	30,00	0,055		1,65
Compactador semipesado	1,00	4,50	4,50	0,055		0,25
Bomba de 4"	0,50	3,75	1,88	0,055		0,10
SUBTOTAL M						2,05
MANO DE OBRA						
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR	
Maestro de obra C1	0,25	4,06	1,02	0,06		0,06
Peon E2	1,00	3,62	3,62	0,06		0,20
Operador de equipo C1	1,00	4,06	4,06	0,06		0,22
Chofer profesiOnal C1	1,00	5,31	5,31	0,06		0,29
SUBTOTAL N						0,77
MATERIALES						
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B		
Agua	m3	0,05	1,85			0,09
Material de prestamo importado	m3	1,25	4,00			5,00
Pruebas de suelo + (proctor)	u	0,00	37,00			0,04
SUBTOTAL O						5,13
TRANSPORTE						
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B		
Incluye transporte						
SUBTOTAL P						0,00
TOTAL COSTO DIRECTO (M+N+O+P)						7,95
INDIRECTOS Y UTILIDADES %25						1,99
OTROS INDIRECTOS %						
COSTO TOTAL DEL RUBRO						9,94
VALOR OFERTADO						9,94

ANALISIS DE PRECIOS UNITARIOS						
OBRA:	Diseño parametrico la estructura de una nave industrial					
RUBRO:	2			UNIDAD	M2	
DETALLE	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM					
EQUIPOS						
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC	
HERRAM. MANUAL		0,05				0,15
Concretera de 1 Saco	1,00	3,50	3,50	0,147		0,51
SUBTOTAL M						0,66
MANO DE OBRA						
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR	
Estruc. Ocup. E2 (Peón)	4,00	3,62	14,48	0,15		2,13
Estruc. Ocup. C1 (Maestro Mayor y El	0,25	4,06	1,02	0,15		0,15
(Albañil,Carpin.,Caden.,Elec.,O.E)	1,00	3,66	3,66	0,15		0,54
SUBTOTAL N						2,82
MATERIALES						
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B		
Cemento Tipo I (en Obra)	kg	18,00	0,16		2,88	
Arena para Hormigón	m^3	0,04	13,00		0,52	
Materíal de Piedra T.	m^3	0,06	14,00		0,84	
Agua	m^3	0,03	1,85		0,06	
Encofrado Metalico	m^2	0,25	1,50		0,38	
SUBTOTAL O					4,68	
TRANSPORTE						
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B		
Incluye transporte						
SUBTOTAL P					0,00	
TOTAL COSTO DIRECTO (M+N+O+P)					8,16	
INDIRECTOS Y UTILIDADES %25					2,04	
OTROS INDIRECTOS %						
COSTO TOTAL DEL RUBRO					10,20	
VALOR OFERTADO					10,20	

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M2
DETALLE	PLINTOS DE HORMIGÓN ARMADO (f´c=210 kg/cm²)		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			3,36
Vibrador de manguera	1,00	4,00	4,00	2,000	8,00
SUBTOTAL M					11,36

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Maestro	0,20	4,06	0,81	2,00	1,62
Albañil	1,00	3,67	3,67	2,00	7,34
Carpintero	2,00	3,67	7,34	2,00	14,68
Peon	6,00	3,62	21,72	2,00	43,44
SUBTOTAL N					67,08

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Hormigon premezclado f´c=280kg/cm2	M3	1,02	120,00	122,40
Encofrado para estructuras	M2	4,00	6,00	24,00
SUBTOTAL O				146,40

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	224,84
INDIRECTOS Y UTILIDADES %25	56,21
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	281,05
VALOR OFERTADO	281,05

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	M2
DETALLE	ACERO DE REFUERZO f'y=4200kg/cm2				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,04
Cortadora-dobladora	1,00	0,50	0,50	0,0320	0,02
SUBTOTAL M					0,06
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
FIERRERO	1,00	5,20	5,20	0,032	0,17
PEON E2	2,00	3,54	7,08	0,032	0,23
MAESTRO C1	1,00	4,20	4,20	0,032	0,13
SUBTOTAL N					0,53
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Acero de refuerzo		kg	1,05	1,07	1,12
Alambre recocido #18		kg	0,03	1,50	0,05
SUBTOTAL O					1,17
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
Incluye transporte					
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					1,76
INDIRECTOS Y UTILIDADES %25					0,44
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					2,20
VALOR OFERTADO					2,20

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	M2
DETALLE	CONTRAPISO DE HORMIGON 180 kg/cm2				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,47
VIBRADOR DE MANGUERA	2,00	2,50	5,00	0,228	1,14
Concretera	1,00	3,13	3,13	0,228	0,71
SUBTOTAL M					2,32
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
MAESTRO DE OBRA MAYOR C1	1,00	4,06	4,06	0,228	0,93
ALBAñIL D2	1,00	3,67	3,67	0,228	0,84
CARPINTERO D2	2,00	3,67	7,34	0,228	1,67
FIERRERO D2	2,00	3,67	7,34	0,228	1,67
PEON E2	5,00	3,54	17,70	0,228	4,04
SUBTOTAL N					9,15
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
Cemento tipo Portland	kg	30,00	0,16	4,65	
Piedra 3/4"	M3	0,08	13,00	0,99	
Arena	M3	0,05	10,00	0,50	
Agua	M3	0,01	1,50	0,02	
Tira de encofrado	u	0,46	1,75	0,81	
Clavo 2"x8	Lb	0,01	1,13	0,01	
SUBTOTAL O				6,98	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
Incluye transporte					
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)				18,45	
INDIRECTOS Y UTILIDADES %25				4,61	
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO				23,06	
VALOR OFERTADO				23,06	

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M2
DETALLE	PLACA DE ANCLAJE		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,05
Cortadora - Dobladora	1,00	1,00	1,00	0,025	0,03
Electrosoldadora 110 V	1,00	5,00	5,00	0,025	0,13
Electrosoldadora 220V	1,00	8,00	8,00	0,025	0,20
SUBTOTAL M					0,41

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
MAESTRO DE OBRA C1	1,00	4,06	4,06	0,025	0,10
SOLDADOR D2	2,00	3,67	7,34	0,025	0,18
Fierrero	3,00	3,66	10,98	0,025	0,27
Peon	3,00	3,62	10,86	0,025	0,27
SUBTOTAL N					0,82

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Acero estructural A36	KG	0,59	1,10	0,65
Acero de refuerzo en barras	KG	0,41	1,00	0,41
Electrodo 6011	KG	0,02	4,62	0,09
Electrodo 7011	KG	0,01	5,12	0,05
Tanque de CO2	u	0,01	52,80	0,53
SUBTOTAL O				1,73

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	2,96
INDIRECTOS Y UTILIDADES %25	0,74
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	3,70
VALOR OFERTADO	3,70

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	M2
DETALLE	SUMINISTRO DE ACERO ESTRUCTURAL				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,01
CARRO GRUA	2,00	18,00	36,00	0,0002	0,01
SUBTOTAL M					0,02
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
OPERADOR CARRO GRUA C1	2,00	7,08	14,16	0,0002	0,0028
PEON E2	4,00	3,54	14,16	0,0002	0,0028
MAESTRO DE OBRA C1	2,00	6,87	13,74	0,0002	0,0027
SUBTOTAL N					0,0084
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Acero en perfil		KG	1,00	1,10	1,10
SUBTOTAL O					1,10
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
Incluye transporte					
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					1,12
INDIRECTOS Y UTILIDADES %25					0,28
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					1,40
VALOR OFERTADO					1,40

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	KG
DETALLE	FABRICACIÓN DE CERCHAS				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,04
Soldadora electrica 300 a	2,00	1,98	3,96	0,021	0,08
SUBTOTAL M					0,12
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Maestro soldador especializado	2,00	4,06	8,12	0,021	0,17
Peon	4,00	3,54	14,16	0,021	0,30
Inspector de soldadura	1,00	4,20	4,20	0,021	0,09
SUBTOTAL N					0,5600
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
Electrodo Revestido E6010	KG	0,30	4,53	1,36	
SUBTOTAL O				1,36	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)				2,04	
INDIRECTOS Y UTILIDADES %25				0,51	
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO				2,55	
VALOR OFERTADO				2,55	

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	KG
DETALLE	MONTAJE DE LA CARGA				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,05
Gruas / Andamios	1,00	19,00	19,00	0,021	0,40
SUBTOTAL M					0,45
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
SOLDADOR D2	2,00	5,20	10,40	0,021	0,22
PEON E2	4,00	3,54	14,16	0,021	0,30
INSPECTOR DE SOLDADURA	1,00	4,20	4,20	0,021	0,09
MAESTRO C1	1,00	6,87	6,87	0,021	0,14
SUBTOTAL N					0,75
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Electrodo Revestido E6010		KG	0,01	4,53	0,05
SUBTOTAL O					0,05
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					1,25
INDIRECTOS Y UTILIDADES %25					0,31
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					1,56
VALOR OFERTADO					1,56

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	KG
DETALLE	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/DIA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,01
Cesta elevadora de brazo articulado	1,00	73,77	73,77	0,001	0,07
SUBTOTAL M					0,08
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Pintor	2,00	5,20	10,40	0,001	0,01
Ayudante pintor	4,00	3,54	14,16	0,001	0,01
SUBTOTAL N					0,02
MATERIALES					
DESCRIPCION		UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Imprimación de secado rápido		GAL	0,02	6,17	0,12
Pintura sintética automotriz		GAL	0,01	20,46	0,20
SUBTOTAL O					0,32
TRANSPORTE					
DESCRIPCION		UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
SUBTOTAL P					0,00
TOTAL COSTO DIRECTO (M+N+O+P)					0,42
INDIRECTOS Y UTILIDADES %25					0,10
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					0,52
VALOR OFERTADO					0,52

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M2
DETALLE	PROVISiÓN E INSTALACIÓN DE LA CUBIERTA		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,04
Andamio	2,00	1,00	2,00	0,020	0,04
Cortadora de disco	2,00	2,50	5,00	0,020	0,10
Motosoldadora	4,00	10,00	40,00	0,020	0,80
SUBTOTAL M					0,98

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	4,00	3,62	14,48	0,020	0,29
Soldador en construcción	2,00	3,72	7,44	0,020	0,15
Maestro de obra C1	1,00	4,06	4,06	0,020	0,08
SUBTOTAL N					0,52

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Estilpanel/techos galvalume AR-2000 e=0.30mm	M2	1,00	8,00	8,00
Electrodo Revestido E6010	KG	1,00	4,53	4,53
Perno autoperforante	LB	0,11	6,06	0,67
SUBTOTAL O				13,20

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
Incluye transporte				
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	14,70
INDIRECTOS Y UTILIDADES %25	3,67
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	18,37
VALOR OFERTADO	18,37

ANALISIS DE PRECIOS UNITARIOS

OBRA:	Diseño parametrico la estructura de una nave industrial		
RUBRO:	2	UNIDAD	M
DETALLE	PROVISiÓN E INSTALACIÓN DEL CUMBRERO		

EQUIPOS

DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,09
SUBTOTAL M					0,09

MANO DE OBRA

DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	1,00	3,62	3,62	0,200	0,72
Albañil	1,00	3,66	3,66	0,200	0,73
Maestro de obra	0,25	4,06	1,02	0,200	0,20
SUBTOTAL N					1,65

MATERIALES

DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B
Cumbrero tipo estilpanel	m	1,10	4,00	4,40
PERNO PUNTA DE BROCA10 X 3	u	6,00	0,09	0,54
SUBTOTAL O				4,94

TRANSPORTE

DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B
Incluye transporte				
SUBTOTAL P				0,00

TOTAL COSTO DIRECTO (M+N+O+P)	6,68
INDIRECTOS Y UTILIDADES %25	1,67
OTROS INDIRECTOS %	
COSTO TOTAL DEL RUBRO	8,35
VALOR OFERTADO	8,35

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2		UNIDAD	M2	
DETALLE	MAMPOSTERIA DE BLOQUE				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,22
Andamios	2,00	1,50	3,00	0,27	0,81
SUBTOTAL M					1,03
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	3,00	3,62	10,86	0,270	2,93
Albañil	1,00	3,66	3,66	0,270	0,99
Maestro de obra	0,25	4,06	1,02	0,270	0,27
SUBTOTAL N					4,19
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
Cemento	saco	0,45	7,15	3,22	
Arena	m3	0,03	13,00	0,39	
Agua	m3	0,01	1,85	0,02	
Aditivo	Kg	0,10	1,95	0,20	
Bloque 9x19x39	u.	13,00	0,48	6,24	
Acero en Varillas Fy= 4200Kg/cm2	kg	0,20	1,05	0,21	
SUBTOTAL O				10,28	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
Incluye transporte					
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)				15,50	
INDIRECTOS Y UTILIDADES %25				3,87	
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO				19,37	
VALOR OFERTADO				19,37	

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2		UNIDAD	M2	
DETALLE	ENLUCIDO DE PAREDES				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,24
Andamios	2,00	2,00	4,00	0,25	1,00
SUBTOTAL M					1,24
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	2,00	3,62	7,24	0,250	1,81
Albañil	2,00	3,66	7,32	0,250	1,83
Maestro de obra	1,00	4,06	4,06	0,250	1,02
SUBTOTAL N					4,66
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
Cemento	saco	0,21	7,07	1,48	
Arena fina	m3	0,02	13,00	0,26	
Agua	m3	0,01	1,00	0,01	
SUBTOTAL O				1,75	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
Incluye transporte					
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)				7,65	
INDIRECTOS Y UTILIDADES %25				1,91	
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO				9,56	
VALOR OFERTADO				9,56	

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	M2
DETALLE	ENLUCIDO DE PAREDES				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			0,12
Hidrolavadora	1,00	1,50	1,50	0,180	0,27
Andamio	1,00	1,50	1,50	0,180	0,27
SUBTOTAL M					0,66
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	2,00	3,62	7,24	0,180	1,30
Pintor D2	1,00	3,65	3,65	0,180	0,66
Maestro de obra C1	0,25	4,06	1,02	0,180	0,18
SUBTOTAL N					2,14
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
Empaste	gl	0,06	15,000	0,90	
Pintura blanca	gl	0,06	25,000	1,50	
Agua	m3	0,04	1,850	0,07	
Insumos (lijas, waipe,etc)	u	1,00	0,500	0,50	
SUBTOTAL O				2,97	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)				5,77	
INDIRECTOS Y UTILIDADES %25				1,44	
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO				7,21	
VALOR OFERTADO				7,21	

