Fecha: 30 de marzo del 2024

Nombre del Grupo: US

Tema de Investigación: Cotización con una IA Generativa

Coordinador de Grupo: Christopher Monroy

Carné	Apellidos	Nombre	Email	Trabajó (Si/No)	Páginas
202400860	Monroy Maldonado	Christopher Alejandro	cmonroymaldonado@gmail.com	Si	20
202406827	Sic Sor	Cristopher Pablo Esteban	c.sic.4bdb@gmail.com	Si	49

Índice

Introducción

Exploramos los requerimientos fundamentales para el desarrollo de software que permita la creación automatizada de cotizaciones para obras de ingeniería civil. Este software, diseñado para operar en un entorno familiar como Excel, deberá ser capaz de gestionar una amplia variedad de datos relacionados con materiales de construcción, unidades de trabajo y precios asociados. La interfaz o hoja electrónica desarrollada deberá ofrecer funcionalidades completas para ingresar, almacenar, corregir y eliminar datos, además de generar cotizaciones detalladas que incluyan información clave como el nombre de la obra, lugar, descripción, cantidad de materiales, costos totales, porcentajes de utilidad, impuestos y precios finales.

Resumen

El proyecto de investigación que emplea una Inteligencia Artificial Generativa en el campo de la ingeniería civil representa un avance significativo en la automatización y optimización de procesos en la industria de la construcción. Al combinar técnicas de aprendizaje automático con instrucciones específicas, se busca revolucionar la manera en que se realizan las estimaciones de costos,

con el objetivo de mejorar la eficiencia, precisión y competitividad en la planificación y ejecución de proyectos de construcción.

La introducción de una Inteligencia Artificial Generativa mediante el uso de instrucciones para la creación de cotizaciones se presenta como una innovación prometedora que pretende potenciar notablemente la eficiencia y calidad de dichas estimaciones. Se espera que al proporcionar instrucciones detalladas mediante solicitudes bien diseñadas, la IA pueda generar cotizaciones completas y detalladas, que incluyan información precisa sobre los materiales requeridos, los costos asociados, la mano de obra necesaria y otros aspectos relevantes para llevar a cabo el proyecto. Esto facilitaría a los profesionales de la construcción obtener estimaciones más precisas en un menor tiempo, aliviando la carga administrativa y liberando recursos para labores más estratégicas.

Además, se anticipa que la utilización de una Inteligencia Artificial Generativa para la elaboración de cotizaciones mejorará la consistencia y calidad del proceso, reduciendo al mínimo los errores humanos y asegurando coherencia en los criterios de cálculo y estimación.

Planteamiento del problema

Cómo creamos una cotización acerca de productos de construcción utilizando una IA Generativa usando prompts.

Variable independiente

Utilización de IA generativa con prompts: Es la variable independiente que representa el uso de una IA generativa, como lo es Chat GPT, con prompts específicos para generar cotizaciones de productos de construcción.

Variables Dependientes

1) Eficiencia del proceso de cotización: Se refiere a la capacidad de la IA generativa para agilizar el proceso de cotización, minimizando el tiempo y los recursos necesarios para generar cotizaciones precisas.

2) Adaptabilidad a diferentes tipos de proyectos de construcción: Esta variable indica la capacidad de la IA generativa para generar cotizaciones precisas y relevantes para una variedad de proyectos de construcción, desde obras residenciales hasta proyectos comerciales o industriales.

3) Precisión de las cotizaciones generadas: Esta variable dependiente se refiere a la exactitud y fiabilidad de las cotizaciones generadas por la IA generativa en respuesta a los prompts proporcionados.

Antecedentes:

Los avances en la Inteligencia Artificial, particularmente en modelos generativos como GPT, han demostrado su capacidad para comprender y producir texto de manera similar a los humanos. En el sector de la construcción, donde se busca mejorar la eficiencia y la precisión a través de la

tecnología, esta capacidad se ha aplicado con éxito en diversas áreas, desde la planificación hasta la gestión de proyectos.