ANALISIS DE PRECIOS UNITARIOS					
OBRA:	Diseño parametrico la estructura de una nave industrial				
RUBRO:	2			UNIDAD	U
DETALLE	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA				
EQUIPOS					
DESCRIPCION	CANT. A	TARIFA B	COSTO/HORA AxB	RENDIMIENTO C	COSTO D=(A*B)xC
HERRAM. MANUAL		0,05			10,54
CAMION	1,00	15,00	15,00	9,000	135,00
SUBTOTAL M					145,54
MANO DE OBRA					
DESCRIPCION	CANTIDAD A	JORNAL /HR B	COSTO/HORA A*B	RENDIMIENTO R	COSTO C=(A*B)xR
Peon E2	5,00	3,62	18,10	9,000	162,90
Chofer C1	1,00	5,31	5,31	9,000	47,79
SUBTOTAL N					210,69
MATERIALES					
DESCRIPCION	UNIDAD	CANTIDAD A	C.UNIT. B	COSTO C=A*B	
SACOS DE YUTE	U	25,00	4,200	105,00	
SUBTOTAL O				105,00	
TRANSPORTE					
DESCRIPCION	UNIDAD	CANTIDAD A	TARIFA B	COSTO C=A*B	
Incluye transporte					
SUBTOTAL P				0,00	
TOTAL COSTO DIRECTO (M+N+O+P)					461,23
INDIRECTOS Y UTILIDADES %25					115,31
OTROS INDIRECTOS %					
COSTO TOTAL DEL RUBRO					576,54
VALOR OFERTADO					576,54

4.3 Descripción de cantidades de obra

Preliminares	
Trazado, replanteo y nivelación	
Largo	25 m
Ancho	30 m
Área terreno	750 m2
Caseta de materiales y guardianía	
Área caseta	7,5 m2
Cerramiento provisional	
Perímetro de cerramiento	118 m

Movimientos de tierra	
Excavación y desalojo a máquina para cimentaciones	
Área de cimentación	82,89 m2
Profundidad excavación	1 m
Volumen de excavación	82,89 m3
Relleno como material importado	
Volumen de relleno	74,94 m3
Cerramiento provisional	
Perímetro de cerramiento	118 m

Volumen de zapata 1	
Altura de zapata	0,36 m
Largo de zapata	1,50 m
Ancho de zapata	4,45 m
Volumen de zapata	2,40 m3
Altura de pedestal	0,75 m
Largo de pedestal	0,50 m
Ancho de pedestal	1,00 m
Volumen de pedestal	0,38 m3
Cantidad de zapatas y pedestal	4,00 u
Total volumen zapatas	11,11 m3
Relleno con material importado	18,53 m3

Volumen de zapata 2	
Altura de zapata	0,39 m
Largo de zapata	1,80 m
Ancho de zapata	3,80 m
Volumen de zapata	2,67 m3
Altura de pedestal	0,75 m
Largo de pedestal	0,50 m
Ancho de pedestal	1,00 m
Volumen de pedestal	0,38 m3
Cantidad de zapatas y pedestal	4,00 u
Total volumen zapatas	12,17 m3
Relleno con material importado	19,02 m3

Volumen de zapata 3	
Altura de zapata	0,41 m
Largo de zapata	1,80 m
Ancho de zapata	4,00 m
Volumen de zapata	2,95 m3
Altura de pedestal	0,75 m
Largo de pedestal	0,50 m
Ancho de pedestal	1,00 m
Volumen de pedestal	0,38 m3
Cantidad de zapatas y pedestal	4,00 u
Total volumen zapatas	13,31 m3
Relleno con material importado	20,10 m3

Volumen total cimentación	**36,59 m3**
Volumen total relleno importado	**74,94 m3**

Estructura	
Replantillo de hormigón	
Área de cimentación	82,89 m2
Replantillo de hormigón	
Volumen de cimentación	36,59 m3
Acero de refuerzo fy = 4200 kg/cm2, d = 22mm	
Longitud de acero de refuerzo	815,6 m
Peso total del acero de refuerzo	2436,88 kg
Contrapiso de hormigón	
Área terreno	750 m2
Placa de anclaje	
Cantidad de placas de anclaje	12 u
Suministro de acero estructural	
Peso total del acero estructural	33600 kg
Montaje de la carga	
Peso total del acero estructural	33600 kg

Albañilería	
Mampostería de bloque	
Perímetro de paredes	110 m
Altura de paredes	3 m
Área de paredes	330 m2
Enlucido de paredes	
Área de enlucido	660 m2

Pintura	
Contrapiso de hormigón	
Area de pintura	660 m2

4.4 Presupuesto referencial

#	DETALLES DE RUBROS	UNIDAD	CANTIDAD	COSTO	TOTAL
A	**PRELIMINARES**				
1	TRAZADO, REPLANTEO Y NIVELACION	M2	**750,00**	$ 0,70	$ 525,00
2	CASETA DE MATERIALES Y GUARDIANIA	M2	**7,50**	$ 38,16	$ 286,20
3	ROTULO DE OBRA Y SEÑALETICA DE OBRA	U	**1,00**	$ 60,17	$ 60,17
4	CERRAMIENTO PROVISIONAL h=2.40	M	**134,00**	$ 7,20	$ 964,80
	SUBTOTAL				$ -
B	**MOVIMIENTO DE TIERRAS**				$ -
5	EXCAVACION Y DESALOJO A MAQUINA PARA CIMENTACIONES	M3	**81,42**	$ 13,21	$ 1.075,60
6	RELLENO CON MATERIAL IMPORTADO	M3	**105,85**	$ 9,94	$ 1.052,15
	SUBTOTAL				$ -
C	**ESTRUCTURA**				$ -
7	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM	M2	**81,42**	$ 10,20	$ 830,52
8	HORMIGÓN ARMADO PARA PLINTOS (f´c=210 kg/cm²)	M3	**36,01**	$ 281,05	$ 10.120,25
9	ACERO DE REFUERZO f'y=4200kg/cm2	Kg	**2436,88**	$ 2,20	$ 5.361,15
10	CONTRAPISO DE HORMIGON 180 kg/cm2 E=10cm	M2	**750,00**	$ 23,06	$ 17.295,00
11	PLACAS DE ANCLAJE	KG	**6130,80**	$ 3,70	$ 22.683,96
12	SUMINISTRO DE ACERO ESTRUCTURAL	KG	**16800,00**	$ 1,40	$ 23.520,00
13	FABRICACIÓN DE CERCHAS	KG	**16800,00**	$ 2,55	$ 42.840,00
14	MONTAJE DE LA ESTRUCTURA DE ACERO	KG	**16800,00**	$ 1,56	$ 26.208,00
15	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ	KG	**16800,00**	$ 0,52	$ 8.736,00
16	PROVISIÓN E INSTALACIÓN DE LA CUBIERTA	M2	**764,85**	$ 18,37	$ 14.050,35
17	PROVISIÓN E INSTALACIÓN DEL CUMBRERO	M	**25,00**	$ 8,35	$ 208,75
	SUBTOTAL				$ -
D	**ALBAÑILERIA**				$ -
18	MAMPOSTERIA DE BLOQUE	M2	**0,00**	$ 19,37	$ -
19	ENLUCIDO DE PAREDES	M2	**0,00**	$ 9,56	$ -
	SUBTOTAL				$ -
E	**PINTURA EN MAMPOSTERIA**				$ -
20	EMPASTE, PINTURA BLANCA INTERIOR	M2	**0,00**	$ 7,21	$ -
	SUBTOTAL				$ -
F	**VARIOS**				$ -
21	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA.	U	**1,00**	$ 576,54	$ 576,54
	SUBTOTAL				**$ 205,49**
	SUBTOTAL DEL PROYECTO				**$ 187.479,96**
	GASTOS DE SEGURIDAD INDUSTRIAL				**$ 1.874,80**
	VALOR TOTAL DEL PROYECTO				**$ 189.354,76**

4.5 Valoración integral del costo del proyecto incluyendo las medidas de prevención y mitigación del impacto ambiental

El costo total del proyecto es de USD $189.354,76.

4.6 Cronograma valorado

	DETALLES DE RUBROS	P. TOTAL	Semana 1	Semana 2	Semana 3	Semana 4
A	**PRELIMINARES**					
1	TRAZADO, REPLANTEO Y NIVELACION	$ 525,00	$ 236,25	$ 288,75		
2	CASETA DE MATERIALES Y GUARDIANIA	$ 286,20	$ 286,20			
3	ROTULO DE OBRA Y SEÑALETICA DE OBRA	$ 60,17	$ 60,17			
4	CERRAMIENTO PROVISIONAL h=2.40	$ 964,80		$ 192,96	$ 771,84	
	SUBTOTAL	**$ 1.836,17**				
B	**MOVIMIENTO DE TIERRAS**					
5	EXCAVACION Y DESALOJO A MAQUINA PARA CIMENTACIONES	$ 1.075,60	$ 118,32	$ 591,58	$ 365,70	
6	RELLENO CON MATERIAL IMPORTADO	$ 1.052,15			$ 294,60	
	SUBTOTAL	**$ 2.127,75**				
C	**ESTRUCTURA**					
7	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM	$ 830,52				
8	PLINTOS DE HORMIGÓN ARMADO (f´c=210 kg/cm²)	$ 10.120,25				
9	ACERO DE REFUERZO f'y=4200kg/cm2	$ 5.361,15				
10	CONTRAPISO DE HORMIGON 180 kg/cm2	$ 17.295,00				
11	PLACAS DE ANCLAJE	$ 22.683,96			$ 22.683,96	
12	SUMINISTRO DE ACERO ESTRUCTURAL	$ 23.520,00				$ 16.934,40
13	FABRICACIÓN DE CERCHAS	$ 42.840,00				
	MONTAJE DE LA CARGA	$ 26.208,00				
14	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ	$ 8.736,00				
15	PROVISIÓN E INSTALACIÓN DE LA CUBIERTA	$ 14.050,35				
16	PROVISIÓN E INSTALACIÓN DEL CUMBRERO	$ 208,75				
	SUBTOTAL	**$ 171.853,97**				
D	**ALBAÑILERIA**					
17	MAMPOSTERIA DE BLOQUE	$ -				
18	ENLUCIDO DE PAREDES	$ -				
	SUBTOTAL	**$ -**				
E	**PINTURA**					
19	EMPASTE, PINTURA BLANCA INTERIOR	$ -				
	SUBTOTAL	**$ -**				
F	**VARIOS**					
20	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA.	$ 576,54	$ 44,35	$ 44,35	$ 44,35	$ 44,35
	SUBTOTAL	**$ 576,54**				
	SUBTOTAL DEL PROYECTO	**$ 176.394,43**				
	GASTOS DE SEGURIDAD INDUSTRIAL	**$ 1.874,80**	$ 144,22	$ 144,22	$ 144,22	$ 144,22
			$ 889,50	$ 1.261,86	$ 24.304,67	$ 17.122,96
	VALOR TOTAL DEL PROYECTO	**$ 178.269,23**				

	DETALLES DE RUBROS	P. TOTAL	Semana 5	Semana 6	Semana 7	Semana 8
A	**PRELIMINARES**					
1	TRAZADO, REPLANTEO Y NIVELACION	$ 525,00				
2	CASETA DE MATERIALES Y GUARDIANIA	$ 286,20				
3	ROTULO DE OBRA Y SEÑALETICA DE OBRA	$ 60,17				
4	CERRAMIENTO PROVISIONAL h=2.40	$ 964,80				
	SUBTOTAL	**$ 1.836,17**				
B	**MOVIMIENTO DE TIERRAS**					
5	EXCAVACION Y DESALOJO A MAQUINA PARA CIMENTACIONES	$ 1.075,60				
6	RELLENO CON MATERIAL IMPORTADO	$ 1.052,15			$ 757,55	
	SUBTOTAL	**$ 2.127,75**				
C	**ESTRUCTURA**					
7	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM	$ 830,52	$ 830,52			
8	PLINTOS DE HORMIGÓN ARMADO (f´c=210 kg/cm²)	$ 10.120,25			$ 10.120,25	
9	ACERO DE REFUERZO f'y=4200kg/cm2	$ 5.361,15		$ 5.361,15		
10	CONTRAPISO DE HORMIGON 180 kg/cm2	$ 17.295,00				$ 17.295,00
11	PLACAS DE ANCLAJE	$ 22.683,96				
12	SUMINISTRO DE ACERO ESTRUCTURAL	$ 23.520,00	$ 6.585,60			
13	FABRICACIÓN DE CERCHAS	$ 42.840,00	$ 28.560,00	$ 14.280,00		
	MONTAJE DE LA CARGA	$ 26.208,00				$ 13.104,00
14	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ	$ 8.736,00				
15	PROVISIÓN E INSTALACIÓN DE LA CUBIERTA	$ 14.050,35				
16	PROVISIÓN E INSTALACIÓN DEL CUMBRERO	$ 208,75				
	SUBTOTAL	**$ 171.853,97**				
D	**ALBAÑILERIA**					
17	MAMPOSTERIA DE BLOQUE	$ -				
18	ENLUCIDO DE PAREDES	$ -				
	SUBTOTAL	**$ -**				
E	**PINTURA**					
19	EMPASTE, PINTURA BLANCA INTERIOR	$ -				
	SUBTOTAL	**$ -**				
F	**VARIOS**					
20	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA.	$ 576,54	$ 44,35	$ 44,35	$ 44,35	$ 44,35
	SUBTOTAL	**$ 576,54**				
	SUBTOTAL DEL PROYECTO	**$ 176.394,43**				
	GASTOS DE SEGURIDAD INDUSTRIAL	**$ 1.874,80**	$ 144,22	$ 144,22	$ 144,22	$ 144,22
			$ 36.164,68	$ 19.829,71	$ 11.066,36	$ 30.587,56
	VALOR TOTAL DEL PROYECTO	**$ 178.269,23**				