Justificación:

Es esencial en el ámbito de la construcción elaborar cotizaciones precisas y eficientes para los productos necesarios. No obstante, el proceso tradicional puede ser largo y propenso a errores, ya que implica reunir información, calcular costos y generar la cotización final. El uso de la Inteligencia Artificial Generativa, especialmente a través de prompts, ofrece una solución innovadora a este desafío al automatizar gran parte del proceso y permitir la generación de cotizaciones más rápidas y personalizadas.

Alcances:

1. Automatización de procesos: Mediante el uso de IA generativa y prompts, es posible automatizar la recopilación de información sobre los productos de construcción necesarios para un proyecto específico, generando una lista detallada de materiales de manera eficiente.
2. Cotizaciones personalizadas: La IA puede generar cotizaciones adaptadas a las necesidades individuales de cada proyecto, considerando detalles como tipo de construcción, ubicación, materiales y plazos de entrega, lo que permite ofrecer cotizaciones precisas a los clientes.
3. Reducción de errores: Al automatizar gran parte del proceso de cotización, se minimiza la posibilidad de errores humanos, mejorando

así la precisión y evitando costos adicionales y retrasos en el proyecto debido a estimaciones incorrectas.

4. Optimización del tiempo: La generación automatizada de cotizaciones a través de IA permite ahorrar tiempo significativo en comparación con los métodos tradicionales, lo que libera recursos para tareas más estratégicas y mejora la eficiencia operativa.

Objetivos generales

El objetivo principal de este proyecto de investigación es desarrollar un sistema basado en inteligencia artificial generativa, utilizando prompts específicos, para automatizar y mejorar el proceso de creación de cotizaciones en obras de ingeniería civil. El sistema buscará agilizar la generación de cotizaciones precisas, adaptadas a diferentes tipos de proyectos de construcción, con el fin de aumentar la eficiencia, la precisión y la competitividad en el sector de la construcción.

Objetivos Específicos

1. Investigar y analizar el estado del arte de la inteligencia artificial generativa y su aplicación en la creación de cotizaciones en el sector de la construcción.
2. Identificar los requerimientos y especificaciones necesarios para el desarrollo de un sistema basado en IA generativa con prompts para la generación automatizada de cotizaciones en obras de ingeniería civil.

3. Diseñar y desarrollar un software que integre la IA generativa y prompts específicos para la creación automatizada de cotizaciones, considerando la eficiencia del proceso, la adaptabilidad a diferentes tipos de proyectos y la precisión de las cotizaciones generadas.

Marco Teórico

¿Qué es una cotización y cómo se usan para cotizar las obras?

(Recuperado de https://enciclopediaeconomica.com/cotizacion/ el 11 de enero del año 2023)

Una cotización, en su significado más profundo, trasciende la simple asignación de un precio a un bien o servicio ofrecido por una empresa o entidad. Más bien, representa un proceso de evaluación minuciosa que busca determinar de manera detallada y precisa el costo asociado con la transacción en cuestión. Esta estimación no se limita únicamente a fijar un precio; su propósito fundamental radica en proporcionar una visión clara y transparente de todos los costos involucrados en la transacción.

Al elaborar una cotización, se lleva a cabo un análisis exhaustivo que abarca diversos aspectos, desde el costo de los materiales y la mano de obra hasta los gastos indirectos y los márgenes de beneficio. Cada elemento se desglosa y se evalúa cuidadosamente para garantizar que la cifra final refleje de manera precisa la inversión requerida por parte del cliente.

Esta transparencia en la información facilita enormemente el proceso de negociación entre las partes interesadas. Tanto el cliente como la empresa proveedora pueden comprender completamente los costos y las implicaciones

financieras de la transacción, lo que crea una base sólida para llegar a acuerdos mutuamente beneficiosos. Además, al tener una visión clara de los costos desde el principio, se reducen las posibilidades de malentendidos o disputas durante el proceso de ejecución del proyecto.

En el ámbito de la construcción y proyectos de ingeniería, la elaboración de una cotización va más allá de simplemente estimar el costo de los materiales y la mano de obra. Se trata de un proceso exhaustivo que abarca diversos aspectos esenciales para la realización exitosa del proyecto. Además del coste económico, una cotización integral considera el tiempo estimado de ejecución, los recursos necesarios, así como cualquier otro detalle relevante que pueda afectar el desarrollo y resultado final del proyecto. La inclusión de estos elementos adicionales en la cotización permite a los clientes obtener una visión completa y detallada de lo que implica el trabajo propuesto.