	DETALLES DE RUBROS	P. TOTAL	Semana 9	Semana 10	Semana 11	Semana 12	Semana 13
A	**PRELIMINARES**						
1	TRAZADO, REPLANTEO Y NIVELACION	$ 525,00					
2	CASETA DE MATERIALES Y GUARDIANIA	$ 286,20					
3	ROTULO DE OBRA Y SEÑALETICA DE OBRA	$ 60,17					
4	CERRAMIENTO PROVISIONAL h=2.40	$ 964,80					
	SUBTOTAL	**$ 1.836,17**					
B	**MOVIMIENTO DE TIERRAS**						
5	EXCAVACION Y DESALOJO A MAQUINA PARA CIMENTACIONES	$ 1.075,60					
6	RELLENO CON MATERIAL IMPORTADO	$ 1.052,15					
	SUBTOTAL	**$ 2.127,75**					
C	**ESTRUCTURA**						
7	REPLANTILLO DE HORMIGON DE 140 KG/CM2 E= 5CM	$ 830,52					
8	PLINTOS DE HORMIGÓN ARMADO (f´c=210 kg/cm²)	$ 10.120,25					
9	ACERO DE REFUERZO f'y=4200kg/cm2	$ 5.361,15					
10	CONTRAPISO DE HORMIGON 180 kg/cm2	$ 17.295,00					
11	PLACAS DE ANCLAJE	$ 22.683,96					
12	SUMINISTRO DE ACERO ESTRUCTURAL	$ 23.520,00					
13	FABRICACIÓN DE CERCHAS	$ 42.840,00					
	MONTAJE DE LA CARGA	$ 26.208,00	$ 13.104,00				
14	PINTADO DE ESTRUCTURA CON SINTÉTICO AUTOMOTRIZ	$ 8.736,00	$ 3.744,00	$ 4.992,00			
15	PROVISIÓN E INSTALACIÓN DE LA CUBIERTA	$ 14.050,35		$ 14.050,35			
16	PROVISIÓN E INSTALACIÓN DEL CUMBRERO	$ 208,75		$ 208,75			
	SUBTOTAL	**$ 171.853,97**					
D	**ALBAÑILERIA**						
17	MAMPOSTERIA DE BLOQUE	$ -		$ -	$ -	$ -	
18	ENLUCIDO DE PAREDES	$ -				$ -	
	SUBTOTAL	**$ -**					
E	**PINTURA**						
19	EMPASTE, PINTURA BLANCA INTERIOR	$ -					$ -
	SUBTOTAL	**$ -**					
F	**VARIOS**						
20	LIMPIEZA Y DESALOJO PROGRESIVO DE ESCOMBROS EN OBRA.	$ 576,54	$ 44,35	$ 44,35	$ 44,35	$ 44,35	$ 44,35
	SUBTOTAL	**$ 576,54**					
	SUBTOTAL DEL PROYECTO	**$ 176.394,43**					
	GASTOS DE SEGURIDAD INDUSTRIAL	**$ 1.874,80**	$ 144,22	$ 144,22	$ 144,22	$ 144,22	$ 144,22
			$ 17.036,56	$ 19.439,66	$ 188,56	$ 188,56	$ 188,56
	VALOR TOTAL DEL PROYECTO	**$ 178.269,23**					**$ 178.269,23**

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Se desarrolló una aplicación de escritorio que diseña exitosamente los elementos estructurales de una nave industrial. Para comparar los resultados obtenidos, se realizó un cálculo manual desde el prediseño de los elementos y se comprobó que los resultados son satisfactorios.

El diseño estructural se automatizó completamente, y el usuario puede ver a tiempo real como la estructura se va generando en Sap2000. Además, se logró automatizar la modelación de la estructura y la generación de los planos en Tekla Structures mediante el uso de su API.

A pesar de que existen herramientas implementadas en otros programas que diseñan cierto tipo de elemento estructural, o incluso una estructura, el presente trabajo elimina la modelación del proceso de diseño, lo que reduce en gran manera el tiempo de diseño.

La interfaz de programación de aplicaciones de Sap2000 permitió automatizar el problema común en la ingeniería estructural, la iteración en el proceso de diseño. Sin embargo, también se puede desarrollar soluciones para problemas que no necesariamente requieran automatización.

Recomendaciones

Para futuras versiones del programa se puede automatizar otros tipos de galpones, como curvos o a diente sierra. Aunque también se podría trabajar con edificaciones en general.

Si se desea desarrollar una aplicación similar, pero que trabaje en conjunto con otro software de la gamma existente en la ingeniería civil, es recomendable conocer muy bien el software porque esto facilita comprender su respectiva interfaz de programación de aplicaciones, si esta posee una.

Se recomienda implementar mejoras a la generación automática de los planos, como el poder introducir las plantillas de acero en función de los cálculos realizados en el diseño.

Aunque la interfaz gráfica de la aplicación realizada es sencilla y posee pocos parámetros que el maestro de obra debe manipular, se recomienda desarrollar una versión más avanzada para ingenieros.

Como en el diseño de la cimentación existen muchos parámetros, que se limitaron en este proyecto debido a que el usuario objetivo es un maestro de obra, se recomienda implementar el diseño de otros tipos de cimentaciones, o incluso utilizar el API de algún programa de geotecnia para este propósito.

BIBLIOGRAFÍA

Briaud, J.-L. (2013). *Geotechnical Engineering*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781118686195

Budhu, M. (2000). *Soil Mechanics and Foundations* (John Wiley & Sons).

Canchari, E. (2009). *Interfaz de Programación para Aplicaciones del Sap2000 y Visual C# Fundamentado en MathCad*.

Coba, G. (2021, March 27). *Las bodegas corporativas ganan terreno en Guayaquil y Cuenca*. PRIMICIAS. https://www.primicias.ec/noticias/economia/bodegas-corporativas-espacio-sector-inmobiliario/

Coduto, D. P. (2001). *Foundation design: Principles and practices* (Prentice Hall, Ed.; Upper Saddle River).

Córdova Reyes, M. F. (2014). *ESTUDIO COMPARATIVO DEL SISTEMA CONSTRUCTIVO EN HORMIGÓN Y ACERO, EN UN EDIFICIO*.

Das, B. M., & González, S. R. C. (2015). *Fundamentos de ingeniería geotécnica* (Cengage Learning, Ed.).

Diario Expreso. (2019). La necesidad de más inmobiliaria industrial en el país. *Diario Expreso*. https://www.pressreader.com/ecuador/diario-expreso/20190329/281694026135006

Ministerio de Desarrollo Urbano y Vivienda. (2014). *Norma Ecuatoriana de la Construcción. CARGAS (NO SÍSMICAS)*. https://www.habitatyvivienda.gob.ec/documentos-normativos-nec-norma-ecuatoriana-de-la-construccion/

Padilla & Portugal. (2013). *Especialistas: Naves Industriales Techos Metálicos Estructuras Metálicas*. Padilla y Portugal. https://padillayportugal.com/

Salgado, R. (2008). *The engineering of foundations* (McGraw Hill, Ed.).

Wagner, B. (2017, July 1). *Namespaces*. Microsoft. https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/namespaces

Wagner, B. (2020, June 8). *Clases y Objetos*. Microsoft. https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/types

Wagner, B., Nakamura, E., & Warren, G. (2020, October 21). *Overview of .NET Framework*. Microsoft. https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

PLANOS Y ANEXOS

Anexo 1

```
namespace WPF_Industrial_Shed_Designer
{
    public class FrameForce
    {
        public string Obj;
        public double ObjSta;
        public string Elm;
        public double ElmSta;
        public string LoadCase;
        public string StepType;
        public double StepNum;
        public double P;
        public double V2;
        public double V3;
        public double T;
        public double M2;
        public double M3;

        public async static Task<List<FrameForce>> RetrieveForces(esaSap2000 esaSap2000,
string Name, eItemTypeElm ItemTypeElm)
        {
            int NumberResults = 0;
            string[] Obj = null;
            double[] ObjSta = null;
            string[] Elm = null;
            double[] ElmSta = null;
            string[] LoadCase = null;
            string[] StepType = null;
            double[] StepNum = null;
            double[] P = null;
            double[] V2 = null;
            double[] V3 = null;
            double[] T = null;
            double[] M2 = null;
            double[] M3 = null;

            var a = await esaSap2000.ResultsFrameForce(

                Name,
                ItemTypeElm,
                NumberResults,
                Obj,
                ObjSta,
                Elm,
                ElmSta,
                LoadCase,
                StepType,
                StepNum,
                P,
                V2,
                V3,
                T,
                M2,
                M3
            );
            NumberResults = a.NumberResults;

            List<FrameForce> List = new List<FrameForce>();
            for (int i = 0; i < NumberResults; i++)
            {
```

```
                FrameForce frameForce = new FrameForce();
                frameForce.Obj = a.Obj[i];
                frameForce.ObjSta = a.ObjSta[i];
                frameForce.Elm = a.Elm[i];
                frameForce.ElmSta = a.ElmSta[i];
                frameForce.LoadCase = a.LoadCase[i];
                frameForce.StepType = a.StepType[i];
                frameForce.StepNum = a.StepNum[i];
                frameForce.P = a.P[i];
                frameForce.V2 = a.V2[i];
                frameForce.V3 = a.V3[i];
                frameForce.T = a.T[i];
                frameForce.M2 = a.M2[i];
                frameForce.M3 = a.M3[i];

                List.Add(frameForce);
            }
            return List;
        }
    }
}
```

Anexo 2

```
public async Task CalculateTrussGeometry(int i, double Ycoor)
        {
            angulo = Math.Atan(h2value / (L1value / 2));
            double m1 = 0;
            double m2 = 0;
            Point3D intersectJoint = new Point3D();

            await esaSap2000.SetPresentUnits(eUnits.kgf_m_C);

            if (a1 != b1)
            {
                m1 = h2value / (L1value / 2);
                intersectJoint = new Point3D(a1, Ycoor, m1 * a1 + h1value);
            }
            else
            {
                m1 = h2value / (L1value / 2);
                intersectJoint = new Point3D(a1, Ycoor, m1 * a1 + h1value);
            }

            //Primera parte
            Point3D initialJoint1 = new Point3D(0, Ycoor, 0);
            Point3D endJoint1 = new Point3D(0, Ycoor, h1value - a1);
            Point3D initialJoint2 = new Point3D(b1, Ycoor, 0);
            Point3D endJoint2 = new Point3D(a1, Ycoor, h1value - a1);

            Tuple<int, int> numberofFramesTuple = await generateTruss
                (initialJoint1, endJoint1,
                initialJoint2, endJoint2,
                nParts1,
                SectionCName, SectionLName,
                numberofCframes, numberofLframes,
                true,
                true, true,
```

```
                90, -90, 0, 0,
                SectionCName, SectionLName,
                90, 0);

            numberofCframes = numberofFramesTuple.Item1;
            numberofLframes = numberofFramesTuple.Item2;

            //Segunda parte
            initialJoint1 = new Point3D(0, Ycoor, h1value - a1);
            endJoint1 = new Point3D(0, Ycoor, h1value);
            initialJoint2 = new Point3D(a1, Ycoor, h1value - a1);
            endJoint2 = intersectJoint;

            numberofFramesTuple = await generateTrussCorner
                (initialJoint1, endJoint1,
                 initialJoint2, endJoint2,
                 SectionCName, SectionDoubleCName,
                 numberofCframes, numberofDoubleCframes,
                 true,
                 90, -90, 90);
            numberofCframes = numberofFramesTuple.Item1;
            numberofDoubleCframes = numberofFramesTuple.Item2;

            //Tercera parte
            initialJoint1 = intersectJoint;
            endJoint1 = new Point3D(L1value / 4, Ycoor, m1 * L1value / 4 + h1value);
            initialJoint2 = new Point3D(a1, Ycoor, h1value - a1);
            double ratio = Math.Abs(FrameForceResultsList.ElementAt(1).ElementAt(5).M3) /
momentoMaximo;
            if (ratio < 0.5)
            {
                ratio = 0.5;
            }
            endJoint2 = new Point3D(L1value / 4, Ycoor, m1 * L1value / 4 + h1value -
ratio);

            numberofFramesTuple = await generateTruss
                (initialJoint1, endJoint1,
                 initialJoint2, endJoint2,
                 nParts2,
                 SectionCName, SectionLName,
                 numberofCframes, numberofLframes,
                 false,
                 true, false,
                 -90, 90, 0, 0,
                 SectionLName, SectionLName,
                 0, 0);
            numberofCframes = numberofFramesTuple.Item1;
            numberofLframes = numberofFramesTuple.Item2;


            //Cuarta parte
            initialJoint1 = new Point3D(L1value / 4, Ycoor, m1 * L1value / 4 + h1value);
            endJoint1 = new Point3D(L1value / 2, Ycoor, m1 * L1value / 2 + h1value);
            initialJoint2 = new Point3D(L1value / 4, Ycoor, m1 * L1value / 4 + h1value -
ratio);
            endJoint2 = new Point3D(L1value / 2, Ycoor, m1 * L1value / 2 + h1value -
ratio);

            numberofFramesTuple = await generateTruss2
```

```
            (initialJoint1, endJoint1,
             initialJoint2, endJoint2,
             nParts3,
             SectionCName, SectionLName,
             numberofCframes, numberofLframes,
             false,
             true, false,
             -90, 90, 0, 0,
             SectionLName, SectionLName,
             0, 0, 1);
        numberofCframes = numberofFramesTuple.Item1;
        numberofLframes = numberofFramesTuple.Item2;

        //Quinta parte
        initialJoint1 = new Point3D(L1value, Ycoor, 0);
        endJoint1 = new Point3D(L1value, Ycoor, h1value - a1);
        initialJoint2 = new Point3D(L1value - b1, Ycoor, 0);
        endJoint2 = new Point3D(L1value - a1, Ycoor, h1value - a1);

        numberofFramesTuple = await generateTruss
            (initialJoint1, endJoint1,
            initialJoint2, endJoint2,
            nParts1,
            SectionCName, SectionLName,
            numberofCframes, numberofLframes,
            true,
            true, true,
            -90, 90, 0, 0,
            SectionCName, SectionLName,
            90, 0);

        numberofCframes = numberofFramesTuple.Item1;
        numberofLframes = numberofFramesTuple.Item2;

        //Sexta parte
        initialJoint1 = new Point3D(L1value, Ycoor, h1value - a1);
        endJoint1 = new Point3D(L1value, Ycoor, h1value);
        initialJoint2 = new Point3D(L1value - a1, Ycoor, h1value - a1);
        endJoint2 = new Point3D(L1value - intersectJoint.X, Ycoor, intersectJoint.Z);

        numberofFramesTuple = await generateTrussCorner
            (initialJoint1, endJoint1,
             initialJoint2, endJoint2,
             SectionCName, SectionDoubleCName,
             numberofCframes, numberofDoubleCframes,
             true,
             -90, -90, 90);
        numberofCframes = numberofFramesTuple.Item1;
        numberofDoubleCframes = numberofFramesTuple.Item2;

        //Septima parte
        initialJoint1 = new Point3D(L1value - intersectJoint.X, Ycoor,
intersectJoint.Z);
        endJoint1 = new Point3D(L1value / 4 + L1value / 2, Ycoor, m1 * L1value / 4 +
h1value);
        initialJoint2 = new Point3D(L1value - a1, Ycoor, h1value - a1);
        endJoint2 = new Point3D(L1value / 4 + L1value / 2, Ycoor, m1 * L1value / 4 +
h1value - ratio);

        numberofFramesTuple = await generateTruss
```

```
                (initialJoint1, endJoint1,
                 initialJoint2, endJoint2,
                 nParts2,
                 SectionCName, SectionLName,
                 numberofCframes, numberofLframes,
                 false,
                 true, false,
                 -90, 90, 0, 0,
                 SectionLName, SectionLName,
                 0, 0);
            numberofCframes = numberofFramesTuple.Item1;
            numberofLframes = numberofFramesTuple.Item2;

            //Octava parte
            initialJoint1 = new Point3D(L1value / 4 + L1value / 2, Ycoor, m1 * L1value /
4 + h1value);
            endJoint1 = new Point3D(L1value / 2, Ycoor, m1 * L1value / 2 + h1value);
            initialJoint2 = new Point3D(L1value / 4 + L1value / 2, Ycoor, m1 * L1value /
4 + h1value - ratio);
            endJoint2 = new Point3D(L1value / 2, Ycoor, m1 * L1value / 2 + h1value -
ratio);

            numberofFramesTuple = await generateTruss2
                (initialJoint1, endJoint1,
                 initialJoint2, endJoint2,
                 nParts3,
                 SectionCName, SectionLName,
                 numberofCframes, numberofLframes,
                 false,
                 true, true,
                 -90, 90, 0, 0,
                 SectionLName, SectionLName,
                 0, 0, -1);
            numberofCframes = numberofFramesTuple.Item1;
            numberofLframes = numberofFramesTuple.Item2;


            //Set joints restraints
            int TotalNumberOfCFrames = 4 * (nParts1 + nParts2 + nParts3) + 8;
            string PointName1 = " ";
            string PointName2 = " ";
            bool[] Restraint = new bool[6] { true, true, true, true, true, true };
            var tuple = await esaSap2000.FrameObjGetPoints("C-" + Convert.ToString((i -
1) * TotalNumberOfCFrames + 2), PointName1, PointName2);
            await esaSap2000.PointObjStRestraint(tuple.Item1, Restraint);

            tuple = await esaSap2000.FrameObjGetPoints("C-" + Convert.ToString((i - 1) *
TotalNumberOfCFrames + 3), PointName1, PointName2);
            await esaSap2000.PointObjStRestraint(tuple.Item1, Restraint);

            tuple = await esaSap2000.FrameObjGetPoints("C-" + Convert.ToString((i - 1) *
TotalNumberOfCFrames + 2 * (nParts1 + nParts2 + nParts3) + 6), PointName1, PointName2);
            await esaSap2000.PointObjStRestraint(tuple.Item1, Restraint);

            tuple = await esaSap2000.FrameObjGetPoints("C-" + Convert.ToString((i - 1) *
TotalNumberOfCFrames + 2 * (nParts1 + nParts2 + nParts3) + 7), PointName1, PointName2);
            await esaSap2000.PointObjStRestraint(tuple.Item1, Restraint);

            await esaSap2000.RefreshView();
        }
```

Anexo 3

```
public async Task CalculateGFramesGeometry(int i, double Ycoor)
        {
            int TotalNumberOfCFrames = 4 * (nParts1 + nParts2 + nParts3) + 8;

            //Adding G frames
            // Part 1
            int jLimit = (2 * nParts1 + 3) + (2 * nParts2);
            int jInit = (2 * nParts1 + 3) + 1;
            double localAxesG = 180 - angulo * 180 / Math.PI;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 8, localAxesG);

            //Part 2
            jLimit = (2 * nParts1 + 3) + (2 * nParts2) + (2 * nParts3);
            jInit = (2 * nParts1 + 3) + (2 * nParts2) + 1;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 8, localAxesG);

            //Part 2*
            jLimit = (2 * nParts1 + 3) + (2 * nParts2) + (2 * nParts3) + 1;
            jInit = (2 * nParts1 + 3) + (2 * nParts2) + (2 * nParts3) + 1;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 8, localAxesG);

            //Part 3
            jLimit = 2 * (2 * nParts1 + 3) + 2 * (2 * nParts2) + (2 * nParts3) + (1);
            jInit = 2 * (2 * nParts1 + 3) + (2 * nParts2) + (2 * nParts3) + (1) + 1;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 2, angulo * 180 / Math.PI);

            //Part 4
            jLimit = 2 * (2 * nParts1 + 3) + 2 * (2 * nParts2) + 2 * (2 * nParts3) + (1);
            jInit = 2 * (2 * nParts1 + 3) + 2 * (2 * nParts2) + 2 * nParts3 + (1) + 1;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 2, angulo * 180 / Math.PI);

            //Part 4*
            jLimit = 2 * (2 * nParts1 + 3) + 2 * (2 * nParts2) + 2 * (2 * nParts3) + 2 *
(1);
            jInit = 2 * (2 * nParts1 + 3) + 2 * (2 * nParts2) + 2 * (2 * nParts3) + (1) +
1;
            numberofGframes = await generateGFrames(i, jInit, jLimit,
TotalNumberOfCFrames, numberofGframes, SectionGName, 2, angulo * 180 / Math.PI);

        }
```

Anexo 4

```
public async Task CalculateTensorsGeometry(int i, double Ycoor, double Yseparation, int
nOf6Crosses, int nOf5Crosses, int nOf4Crosses, int nOf3Crosses, int nOf6Crossesdif, int
nOf5Crossesdif, int nOf4Crossesdif, int nOf3Crossesdif)
        {
            await CalculateTensorsOnTheCover(i, Ycoor, Yseparation, nOf6Crosses,
nOf5Crosses, nOf4Crosses, nOf3Crosses, nOf6Crossesdif, nOf5Crossesdif, nOf4Crossesdif,
nOf3Crossesdif);
            await CalculateTensorsBetweenGFrames(i, Ycoor, Yseparation);
        }


public async Task CalculateTensorsOnTheCover(int i, double Ycoor, double Yseparation, int
nOf6Crosses, int nOf5Crosses, int nOf4Crosses, int nOf3Crosses, int nOf6Crossesdif, int
nOf5Crossesdif, int nOf4Crossesdif, int nOf3Crossesdif)
        {
            if (i > 1)
            {
                //Adding Crosses
                string pointName = "";
                string pointName2 = "";
                double x = 0;
                double y = 0;
                double z = 0;
                Point3D point1 = new Point3D(0, Ycoor - Yseparation, h1value);
                Point3D point2 = new Point3D(0, Ycoor, h1value);
                Point3D point3 = new Point3D();
                Point3D point4 = new Point3D();
                int initlimit = 0;

                if (nOf3Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(3 + (i - 2) * TotalNumberOfGFrames), pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    initlimit = 3;
                    nOf3Crossesdif = 1;
                }
                else if (nOf4Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(4 + (i - 2) * TotalNumberOfGFrames), pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    initlimit = 4;
                    nOf4Crossesdif = 1;
                }
                else if (nOf5Crosses != 0)
                {
```

```
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(5 + (i - 2) * TotalNumberOfGFrames), pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    initlimit = 5;
                    nOf5Crossesdif = 1;
                }
                else if (nOf6Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(6 + (i - 2) * TotalNumberOfGFrames), pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    initlimit = 6;
                    nOf6Crossesdif = 1;
                }

                nOf6Crosses -= nOf6Crossesdif;
                nOf5Crosses -= nOf5Crossesdif;
                nOf4Crosses -= nOf4Crossesdif;
                nOf3Crosses -= nOf3Crossesdif;

                numberofTensors = await generateTensor(numberofTensors, point1, point4,
result.Tables[13].Rows);
                numberofTensors = await generateTensor(numberofTensors, point3, point2,
result.Tables[13].Rows);

                int limit = initlimit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 3, nOf3Crosses, result.Tables[13].Rows);
                limit = nOf3Crosses * 3 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 4, nOf4Crosses, result.Tables[13].Rows);
                limit = nOf4Crosses * 4 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 5, nOf5Crosses, result.Tables[13].Rows);
                limit = nOf5Crosses * 5 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 6, nOf6Crosses, result.Tables[13].Rows);

                point1 = new Point3D(L1value, Ycoor - Yseparation, h1value);
                point2 = new Point3D(L1value, Ycoor, h1value);

                nOf6Crosses += nOf6Crossesdif;
                nOf5Crosses += nOf5Crossesdif;
                nOf4Crosses += nOf4Crossesdif;
                nOf3Crosses += nOf3Crossesdif;

                if (nOf3Crosses != 0)
                {
```

```
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(3 + TotalNumberOfGFrames / 2 + (i - 2) * TotalNumberOfGFrames),
pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                }
                else if (nOf4Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(4 + TotalNumberOfGFrames / 2 + (i - 2) * TotalNumberOfGFrames),
pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                }
                else if (nOf5Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(5 + TotalNumberOfGFrames / 2 + (i - 2) * TotalNumberOfGFrames),
pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                }
                else if (nOf6Crosses != 0)
                {
                    var tuple = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(6 + TotalNumberOfGFrames / 2 + (i - 2) * TotalNumberOfGFrames),
pointName, pointName2);
                    var tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item1,
x, y, z);
                    point3 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                    tuple2 = await esaSap2000.PointObjGetCoordCartesian(tuple.Item2, x,
y, z);
                    point4 = new Point3D(tuple2.Item1, tuple2.Item2, tuple2.Item3);
                }

                numberofTensors = await generateTensor(numberofTensors, point1, point4,
result.Tables[13].Rows);
                numberofTensors = await generateTensor(numberofTensors, point3, point2,
result.Tables[13].Rows);


                nOf6Crosses -= nOf6Crossesdif;
                nOf5Crosses -= nOf5Crossesdif;
                nOf4Crosses -= nOf4Crossesdif;
                nOf3Crosses -= nOf3Crossesdif;

                limit = nParts2 + nParts3 + 1 + initlimit;
```

```
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 3, nOf3Crosses, result.Tables[13].Rows);
                limit = nOf3Crosses * 3 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 4, nOf4Crosses, result.Tables[13].Rows);
                limit = nOf4Crosses * 4 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 5, nOf5Crosses, result.Tables[13].Rows);
                limit = nOf5Crosses * 5 + limit;
                numberofTensors = await generateCross(numberofTensors,
TotalNumberOfGFrames, i, limit, 6, nOf6Crosses, result.Tables[13].Rows);

            }
        }


public async Task CalculateTensorsBetweenGFrames(int i, double Ycoor, double Yseparation)
        {
            double x11 = 0;
            double y11 = 0;
            double z11 = 0;
            double x22 = 0;
            double y22 = 0;
            double z22 = 0;
            string point11 = "";
            string point22 = "";
            string point33 = "";
            string point44 = "";

            if (i > 1)
            {
                for (int j = 1; j <= TotalNumberOfGFrames / 2 - 1; j++)
                {

                    var tuple1 = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(j + (i - 2) * TotalNumberOfGFrames), point11, point22);
                    var tuple2 = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(j + 1 + (i - 2) * TotalNumberOfGFrames), point33, point44);
                    var point = await esaSap2000.PointObjGetCoordCartesian(tuple1.Item1,
x11, y11, z11);
                    var point2 = await esaSap2000.PointObjGetCoordCartesian(tuple2.Item1,
x22, y22, z22);
                    x11 = point.Item1;
                    y11 = point.Item2;
                    z11 = point.Item3;
                    x22 = point2.Item1;
                    y22 = point2.Item2;
                    z22 = point2.Item3;

                    if (Yseparation >= 4 && Yseparation < 5)
                    {
                        double ycoor = Ycoor - Yseparation + Yseparation / 2;
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor, z11), new Point3D(x22, ycoor, z22), result.Tables[13].Rows);

                        if (j == 1)
                        {
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(0, ycoor, h1value), new Point3D(x11, ycoor, z11), result.Tables[13].Rows);
                        }
```

```
                    }
                    else if (Yseparation <= 6)
                    {
                        double ycoor = Ycoor - Yseparation + Yseparation / 3;
                        double ycoor2 = Ycoor - Yseparation + 2 * Yseparation / 3;
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor, z11), new Point3D(x22, ycoor, z22), result.Tables[13].Rows);
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor2, z11), new Point3D(x22, ycoor2, z22), result.Tables[13].Rows);

                        if (j == 1)
                        {
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(0, ycoor, h1value), new Point3D(x11, ycoor, z11), result.Tables[13].Rows);
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(0, ycoor2, h1value), new Point3D(x11, ycoor2, z11), result.Tables[13].Rows);
                        }
                    }
                }

                for (int j = TotalNumberOfGFrames / 2 + 1; j <= TotalNumberOfGFrames - 1;
j++)
                {
                    var tuple1 = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(j + (i - 2) * TotalNumberOfGFrames), point11, point22);
                    var tuple2 = await esaSap2000.FrameObjGetPoints("G-" +
Convert.ToString(j + 1 + (i - 2) * TotalNumberOfGFrames), point33, point44);
                    var point = await esaSap2000.PointObjGetCoordCartesian(tuple1.Item1,
x11, y11, z11);
                    var point2 = await esaSap2000.PointObjGetCoordCartesian(tuple2.Item1,
x22, y22, z22);
                    x11 = point.Item1;
                    y11 = point.Item2;
                    z11 = point.Item3;
                    x22 = point2.Item1;
                    y22 = point2.Item2;
                    z22 = point2.Item3;

                    if (Yseparation >= 4 && Yseparation < 5)
                    {
                        double ycoor = Ycoor - Yseparation + Yseparation / 2;
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor, z11), new Point3D(x22, ycoor, z22), result.Tables[13].Rows);

                        if (j == TotalNumberOfGFrames / 2 + 1)
                        {
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(L1value, ycoor, h1value), new Point3D(x11, ycoor, z11), result.Tables[13].Rows);
                        }
                    }
                    else if (Yseparation <= 6)
                    {
                        double ycoor = Ycoor - Yseparation + Yseparation / 3;
                        double ycoor2 = Ycoor - Yseparation + 2 * Yseparation / 3;
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor, z11), new Point3D(x22, ycoor, z22), result.Tables[13].Rows);
                        numberofTensors = await generateTensor(numberofTensors, new
Point3D(x11, ycoor2, z11), new Point3D(x22, ycoor2, z22), result.Tables[13].Rows);

                        if (j == TotalNumberOfGFrames / 2 + 1)
```

```
                        {
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(L1value, ycoor, h1value), new Point3D(x11, ycoor, z11), result.Tables[13].Rows);
                            numberofTensors = await generateTensor(numberofTensors, new
Point3D(L1value, ycoor2, h1value), new Point3D(x11, ycoor2, z11),
result.Tables[13].Rows);
                        }
                    }
                }
            }
        }
```