¿Qué es Chat GPT y cómo se usará dentro del proyecto?

(Recuperado de https://www.xataka.com/basics/chatgpt-que-como-usarlo-que-puedes-hacer-este-chat-inteligencia-artificial el 26 de marzo del año 2024)

Chat GPT, desarrollado por OpenAL, es un sistema de conversación basado en el modelo de lenguaje GPT-3.5, que cuenta con más de 175 millones de parámetros. Este modelo ha sido entrenado con enormes volúmenes de texto para realizar diversas tareas relacionadas con el lenguaje, como traducción y generación de texto.

El proceso de entrenamiento de una inteligencia artificial como Chat GPT implica exponerse a textos variados y hacerle preguntas para que aprenda a

responder de manera adecuada. Con el tiempo y la retroalimentación, el sistema se "entrena" para mejorar su capacidad de mantener conversaciones coherentes y responder preguntas de manera precisa.

Las posibilidades de uso de Chat GPT son diversas y se adaptan a nuestras necesidades y objetivos específicos. Podemos aprovechar su capacidad para proporcionar información detallada y precisa en una amplia gama de temas, lo que nos permite obtener conocimientos y realizar análisis según nuestras necesidades particulares.

En términos generales, los sistemas de IA funcionan al analizar grandes volúmenes de datos de entrenamiento etiquetados, buscando correlaciones y patrones, y utilizando esta información para realizar predicciones sobre futuros eventos. De este modo, un chatbot que recibe ejemplos de texto puede aprender a mantener conversaciones realistas con personas, o una herramienta de reconocimiento de imágenes puede aprender a identificar y describir objetos revisando una gran cantidad de ejemplos. La automatización de procesos con inteligencia artificial está transformando la eficiencia empresarial en diversos sectores, como el financiero, sanitario y minorista. La IA puede automatizar tareas que antes requerían intervención humana, lo que aumenta la eficiencia, mejora la productividad y reduce costos.

En el contexto de la generación de cotizaciones para obras de construcción, la IA puede aplicarse mediante modelos de lenguaje como Chat GPT para interpretar y generar respuestas basadas en inputs de usuario. Estos modelos pueden analizar grandes conjuntos de datos en tiempo real, identificar patrones y predecir tendencias futuras. Además, los sistemas de aprendizaje automático

permiten a las máquinas mejorar su rendimiento con el tiempo, ajustando sus procesos según la retroalimentación y las nuevas experiencias.

La automatización de procesos con IA no solo reduce la carga de trabajo manual, sino que también contribuye a una toma de decisiones más eficiente. Sin embargo, es esencial seleccionar proveedores de IA confiables y seguir las mejores prácticas de seguridad para proteger la integridad y confidencialidad de los datos.

En nuestro plan, utilizaremos Chat GPT como una herramienta para mejorar la comunicación con los clientes y recopilar los datos esenciales necesarios para estimar el costo de un proyecto de construcción. Implementaremos una interfaz de chat diseñada para ser fácil de usar por los usuarios, donde Chat GPT hará preguntas específicas para recopilar los detalles necesarios de manera clara y efectiva.

Una vez recopilada toda la información necesaria, se almacenará en una base de datos centralizada, utilizando herramientas como Microsoft Excel debido a su capacidad para manejar grandes volúmenes de datos de manera organizada y su familiaridad para una amplia audiencia. Posteriormente, el sistema calculará automáticamente el presupuesto de construcción, estimando los costos de materiales, impuestos y otros gastos relevantes, proporcionando a los clientes una cifra precisa y detallada.

¿Qué es un prompt de Chat GPT y cómo usar los prompts dentro del proyecto? (Recuperado de

https://www.searchenginejournal.com/how-to-write-chatgpt-prompts/479324/ el 3 de mayo del año 2023)

Los prompts de Chat GPT no se limitan a simples sugerencias; representan la puerta de entrada a un vasto mundo de posibilidades creativas y diálogos significativos. Estas indicaciones marcan el inicio de una interacción dinámica entre el usuario y el modelo de inteligencia artificial, donde se fomenta la expresión creativa, el intercambio de ideas y la exploración de nuevos conceptos.

Imaginarlos como el punto de partida de un viaje intelectual, donde tanto el usuario como Chat GPT se sumergen en una travesía hacia la exploración de ideas, la resolución de problemas o simplemente la generación de contenido fascinante. Pueden adoptar diversas formas: desde preguntas provocativas que despiertan la curiosidad hasta afirmaciones que invitan a una profunda reflexión, o cualquier otro estímulo diseñado para estimular la mente y el espíritu.

La verdadera riqueza de los prompts de Chat GPT radica en su capacidad de adaptarse a una amplia variedad de temas e intereses. Desde discusiones sobre ciencia y tecnología hasta debates sobre arte y cultura, los prompts pueden ser personalizados para satisfacer las necesidades y preferencias únicas de cada usuario. Esta capacidad de personalización garantiza que cada interacción con Chat GPT sea única y relevante para el individuo que lo inicia.

Además de ser herramientas para la creatividad y la exploración intelectual, los prompts de Chat GPT desempeñan un papel fundamental en proyectos como el mencionado. Al interactuar con el usuario, Chat GPT puede recopilar de

manera completa todas las medidas necesarias para calcular los materiales y la mano de obra requeridos en la ejecución del proyecto de construcción. Esta interacción fluida y eficiente facilita una comunicación efectiva entre el usuario y la inteligencia artificial, agilizando así el proceso de planificación y estimación de costos para el proyecto en cuestión.El uso de prompts posibilitará que ChatGPT solicite al usuario información detallada sobre las dimensiones de las paredes, teniendo en cuenta aspectos como la longitud, altura y cualquier otro detalle relevante. Este enfoque asegurará la captura completa de los datos necesarios para calcular con precisión la cantidad de materiales y mano de obra requeridos para llevar a cabo la construcción.

Asimismo, se planea incluir información adicional en la cotización, como el nombre de la empresa, dirección, número de teléfono, sitio web y dirección de correo electrónico. Esta información también se recopiló a través de prompts, garantizando la obtención de todos los detalles necesarios para identificar de manera clara la empresa responsable del proyecto y facilitar la comunicación con los clientes.

Una vez obtenida toda la información necesaria, Chat GPT utilizará estos datos para generar la cotización correspondiente en formato de hoja Excel. La cotización contendrá la cantidad de materiales y mano de obra necesarios, junto con los precios unitarios y totales de cada elemento. Además, se calcularán y mostrarán los impuestos aplicables, como el IVA y el ISR, con el fin de proporcionar una visión completa del costo total del proyecto.

Para implementar este prompt, se utilizará un lenguaje de programación para interactuar con el usuario y recopilar la información necesaria para crear la

cotización. El programa estará diseñado para hacer preguntas específicas y guiar al usuario a través del proceso de proporcionar los detalles requeridos para la cotización del proyecto.

Una vez recopilada toda la información del usuario, el programa utilizará Microsoft Excel para almacenar y organizar los datos. Se utilizará el lenguaje de programación para manipular la hoja de cálculo y escribir los datos proporcionados por el usuario en las celdas correspondientes.

En la hoja de Excel, se mostrarán los datos de manera clara y organizada, lo que permitirá una fácil visualización y análisis de la información. Se pueden aplicar fórmulas y funciones en Excel para calcular automáticamente el presupuesto de construcción utilizando los datos proporcionados por el usuario.

¿Qué es Excel y uso de base de datos? (Recuperado de https://www.techtarget.com/searchenterprisedesktop/definition/Excel#:~:text=Excel%20is%20a%20spreadsheet%20program,calculate%20data%20in%20a%20 spreadsheet. El 12 de Noviembre del año 2022)

Microsoft Excel es una de las herramientas más utilizadas en entornos empresariales y académicos para realizar una amplia gama de tareas relacionadas con la gestión y análisis de datos. Además de su funcionalidad básica de hoja de cálculo, Excel ofrece una serie de características avanzadas que permiten a los usuarios realizar análisis complejos, crear gráficos dinámicos, automatizar tareas repetitivas mediante macros, y colaborar en tiempo real con otros usuarios a través de la integración con servicios en la nube como OneDrive o SharePoint.