Anexo 5

```
public async Task CalculateTieBeamsGeometry(int i, double Ycoor, double Yseparation, int
nPartsCerchaAmarre)
        {
            if (i > 1)
            {
                //Parte 1
                Point3D initialJoint1 = new Point3D(0, Ycoor - Yseparation, h1value -
a1);
                Point3D endJoint1 = new Point3D(0, Ycoor - Yseparation + h3value, h1value
- h3value);
                Point3D initialJoint2 = new Point3D(0, Ycoor - Yseparation, h1value);
                Point3D endJoint2 = new Point3D(0, Ycoor - Yseparation + h3value,
h1value);

                Tuple<int, int> numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     1,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     true,
                     false, false,
                     90, -90, 0, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 2
                initialJoint1 = new Point3D(0, Ycoor - Yseparation + h3value, h1value -
h3value);
                endJoint1 = new Point3D(0, Ycoor - Yseparation / 2, h1value - h3value);
                initialJoint2 = new Point3D(0, Ycoor - Yseparation + h3value, h1value);
                endJoint2 = new Point3D(0, Ycoor - Yseparation / 2, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     nPartsCerchaAmarre,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     true,
```

```
                    true, false,
                    90, -90, 90, 0,
                    SectionLName, SectionLName,
                    0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 3
                initialJoint1 = new Point3D(0, Ycoor - Yseparation / 2, h1value -
h3value);
                endJoint1 = new Point3D(0, Ycoor - h3value, h1value - h3value);
                initialJoint2 = new Point3D(0, Ycoor - Yseparation / 2, h1value);
                endJoint2 = new Point3D(0, Ycoor - h3value, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     nPartsCerchaAmarre,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     false,
                     true, true,
                     90, -90, 90, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 4
                initialJoint1 = new Point3D(0, Ycoor - h3value, h1value - h3value);
                endJoint1 = new Point3D(0, Ycoor, h1value - a1);
                initialJoint2 = new Point3D(0, Ycoor - h3value, h1value);
                endJoint2 = new Point3D(0, Ycoor, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     1,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     false,
                     false, false,
                     90, -90, 0, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Add other half side tie beams
                //Parte 1
                initialJoint1 = new Point3D(L1value, Ycoor - Yseparation, h1value - a1);
                endJoint1 = new Point3D(L1value, Ycoor - Yseparation + h3value, h1value -
h3value);
                initialJoint2 = new Point3D(L1value, Ycoor - Yseparation, h1value);
                endJoint2 = new Point3D(L1value, Ycoor - Yseparation + h3value, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     1,
```

```
                    SectionCName, SectionLName,
                    numberofCframes, numberofLframes,
                    true,
                    false, false,
                    90, -90, 0, 0,
                    SectionLName, SectionLName,
                    0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 2
                initialJoint1 = new Point3D(L1value, Ycoor - Yseparation + h3value,
h1value - h3value);
                endJoint1 = new Point3D(L1value, Ycoor - Yseparation / 2, h1value -
h3value);
                initialJoint2 = new Point3D(L1value, Ycoor - Yseparation + h3value,
h1value);
                endJoint2 = new Point3D(L1value, Ycoor - Yseparation / 2, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     nPartsCerchaAmarre,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     true,
                     true, false,
                     90, -90, 90, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 3
                initialJoint1 = new Point3D(L1value, Ycoor - Yseparation / 2, h1value -
h3value);
                endJoint1 = new Point3D(L1value, Ycoor - h3value, h1value - h3value);
                initialJoint2 = new Point3D(L1value, Ycoor - Yseparation / 2, h1value);
                endJoint2 = new Point3D(L1value, Ycoor - h3value, h1value);

                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     nPartsCerchaAmarre,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     false,
                     true, true,
                     90, -90, 90, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;

                //Parte 4
                initialJoint1 = new Point3D(L1value, Ycoor - h3value, h1value - h3value);
                endJoint1 = new Point3D(L1value, Ycoor, h1value - a1);
                initialJoint2 = new Point3D(L1value, Ycoor - h3value, h1value);
                endJoint2 = new Point3D(L1value, Ycoor, h1value);
```

```
                numberofFramesTuple = await generateTruss
                    (initialJoint1, endJoint1,
                     initialJoint2, endJoint2,
                     1,
                     SectionCName, SectionLName,
                     numberofCframes, numberofLframes,
                     false,
                     false, false,
                     90, -90, 0, 0,
                     SectionLName, SectionLName,
                     0, 0);
                numberofCframes = numberofFramesTuple.Item1;
                numberofLframes = numberofFramesTuple.Item2;
            }
        }
```

Anexo 6

```
public async Task RetrieveJointForcesForDesignFoundation()
        {
            JointList = await Joint.GetJointObjects(esaSap2000);
            JointForceResultsList = new List<IEnumerable<JointForce>>();

            //Save model
            await esaSap2000.FileSave();

            //Run analysis
            await esaSap2000.AnalyzeRunAnalysis();

            await esaSap2000.ResultsDeselectAllCasesAndCombosForOutput();
            await esaSap2000.ResultsSetCaseSelectedForOutput("CM");
            await esaSap2000.ResultsSetCaseSelectedForOutput("CV");
            await esaSap2000.ResultsSetCaseSelectedForOutput("Sx");
            await esaSap2000.ResultsSetCaseSelectedForOutput("Sy");
            await esaSap2000.ResultsSetComboSelectedForOutput("CM + CV");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.2D+1.6L");

            await esaSap2000.ResultsSetComboSelectedForOutput("0.9D+Sx");
            await esaSap2000.ResultsSetComboSelectedForOutput("0.9D-Sx");
            await esaSap2000.ResultsSetComboSelectedForOutput("0.9D+Sy");
            await esaSap2000.ResultsSetComboSelectedForOutput("0.9D-Sy");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.2D+L+Sx");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.2D+L-Sx");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.2D+L+Sy");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.2D+L-Sy");
            await esaSap2000.ResultsSetComboSelectedForOutput("1.4D");

            await esaSap2000.SetPresentUnits(eUnits.Ton_m_C);

            foreach (var jointName in JointList)
            {
                List<JointForce> jointForces = await
JointForce.GetJointForces(esaSap2000, jointName, eItemTypeElm.ObjectElm);
                JointForceResultsList.Add(jointForces);
            }
```

```
            await esaSap2000.SetModelIsLocked();

            int numberofTrusses = nParts4 + 1;
            int numberofCframesPerTruss = 4 * (nParts1 + nParts2 + nParts3) + 8;

            JointforcesList = new List<JointForces>();

            double Ycoor = 0;
            int numberofSpaces = nParts4;
            double Yseparation = L2value / numberofSpaces;


            for (int i = 1; i <= numberofTrusses; i++)
            {
                int Index = numberofCframesPerTruss * (i - 1);
                string frameName = "C-" + Convert.ToString(Index + 1);
                string Point1 = "";
                string Point2 = "";
                var tuple = await esaSap2000.FrameObjGetPoints(frameName, Point1,
Point2);
                Point1 = tuple.Item1;
                Point2 = tuple.Item2;

                double x1 = 0;
                double y1 = 0;
                double z1 = 0;
                double x2 = 0;
                double y2 = 0;
                double z2 = 0;
                var a1 = await esaSap2000.PointObjGetCoordCartesian(Point1, x1, y1, z1);
                var a2 = await esaSap2000.PointObjGetCoordCartesian(Point2, x2, y2, z2);
                x1 = a1.Item1;
                y1 = a1.Item2;
                z1 = a1.Item3;
                x2 = a2.Item1;
                y2 = a2.Item2;
                z2 = a2.Item3;

                double distance = Math.Sqrt(Math.Pow(x2 - x1, 2));

                double F3point1 = 0;
                double F3point2 = 0;
                double M1point1 = 0;
                double M1point2 = 0;
                double M2point1 = 0;
                double M2point2 = 0;

                double Fu3point1 = 0;
                double Fu3point2 = 0;
                double Mu1point1 = 0;
                double Mu1point2 = 0;
                double Mu2point1 = 0;
                double Mu2point2 = 0;

                List<JointForce> jointForce1 = new List<JointForce>();
                List<JointForce> jointForce2 = new List<JointForce>();

                for (int j = 0; j <= JointForceResultsList.Count - 1; j++)
                {
                    var a = JointForceResultsList.ElementAt(j);
```

```
                    foreach(var c in a)
                    {
                        if (c.LocalName == Point1 && c.LoadCase == "CM + CV")
                        {
                            F3point1 = c.F3;
                            M1point1 = c.M1;
                            M2point1 = c.M2;
                        }
                        else if (c.LocalName == Point2 && c.LoadCase == "CM + CV")
                        {
                            F3point2 = c.F3;
                            M1point2 = c.M1;
                            M2point2 = c.M2;
                        }

                        if (c.LocalName == Point1)
                        {
                            jointForce1 = a.ToList();
                        }
                        else if (c.LocalName == Point2)
                        {
                            jointForce2 = a.ToList();
                        }
                    }
                }

                for (int j = 0; j <= JointForceResultsList.Count - 1; j++)
                {
                    var a = JointForceResultsList.ElementAt(j);
                    foreach (var c in a)
                    {
                        if (c.LocalName == Point1 && c.LoadCase == "1.2D+1.6L")
                        {
                            Fu3point1 = c.F3;
                            Mu1point1 = c.M1;
                            Mu2point1 = c.M2;
                        }
                        else if (c.LocalName == Point2 && c.LoadCase == "1.2D+1.6L")
                        {
                            Fu3point2 = c.F3;
                            Mu1point2 = c.M1;
                            Mu2point2 = c.M2;
                        }
                    }
                }

                JointForces jointforces = new JointForces();
                jointforces.F3point1 = F3point1;
                jointforces.F3point2 = F3point2;
                jointforces.M2point1 = M2point1;
                jointforces.M2point2 = M2point2;
                jointforces.distance = distance;
                jointforces.Fu3point1 = Fu3point1;
                jointforces.Fu3point2 = Fu3point2;
                jointforces.Mu2point1 = Mu2point1;
                jointforces.Mu2point2 = Mu2point2;
                jointforces.CenterPoint = new Point3D(Math.Min(x1,x2) + distance / 2,
Ycoor, 0);
                jointforces.jointForces1 = jointForce1;
                jointforces.jointForces2 = jointForce2;
```

```
            JointforcesList.Add(jointforces);

            Ycoor += Yseparation;
        }
    }

    public struct JointForces
    {
        public double F3point1;
        public double F3point2;
        public double M1point1;
        public double M1point2;
        public double M2point1;
        public double M2point2;
        public double distance;
        public double Fu3point1;
        public double Fu3point2;
        public double Mu1point1;
        public double Mu1point2;
        public double Mu2point1;
        public double Mu2point2;
        public Point3D CenterPoint;
        public List<JointForce> jointForces1;
        public List<JointForce> jointForces2;
    }
```

Anexo 7

```
public bool CalculateShearStresses()
    {
        double Ma = Mux;
        double Mb = Muy;

        double q1a = Pu / CimArea + 6 * Ma / (FootingB * Math.Pow(FootingA, 2));
        double q2a = Pu / CimArea - 6 * Ma / (FootingB * Math.Pow(FootingA, 2));

        double q1b = Pu / CimArea + 6 * Mb / (FootingA * Math.Pow(FootingB, 2));
        double q2b = Pu / CimArea - 6 * Mb / (FootingA * Math.Pow(FootingB, 2));

        double volB = FootingB - pedestalB;
        double volA = (FootingA - pedestalA) / 2;
        double q3a = (q1a - q2a) * (FootingA - volA + d) / FootingA + q2a;
        double q3b = (q1b - q2b) * (FootingB - volB + d) / FootingB + q2b;

        double shearStressA = (q1a + q3a) * (volA - d) / (2 * d);
        double shearStressB= (q1b + q3b) * (volB - d) / (2 * d);
        double shearStress = Math.Max(shearStressA, shearStressB);

        double resistantStress = 0.53 * Math.Sqrt(fc) *10 * 0.75;

        bool ShearPassed = false;
        if  (resistantStress > shearStress)
        {
            ShearPassed = true;
        }
        else
```

```
            {
                d += 0.01;
                CalculateFootingHeight();
            }

            return ShearPassed;
        }


public bool CalculatePunching()
        {
            double bo = 2 * (pedestalB + dadoBoffset + d / 2) + (pedestalA + 2 *
dadoAoffset) + d;
            double Ap = FootingB * FootingA - (pedestalB + d / 2) * (pedestalA + d);

            double Ma = Mux;
            double Mb = Muy;

            double q1b = Pu / CimArea + 6 * Mb / (FootingA * Math.Pow(FootingB, 2));
            double q2b = Pu / CimArea - 6 * Mb / (FootingA * Math.Pow(FootingB, 2));

            double Fp = ((q1b + q2b) / 2) * CimArea;
            double PunchingStress = Fp / (bo * d);

            double ResistantStress1 = 1.06 * Math.Sqrt(fc) * 10 * 0.75;
            double ResistantStress2 = 0.26 * (2 + 4 * pedestalA / pedestalB) *
Math.Sqrt(fc) * 10 * 0.75;
            double alpha = 30; //Para zapatas medianeras
            double ResistantStress3 = 0.26 * ((alpha * d)/ bo + 2) * Math.Sqrt(fc) * 10 *
0.75;
            double ResistantStress = Math.Min(Math.Min(ResistantStress1,
ResistantStress2), ResistantStress3);

            bool PunchingPassed = false;
            if (ResistantStress > PunchingStress)
            {
                PunchingPassed = true;
            }
            else
            {
                dadoAoffset += 0.05;
                dadoBoffset += 0.05;
            }

            return PunchingPassed;
        }
```

Anexo 8

```
public void CalculateFootingSteel()
        {
            var tuple = CalculateFootingSteelByMoment(Muy, FootingA);
            numberofbarsA = tuple.Item1;
            sepA = tuple.Item2;

            numberofbarB = (int)Math.Ceiling((FootingB - 0.1) / (sepA / 100));
            sepB = (FootingB * 100 - 10 - 4 * bardiameter / 10) / (numberofbarB - 1);
        }

        public Tuple<int, double> CalculateFootingSteelByMoment(double Mu, double dim)
        {
            double Asmin = 0.0018 * dim * 100 * FootingH * 100; //[cm2]
            double As = Mu * 100000 / (0.9 * 0.9 * fy * d * 100);
            As = Math.Max(Asmin, As);

            List<double> barDiameters = new double[6] { 1.6, 2.0, 2.2, 2.5, 2.8, 3.2
}.ToList();
            List<double> barArea = new List<double>();
            for (int i = 0; i <= barDiameters.Count - 1; i++)
            {
                var area = Math.PI * Math.Pow(barDiameters.ElementAt(i), 2) / 4;
                barArea.Add(area);
            }

            int numberofbars = 0;
            double sep = 0;


            for (int i = 1; i <= barArea.Count; i++)
            {
                numberofbars = (int)Math.Round(As / barArea.ElementAt(i), 0);
                sep = (dim * 100 - 10 - barDiameters.ElementAt(i)) / (numberofbars - 1);

                if (sep > 20)
                {
                    bardiameter = barDiameters.ElementAt(i) * 10;
                    break;
                }
            }

            var tuple = Tuple.Create(numberofbars, sep);
            return tuple;
        }
```

Anexo 9

```
public void DesignBasePLate()
        {
            double Pumax = 0;
            for (int i = 1; i <= jointForces1.Count; i++)
            {
                var a = jointForces1.ElementAt(i - 1);
                var b = jointForces2.ElementAt(i - 1);
                var P = Math.Abs(a.F3 + b.F3);
                if (P > Pumax)
                {
                    Pumax = P;
                }
            }
            //Revision de la resistencia al contacto del concreto
            //Como primera instancia se asumira una placa base de dimensiones iguales al
pedestal
            double plateA = pedestalA;
            double plateB = pedestalB;
            double A1 = plateA * plateB * 10000;
            double A2 = pedestalA * pedestalB * 10000;
            double ratio = Math.Sqrt(A2 / A1);
            double Pr = 0.65 * 0.85 * fc * A1 * ratio;
            while (Pr < Pumax * 1000)
            {
                pedestalA += 0.05;
                pedestalB += 0.05;
                ratio = CalulateConcreteResistanceRatio(plateA, plateB);
                while (ratio > 2)
                {
                    plateA += 0.05;
                    plateB += 0.05;
                    ratio = CalulateConcreteResistanceRatio(plateA, plateB);
                }
                Pr = CalculateConcreteResistance(plateA, plateB);
                Trace.WriteLine("Pr " + Pr + " Pumax " + Pumax);
            }
            //Espesor de la placa
            double m = (plateA - 0.95 * plateCA) / 2;
            double n = (plateB - 0.8 * plateb1) / 2;
            double n2 = (Math.Sqrt(plateb1 * plateCA) / 4);
            double t = Math.Max(Math.Max(m, n), n2);
            BasePLateA = Math.Round(plateA,3);
            BasePlateB = Math.Round(plateB, 3);
            BasePLatet = Math.Round(t, 3);
        }

        public double CalulateConcreteResistanceRatio(double plateA, double plateB)
        {
            double A1 = plateA * plateB * 10000;
            double A2 = pedestalA * pedestalB * 10000;
            double ratio = Math.Sqrt(A2 / A1);
            return ratio;
        }

        public double CalculateConcreteResistance(double plateA, double plateB)
        {
            double A1 = plateA * plateB * 10000;
```

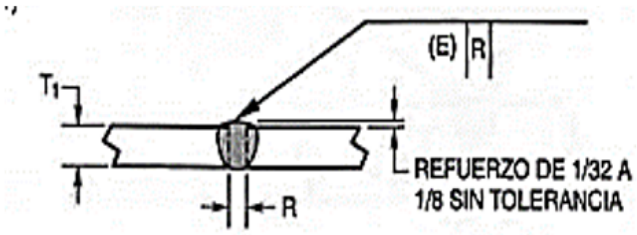
```
    double A2 = pedestalA * pedestalB * 10000;
    double ratio = Math.Sqrt(A2 / A1);

    double Pr = 0.65 * 0.85 * fc * A1 * ratio;
    return Pr;
}
```

Anexo 10

Nombre compañía:	Kevin Caicedo		
Norma:	AWS D1.1 Structural welding Code	**Código**:	Tesis-Wpn-001
Tipo de Junta:	A tope (B)	**Metal base**	
Metal de aporte		**Especif:**	A36
Espec SFA:	5.1	**Espesor:**	12 mm
Diámetro:	1.2 mm		
Tipo:	E6011	**Observación:**	Limpiar antes de soldar.

Detalle de junta

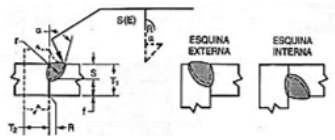


Proceso de soldadura	Tipo y polaridad de corriente	Designación de junta	Abertura de raíz (mm)	Posiciones de soldar	Tamaño de soldadura
SMAW	DC+	B-P1a	0.16	Todas	11.2 mm

Anexo 11

Nombre compañía:	MerchanVasquez		
Norma:	AWS D1.1 Structural welding Code	**Código**:	Tesis-Wpn-005
Tipo de Junta:	A tope (B), T(T), de esquina (C).	**Metal base**	
Metal de aporte		**Especif:**	A36
Espec SFA:	5.1	**Espesor:**	12 mm
Diámetro:	3.2mm		
Tipo:	E6011	**Observación:**	Limpiar antes de soldar.

Detalle de junta

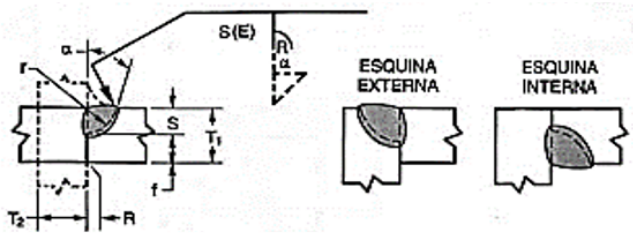


Proceso de soldadura	Tipo y polaridad de corriente	Designación de junta	Abertura de raíz (mm)	Posiciones de soldar	Tamaño de soldadura
SMAW	DC+	B-U5b	6	Todas	Varía

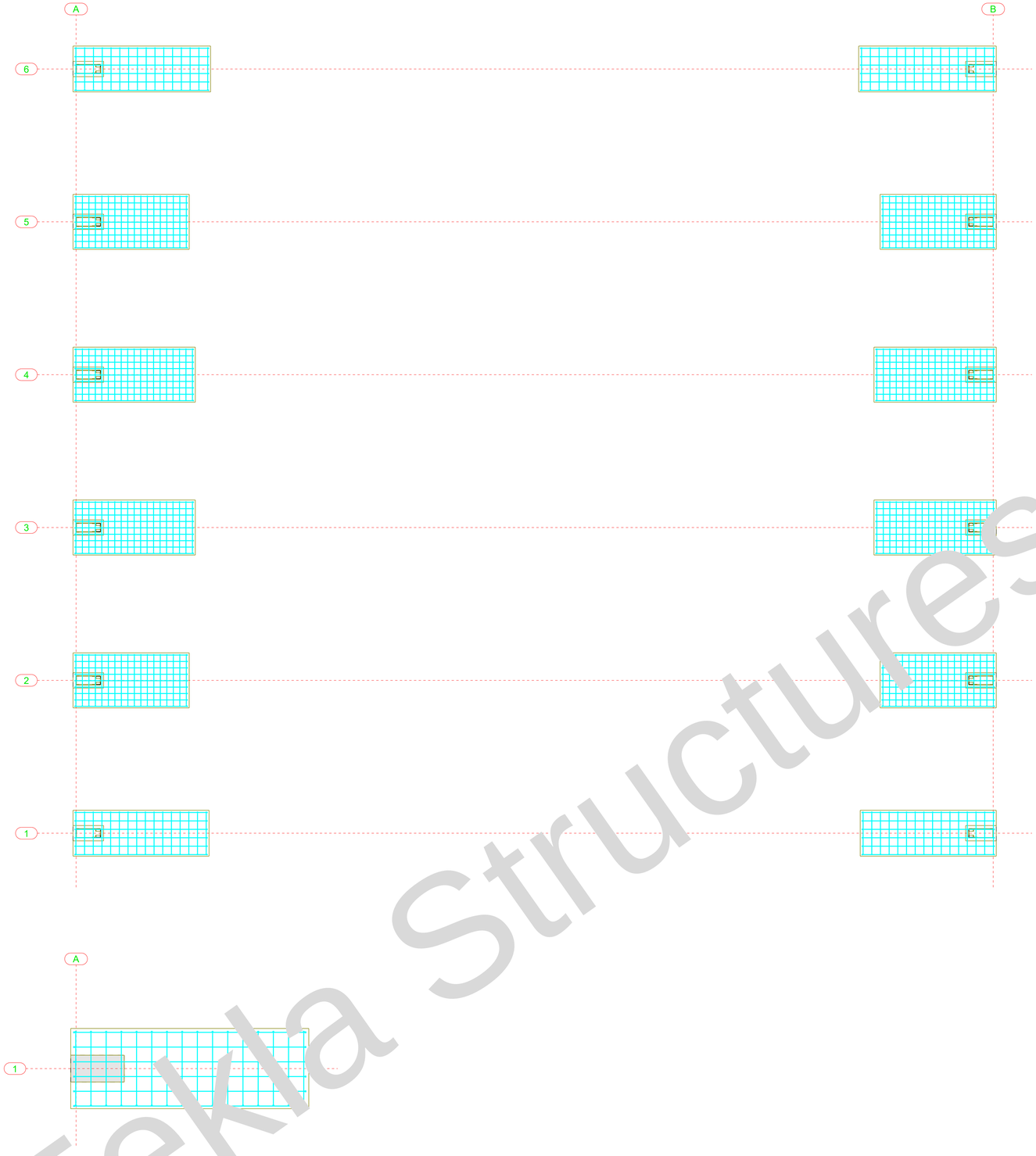
Anexo 12

Nombre compañía:	Kevin Caicedo		
Norma:	AWS D1.1 Structural welding Code	**Código**:	Tesis-Wpn-004
Tipo de Junta:	A tope (B), T(T), de esquina (C).	**Metal base**	
Metal de aporte		**Especif:**	A36
Espec SFA:	5.1	**Espesor:**	12 mm
Diámetro:	3.2mm		
Tipo:	ER70S-4	**Observación:**	Limpiar antes de soldar.
Gas de protección	CO2		
Resistencia a la tensión	70 KSI		

Detalle de junta



Proceso de soldadura	Designación de junta	Abertura de raíz (mm)	Posiciones de soldar	Tamaño de soldadura
SMAW	B-P8-GF	0.16	Todas	Varias



RECUBRIMIENTOS MÍNIMOS SEGÚN EL ACI 318-08
HORMIGÓN CONSTRUIDO EN SITIO (NO PRE... ZADO)

CASOS	(mm)
HORMIGÓN VACIADO DIRECTAMENTE EN EL SUELO ... EXPUE... O PERMANENTEMENTE EN ÉL.	75
HORMIGÓN EXPUESTO AL SUELO O A LA INT... ERIE	
BARRAS ø20mm Y MAYORES	50
BARRAS ø16mm Y MENORES	40
HORMIGÓN NO EXPUESTO A LA INTEMPE... NI EN CONTAC... ON EL SUELO	
LOSAS, MUROS, VIGUETAS:	
BARRAS ø35mm ...	20
VIGAS, COLUMNA...	
REFUERZO PR... IPAL	40
ESTRIBOS, E... ALES	40
CASCARAS, P... AS PLEGADAS:	
BARRAS ø20... Y MAYORES	20
BA... S ø16r... Y MENORES	15

ESPECI... CIONES

GA... HOS Y ...

GANCHOS A 180°

GANCHOS A 90°

(A*) = LONGITUD NECESARIA PARA FORMAR EL GANCHO

ø	D en cm.
3/8	10 60
1/2	12 60
5/8	16 80
3/4	10 80
7/8	22 80
1"	25 80
1 1/8	28 100

EN ESTRIBOS DOBLES EN ESTRIBO

8.0 7 cm. mínimo 135°

D=6ø

EN ELEMENTOS ESTRUCTURALES
RECUBRIMIENTO

4cm.

RECUBRIMIENTOS:
SUPERIOR= 4.0cm
LATERAL = 4.0cm
INFERIOR= 5.0cm

CON REPLANTILLO DE HORMIGON r=10cm.

SE USARÁ:

HORMIGON	f'c=210 Kg/cm²
ACERO	fy=4200Kg/cm²
PERFILES	ASTM A-36
SOLDADURA	NORMA: AWS D1.1 2006 TABLA 3.1 (Proceso/Soldadura)

REVENIMIENTO 10 a 12 cm
PINTURA ANTICORROSIVA CROMATO DE ZINC O SIMILAR.