La capacidad de dar formato a los datos de manera flexible, organizar la información de forma intuitiva y realizar cálculos precisos son algunas de las características que hacen de Excel una herramienta indispensable para profesionales de diversos campos, incluyendo finanzas, contabilidad, ingeniería, marketing, y muchos más. Además, la amplia disponibilidad de plantillas predefinidas y funciones incorporadas simplifica el proceso de creación y análisis de datos, permitiendo a los usuarios centrarse en la interpretación de los resultados y la toma de decisiones informadas.

Para elaborar una cotización de construcción en Excel, primero creó una hoja de cálculo donde registró los materiales necesarios junto con sus precios unitarios y otros costos, como la mano de obra y gastos adicionales. Utilizó fórmulas para calcular los costos totales de los materiales y también incluye estimaciones para la mano de obra y otros gastos relacionados. Una vez que tengo todos los datos, género cotizaciones detalladas para mis clientes utilizando una plantilla personalizada en Excel. Mantengo mi base de datos actualizada con cualquier cambio en los costos o requisitos a medida que avanza el proyecto para proporcionar información precisa en todo momento.

¿Cuáles son las fórmulas, tipos y uso dentro del proyecto?

(Recuperado de https://excelyvba.com/funciones-de-excel/ en el año 2023)

Las funciones en Excel son una característica clave que potencia este programa para el análisis y manejo de datos. Básicamente, son fórmulas predefinidas que efectúan operaciones utilizando valores específicos en un orden particular. La utilidad de las funciones reside en su capacidad para automatizar tareas complejas y efectuar cálculos con un simple comando.

Cuando se trabaja en Excel, es común realizar una variedad de cálculos, como sumas, promedios, conteos, búsquedas de datos específicos y operaciones matemáticas avanzadas. En vez de realizar manualmente cada cálculo, las funciones predefinidas están diseñadas para llevar a cabo estas tareas de manera eficaz.

Cada función en Excel tiene una estructura única, que consiste en reglas y convenciones que deben seguirse para su correcto uso. Esto incluye el orden en que se deben ingresar los argumentos, que son los valores que la función utilizará en sus cálculos. Es crucial entender la estructura de una función para emplearla de manera efectiva y obtener resultados precisos.

Función Si.Error (Recuperado de https://helpcenter.onlyoffice.com/es/onlyoffice-editors/onlyoffice-spreadsheet-editor/functions/iferror.aspx en el año de 2023)

La función SI.ERROR es una función lógica. Se usa para aprobar si hay algún error en la fórmula en el primer argumento. La función devolverá el resultado de la fórmula si no hay ningún error, o el valor si hay error si hay alguno. La sintaxis de la función SI.ERROR es:

SI.ERROR(valor, valor_si_error,)

donde valor y valor_si_error son valores introducidos a mano o incluidos en la celda a la que usted hace referencia.

En mi proyecto, he decidido integrar una función específica que desempeñará un papel fundamental: la devolución del valor subtotal. Esta función la implementará con el objetivo de mitigar cualquier posibilidad de error que

pueda surgir durante el proceso de cálculo. Al utilizar esta función, no solo estaré asegurando la precisión de los resultados, sino que también estaré mejorando la eficiencia del programa al evitar discrepancias en los cálculos de los precios parciales.

Además, he planificado incluir una funcionalidad adicional en mi aplicación: la suma automática de los precios subtotales y los impuestos correspondientes. Esto no solo simplificará el proceso de generación de informes y cotizaciones, sino que también reducirá significativamente la probabilidad de errores de cálculo. Al consolidar los precios parciales y los impuestos de manera automática, mi programa garantizará una mayor precisión y coherencia en los resultados finales.

Función BuscarV (Recuperado de https://www.ninjaexcel.com/formulas-y-funciones-de-excel/funcion-buscarv/ el 30 de noviembre del año 2023)

Finalidad: BUSCARV nos permite buscar y encontrar datos en una columna específica dentro de una tabla, a partir de coincidencias exactas o aproximadas. Es especialmente útil para cruzar bases de datos que contienen gran cantidad de información.