NOTAS

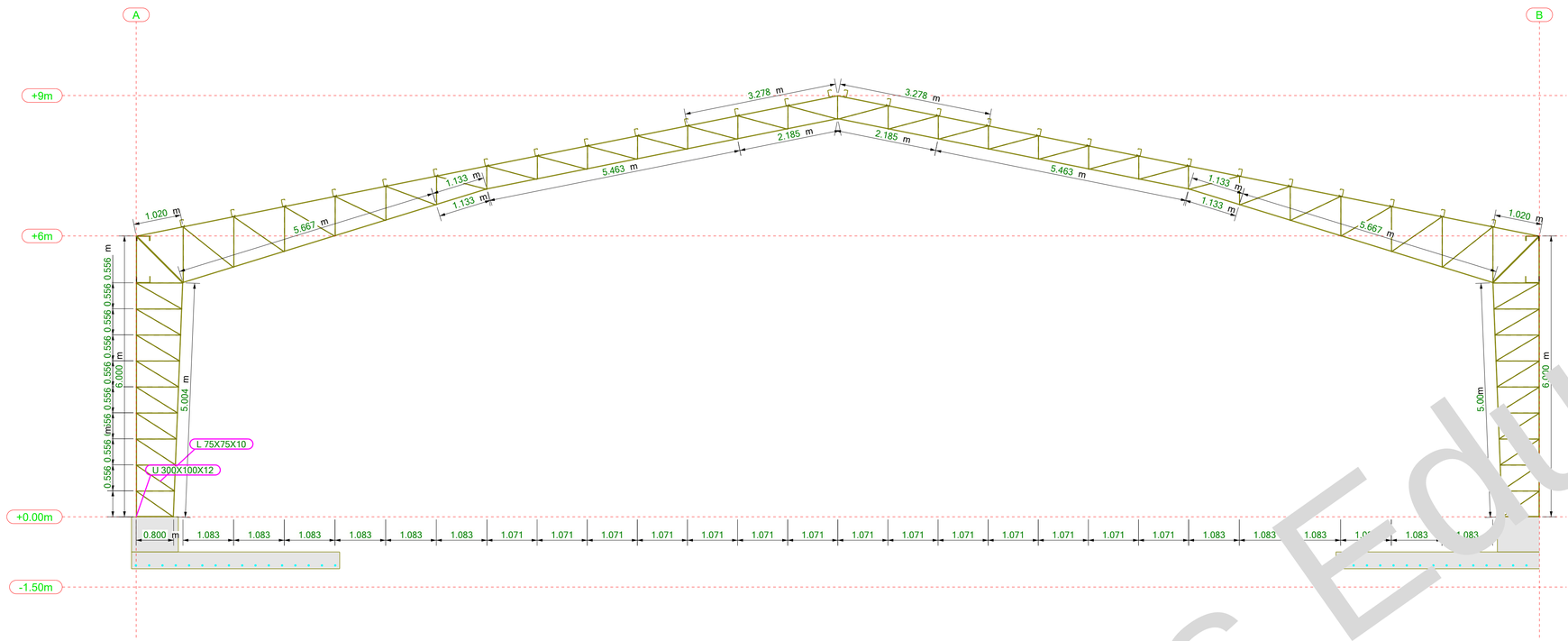
1. ANTES DE PROCEDER A LA CONSTRUCCION SE DEBERA CORRELACIONAR LOS PLANOS ESTRUCTURALES CON LOS PLANOS ARQUITECTONICOS.
2. LAS MEDIDAS PREVALECEN SOBRE LA ESCALA

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA

PROYECTO:
Diseño paramétrico

CONTENIDO:
Planta de cimenta*

Coordinador de Materia Integradora:	Tutores de Conocimiento Específicos:	Estudiante:	Fecha de emisión:	
PhD. Adrés Velástegui	- Msc. Carlos Quishpe - Msc. Jhonny Encalada - Arq. Eunice Lindao	Kevin Caicedo	17.08.2021	
Tutor de Área de Conocimiento:			Lámina: 4/6	Escala: 1:44
MSc. Carlos Quishpe				



RECUBRIMIENTOS MÍNIMOS SEGÚN EL ACI 318-08
HORMIGÓN CONSTRUIDO EN SITIO (NO PRE... ZADO)

CASOS	(mm)
HORMIGÓN VACIADO DIRECTAMENTE EN EL SUELO Y EXPUESTO PERMANENTEMENTE EN ÉL.	75
HORMIGÓN EXPUESTO AL SUELO O A LA INTEMPERIE	
BARRAS ø20mm Y MAYORES	50
BARRAS ø16mm Y MENORES	40
HORMIGÓN NO EXPUESTO A LA INTEMPERIE NI EN CONTACTO CON EL SUELO	
LOSAS, MUROS, VIGUETAS:	
BARRAS ø35mm...	20
VIGAS, COLUMNAS:	
REFUERZO PRINCIPAL	40
ESTRIBOS, ESPIRALES	40
CASCARAS, PLACAS PLEGADAS:	
BARRAS ø20... Y MAYORES	20
BARRAS ø16... Y MENORES	15

ESPECIFICACIONES

GANCHOS Y DOBLECES

GANCHOS A 180°

GANCHOS A 90°

ø	D en cm.
3/8	10 60
1/2	12 60
5/8	16 80
3/4	10 80
7/8	22 80
1"	25 80
1 1/8	28 100

(A*) = LONGITUD NECESARIA PARA FORMAR EL GANCHO

EN ESTRIBOS DOBLES EN ESTRIBO

8.0 7 cm. mínimo 135°

D=6ø

EN ELEMENTOS ESTRUCTURALES
RECUBRIMIENTO

4cm.

RECUBRIMIENTOS:

SUPERIOR= 4.0cm

LATERAL = 4.0cm

INFERIOR= 5.0cm

CON REPLANTILLO DE HORMIGON r=10cm.

SE USARÁ:

HORMIGON f'c=210 Kg/cm²

ACERO fy=4200Kg/cm²

PERFILES ASTM A-36

SOLDADURA NORMA: AWS D1.1 2006
TABLA 3.1 (Proceso/Soldadura)

REVENIMIENTO 10 a 12 cm

PINTURA ANTICORROSIVA CROMATO DE ZINC O SIMILAR.

NOTAS

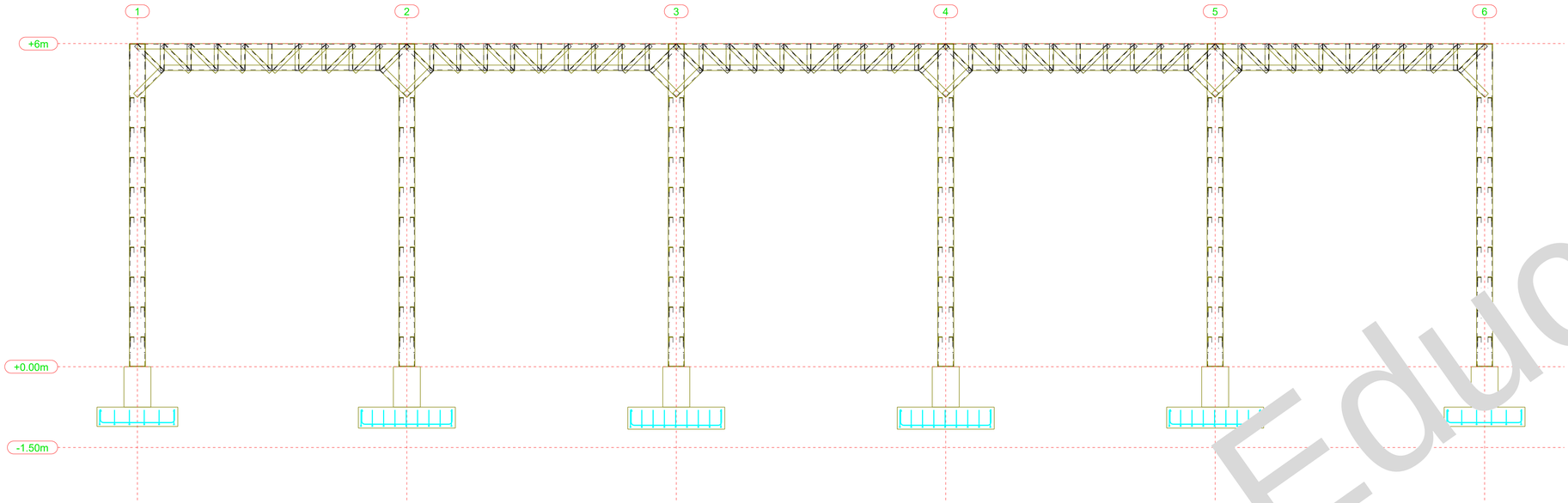
1. ANTES DE PROCEDER A LA CONSTRUCCION SE DEBERA CORRELACIONAR LOS PLANOS ESTRUCTURALES CON LOS PLANOS ARQUITECTONICOS.
2. LAS MEDIDAS PREVALECEN SOBRE LA ESCALA

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA

PROYECTO:
Diseño paramétrico

CONTENIDO:
Vista Frontal

Coordinador de Materia Integradora:	Tutores de Conocimiento Específicos:	Estudiante:	Fecha de emisión:	
PhD. Adrés Velástegui	- Msc. Carlos Quishpe - Msc. Jhonny Encalada - Arq. Eunice Lindao	Kevin Caicedo	17.08.2021	
Tutor de Área de Conocimiento:			Lámina: 1/6	Escala: 1:58
MSc. Carlos Quishpe				



RECUBRIMIENTOS MÍNIMOS SEGÚN EL ACI 318-08
HORMIGÓN CONSTRUIDO EN SITIO (NO PRE...ZADO)

CASOS	(mm)
HORMIGÓN VACIADO DIRECTAMENTE EN EL SUELO Y EXPUESTO PERMANENTEMENTE EN ÉL.	75
HORMIGÓN EXPUESTO AL SUELO O A LA INTEMPERIE	
BARRAS Ø20mm Y MAYORES	50
BARRAS Ø16mm Y MENORES	40
HORMIGÓN NO EXPUESTO A LA INTEMPERIE NI EN CONTACTO CON EL SUELO	
LOSAS, MUROS, VIGUETAS:	
BARRAS Ø35mm...	20
VIGAS, COLUMNAS:	
REFUERZO PRINCIPAL	40
ESTRIBOS, ESPIRALES	40
CASCARAS, PLACAS PLEGADAS:	
BARRAS Ø20... Y MAYORES	20
BARRAS Ø16... Y MENORES	15

ESPECIFICACIONES

GANCHOS Y DOBLECES

GANCHOS A 180°

GANCHOS A 90°

(A*) = LONGITUD NECESARIA PARA FORMAR EL GANCHO

ø	D en cm.
3/8	10 60
1/2	12 60
5/8	16 80
3/4	18 80
7/8	22 80
1"	25 80
1 1/8	28 100

EN ESTRIBOS

DOBLES EN ESTRIBO

8.0 cm. mínimo

D=6ø

EN ELEMENTOS ESTRUCTURALES
RECUBRIMIENTO

4cm.

RECUBRIMIENTOS:

SUPERIOR= 4.0cm

LATERAL = 4.0cm

INFERIOR= 5.0cm

CON REPLANTILLO DE HORMIGON r=10cm.

SE USARÁ:

HORMIGON f'c=210 Kg/cm²

ACERO fy=4200Kg/cm²

PERFILES ASTM A-36

SOLDADURA NORMA: AWS D1.1 2006
TABLA 3.1 (Proceso/Soldadura)

REVENIMIENTO 10 a 12 cm

PINTURA ANTICORROSIVA CROMATO DE ZINC O SIMILAR.

NOTAS

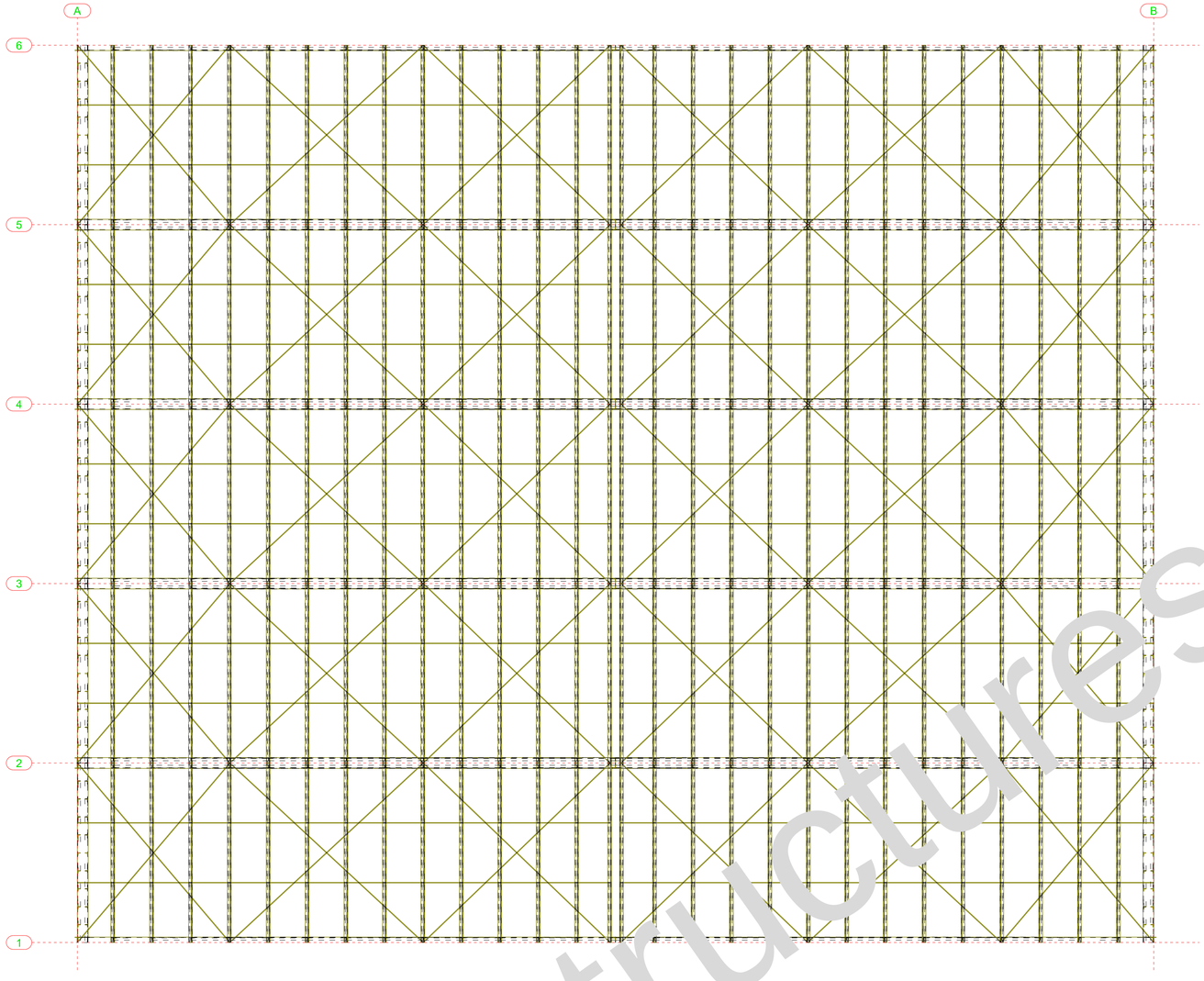
1. ANTES DE PROCEDER A LA CONSTRUCCION SE DEBERA CORRELACIONAR LOS PLANOS ESTRUCTURALES CON LOS PLANOS ARQUITECTONICOS.
2. LAS MEDIDAS PREVALECEN SOBRE LA ESCALA

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA

PROYECTO:
Diseño paramétrico

CONTENIDO:
Vista Lateral

Coordinador de Materia Integradora:	Tutores de Conocimiento Específicos:	Estudiante:	Fecha de emisión:	
PhD. Adrés Velástegui	- Msc. Carlos Quishpe - Msc. Jhonny Encalada - Arq. Eunice Lindao	Kevin Caicedo	17.08.2021	
Tutor de Área de Conocimiento:			Lámina: 2/6	Escala: 1:48
MSc. Carlos Quishpe				

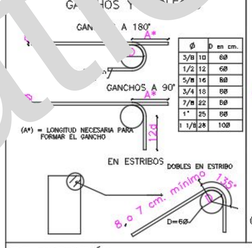


RECUBRIMIENTOS MÍNIMOS SEGÚN EL ACI 318-08
HORMIGÓN CONSTRUIDO EN SITIO (NO PRE... ZADO)

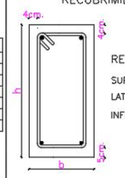
CASOS	(mm)
HORMIGÓN VACIADO DIRECTAMENTE EN EL SUELO ...XPUE... O PERMANENTEMENTE EN ÉL.	75
HORMIGÓN EXPUESTO AL SUELO O A LA INT... ERIE	
BARRAS ø20mm Y MAYORES	50
BARRAS ø16mm Y MENORES	40
HORMIGÓN NO EXPUESTO A LA INTEMPE... NI EN CONTAC... ON EL SUELO	
LOSAS, MUROS, VIGUETAS:	
BARRAS ø35mm...	20
VIGAS, COLUMNA...	
REFUERZO PR... IPAL	40
ESTRIBOS, E... ALES	40
CASCARAS, P... AS PLEGADAS:	
BARRAS ø20... Y MAYORES	20
BA... S ø16r... Y MENORES	15

ESPECI... CIONES

GA... HOS Y ...



EN ELEMENTOS ESTRUCTURALES
RECUBRIMIENTO



RECUBRIMIENTOS:
SUPERIOR= 4.0cm
LATERAL = 4.0cm
INFERIOR= 5.0cm

CON REPLANTILLO DE HORMIGON r=10cm.

SE USARÁ:

HORMIGON	f'c=210 Kg/cm²
ACERO	fy=4200Kg/cm²
PERFILES	ASTM A-36
SOLDADURA	NORMA: AWS D1.1 2006 TABLA 3.1 (Proceso/Soldadura)

REVENIMIENTO 10 a 12 cm
PINTURA ANTICORROSIVA CROMATO DE ZINC O SIMILAR.

NOTAS

1. ANTES DE PROCEDER A LA CONSTRUCCION SE DEBERA CORRELACIONAR LOS PLANOS ESTRUCTURALES CON LOS PLANOS ARQUITECTONICOS.
2. LAS MEDIDAS PREVALECEN SOBRE LA ESCALA

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA

PROYECTO:
Diseño paramétrico

CONTENIDO:
Vista Superior

Coordinador de Materia Integradora:	Tutores de Conocimiento Específicos:	Estudiante:	Fecha de emisión:
PhD. Adrés Velástegui	- Msc. Carlos Quishpe - Msc. Jhonny Encalada	Kevin Caicedo	17.08.2021
Tutor de Área de Conocimiento:	- Arq. Eunice Lindao		Lámina: 3/6 / Escala: 1:77
MSc. Carlos Quishpe			

CARTA DE APROBACIÓN POR CADA TUTOR

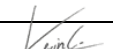
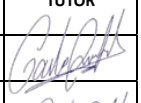
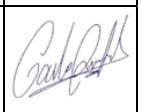
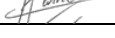
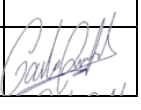
Guayaquil, 7 de septiembre del 2021.

PhD. Miguel Ángel Chávez Moncayo
Coordinador de Materia Integradora de la carrera de Ingeniería Civil
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA
Presente.

Por este medio, le informo que he realizado la revisión del proyecto de materia integradora con título Diseño paramétrico de la estructura de una nave industrial, del estudiante Kevin Rafael Caicedo Cárdenas en la componente de Conocimiento.

Donde doy fe, que los estudiantes han realizado una continua revisión de esta componente en diversas reuniones planificadas.

Adjunto informe.

INFORME PARCIAL DE SEGUIMIENTO DE ACTIVIDADES TUTOREADAS					
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA					
DOCENTE:	Ph.D. Miguel Chávez	**TUTOR:**	MSc. Carlos Quishpe	**CARRERA:**	Ingeniería Civil
NOMBRES Y APELLIDOS DE LOS ESTUDIANTES	**FECHA**	**ACTIVIDADES REALIZADAS**	**SUGERENCIAS Y RECOMENDACIONES**	**FIRMA DE ESTUDIANTE**	**FIRMA DEL TUTOR**
Kevin Rafael Caicedo Cárdenas	21/4/2021	Revisión del primer avance del desarrollo de la aplicación.			
Kevin Rafael Caicedo Cárdenas	1/5/2021	Se revisó el modelo en Tekla Structures y se plantearon alternativas de solucción respecto a qué programa utilizar para el diseño y detallamiento.			
Kevin Rafael Caicedo Cárdenas	10/5/2021	Revisión del prediseño.	Se sugiere añadir nuevos parámetros en el diseño del galpón como: altura de viga de amarre, número de pórticos, ancho de cercha, etc.		
Kevin Rafael Caicedo Cárdenas	19/5/2021	Revisión del prediseño.	Se reviso la metología del cálculo de los perfiles en el prediseño. Se sugiere programar la visualización del modelo de prediseño.		
Kevin Rafael Caicedo Cárdenas	26/5/2021	Revisión del prediseño.	Se sugiere implementar una vista previa del galpón en la ventana principal del programa .		

INFORME PARCIAL DE SEGUIMIENTO DE ACTIVIDADES TUTOREADAS					
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA					
DOCENTE:	Ph.D. Miguel Chávez	**TUTOR:**	MSc. Carlos Quishpe	**CARRERA:**	Ingeniería Civil
NOMBRES Y APELLIDOS DE LOS ESTUDIANTES	**FECHA**	**ACTIVIDADES REALIZADAS**	**SUGERENCIAS Y RECOMENDACIONES**	**FIRMA DE ESTUDIANTE**	**FIRMA DEL TUTOR**
Kevin Rafael Caicedo Cárdenas	3/6/2021	Revisión de cambios en la aplicación.	Se recomienda implementar una vista de ejemplo del galpón donde se ilustren los parámetros.		
Kevin Rafael Caicedo Cárdenas	10/6/2021	Revisión de cambios en la aplicación.	Se recomienda cambiar cierta parte del galpón para que sea modelado como pórtico a momentos.		
Kevin Rafael Caicedo Cárdenas	18/6/2021	Revisión de cambios en la aplicación.	Se recomienda eliminar los tensores laterales del modelo e implementar una opción que permita elegir los perfiles comerciales dentro de la base de datos.		
Kevin Rafael Caicedo Cárdenas	24/6/2021	Revisión presentación parcial.	Se sugiere mejorar la presentación de las diapositvas, resaltar palabras claves y añadir audio al video del final.		
Kevin Rafael Caicedo Cárdenas	8/7/201	Revisión del diseño estructural	Se sugiere mejorar el algoritmo de diseño automatico para que disminuya el espesor de los cordones en ciertos tramos de las cerchas		
Kevin Rafael Caicedo Cárdenas	16/7/2021	Revisión del diseño de las placas de anclaje			
Kevin Rafael Caicedo Cárdenas	23/7/2021	Revisión de la modelación de la estructura en Tekla Structures	Se recomienda grabar la modelación automática del galpón para ser mostrado en la presentación final.		
Kevin Rafael Caicedo Cárdenas	31/72021	Revisión de la generación de los planos en Tekla Structures	Se recomienda aumentar la cantidad de vistas generadas por		
Kevin Rafael Caicedo Cárdenas	5/8/2021	Correccion del modelado en Tekla Structures	De ser posible, que se permita visualizar el armado de las zapatas en Tekla Structures		
Kevin Rafael Caicedo Cárdenas	14/8/2021	Revisión de las correciones en la generación de Planos			
Kevin Rafael Caicedo Cárdenas	17/8/2021	Revisión de la memoria técnica	Se recomienda profundizar el apartado de metogología. Detallar mejor el proceso seguido del proyecto.		
Kevin Rafael Caicedo Cárdenas	28/8/2021	Revisisón del funcionamiento del apartado de presupuesto en la aplicación	El apartado de presuspuesto se genera correctamente, y es intuitivo de usar. Sin recomendaciones.		
Kevin Rafael Caicedo Cárdenas	4/9/2021	Revisión de la presentación del 5min Pitch	Se realizó sugerencias respecto al contenido de las diapositivas de la presentación del 5min pitch. Mejorar la habilidad de marketing del producto.		

Siempre agradecidos con usted.

Atte.
M.Sc. Carlos Paul Quishpe Otacoma
TUTOR MATERIA INTEGRADORA
ÁREA: CONOCIMIENTO
Correo: cquishpe@espol.edu.ec
Teléfono: 099 990 3586

CARTA DE APROBACIÓN POR CADA TUTOR

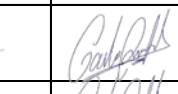
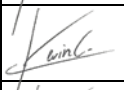
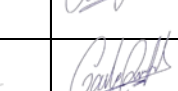
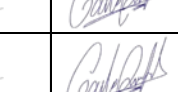
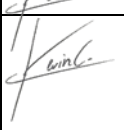
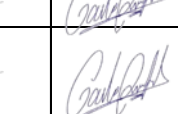
Guayaquil, 7 de septiembre del 2021.

PhD. Miguel Ángel Chávez Moncayo
Coordinador de Materia Integradora de la carrera de Ingeniería Civil
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA
Presente.

Por este medio, le informo que he realizado la revisión del proyecto de materia integradora con título Diseño paramétrico de la estructura de una nave industrial, del estudiante Kevin Rafael Caicedo Cárdenas en la componente de Presupuesto.

Donde doy fe, que los estudiantes han realizado una continua revisión de esta componente en diversas reuniones planificadas.

Adjunto informe.

INFORME PARCIAL DE SEGUIMIENTO DE ACTIVIDADES TUTOREADAS					
FACULTAD DE INGENIERÍA EN CIENCIAS DE LA TIERRA					
DOCENTE:	Ph.D. Miguel	**TUTOR:**	MSc. Carlos Quishpe	**CARRERA:**	Ingeniería Civil
NOMBRES Y APELLIDOS DE LOS ESTUDIANTES	**FECHA**	**ACTIVIDADES REALIZADAS**	**SUGERENCIAS Y RECOMENDACIONES**	**FIRMA DE ESTUDIANTE**	**FIRMA DEL TUTOR**
Kevin Rafael Caicedo Cárdenas	10/5/2021	Primera revisión de rublos	Se recomienda dividir los rublos en las categorías, preliminares, estructura, mampostería, varios.		
Kevin Rafael Caicedo Cárdenas	26/5/2021	Revisión de lista de rublos	Se recomienda investigar los materiales, mano de obra y equipo necesario para cada rublo actual.		
Kevin Rafael Caicedo Cárdenas	10/6/2021	Revisión de Apus	Se sugiere cambiar algunas cuadrillas en los rublos de estructuras.		
Kevin Rafael Caicedo Cárdenas	24/6/2021	Revisión de la vista inicial del presupuesto en el programa	Se sugiere implementar un excel dentro del programa que permita visualizar cada uno de los APUS.		
Kevin Rafael Caicedo Cárdenas	16/7/2021	Revisión de lista de rubros	Añadir e investigar rubros de placa de anclaje, suministro de acero y montaje de estructura.		
Kevin Rafael Caicedo Cárdenas	31/7/2021	Revisión del cálculo de cantidades para el presupuesto	Corregir el rubro de Caseta de guardianía. Debe de ser un porcentaje del area total de la		
Kevin Rafael Caicedo Cárdenas	7/8/2021	Revisión del análisis de precios unitarios de los rublos	Se recomienda corregir ciertos APUS cuyo costo no corresponde con la necesidad del proyecto.		
Kevin Rafael Caicedo Cárdenas	21/8/2021	Revisión del cronograma valorado	Se dieron sugerencias de la duración de ciertas actividades del cronograma para así calcular los porcentajes del precio por semana de cada actividad.		

Siempre agradecidos con usted.

Atte.
M.Sc. Carlos Paul Quishpe Otacoma
TUTOR MATERIA INTEGRADORA
ÁREA: PRESUPUESTO
Correo: cquishpe@espol.edu.ec
Teléfono: 099 990 3586