Características: La letra ‘V’ de su nombre hace referencia a que las búsquedas obedecen a un orden vertical. La dirección de búsqueda que posee esta función es desde nuestra columna de valor buscado hacia la derecha.

Sintaxis: =BUSCARV(valor_buscado; matriz tabla; indicador_columnas; [rango])

En mi proyecto, he decidido implementar la función de Buscar para mejorar la eficiencia y la precisión al momento de trabajar con bases de datos en Excel. Esta función desempeñará un papel crucial al automatizar la búsqueda y recuperación de información relevante a partir de una tabla de datos. En particular, al momento de identificar un código específico en una celda, la función de BuscarV permitirá que la descripción del producto asociado aparezca automáticamente en las celdas siguientes, junto con su precio correspondiente.

Lo interesante de la función de Buscar es que permite realizar esta búsqueda directamente en las tablas de las bases de datos, tanto en la de clientes como en la de materiales o productos de construcción. Esto significa que no será necesario realizar búsquedas manuales laboriosas ni introducir datos repetidamente, lo que agilizará significativamente el proceso.

Función de Concatenar (Recuperado de https://excelyvba.com/funcion-concatenar-en-excel/ en el año de 2023

En Excel, existen dos métodos principales para unir celdas, pero personalmente prefiero uno que no requiere el uso de fórmulas adicionales. Este enfoque no solo simplifica el proceso, sino que también puede mejorar la legibilidad de la hoja de cálculo, lo que facilita el mantenimiento y la comprensión de los datos.

Uno de los métodos más comunes es el uso de la función CONCATENAR en Excel. Esta función es especialmente útil cuando se trabaja con texto o cuando se necesita combinar texto y números en una misma celda. La sintaxis de la función CONCATENAR es bastante sencilla:

=CONCATENAR(arg1; arg2; ...; arg_n)

Aquí, arg1, arg2, etc., representan las diferentes celdas o textos que se desean combinar en una sola celda. Al utilizar esta función, se pueden unir fácilmente múltiples valores en una única celda, lo que puede resultar útil para crear etiquetas, títulos o cualquier otro contenido que requiera la combinación de varios elementos.

En mi proyecto, he decidido incorporar la función CONCATENAR de Excel como parte de mi estrategia para automatizar y organizar el proceso de manejo de datos. Esta función desempeñará un papel crucial al unir el código de identificación de productos desde mi base de datos con sus respectivas descripciones y precios de manera automatizada y ordenada.

La función CONCATENAR me permite combinar de forma eficiente múltiples valores en una sola celda, lo que es especialmente útil en situaciones donde necesito generar información compuesta a partir de datos dispersos en mi hoja de cálculo. En mi caso, aprovecho esta funcionalidad para crear etiquetas de productos completas que incluyan tanto su código de identificación como su descripción y precio asociados.

¿Qué son macros dentro de Excel y cómo se usará en el proyecto?

(Recuperado de https://www.computerworld.com/article/1614590/how-to-use-excel-macros-save-time-automate-work.html el 20 de Diciembre del año 2023

Las macros en Excel representan una herramienta altamente poderosa para automatizar tareas repetitivas y agilizar el flujo de trabajo en hojas de cálculo.

Básicamente, una macro es una serie de comandos y acciones grabados que se pueden reproducir cuantas veces sea necesario. Estos comandos pueden abarcar desde simples operaciones de formato de celdas hasta cálculos complejos y la manipulación de grandes conjuntos de datos.

La versatilidad de las macros permite su aplicación en una amplia gama de escenarios. Por ejemplo, pueden utilizarse para estandarizar el formato de una hoja de cálculo, realizar cálculos automáticos en un conjunto de datos extenso, importar información de otras fuentes de datos de manera automática o incluso generar gráficos dinámicos en función de datos cambiantes.

Para crear una macro, Excel proporciona una herramienta llamada "grabador de macros". Antes de usarla, es necesario habilitar la pestaña "Desarrollador" en la interfaz de Excel, donde se encuentran todas las herramientas relacionadas con macros y programación. Una vez habilitada esta pestaña, se puede acceder al grabador de macros y comenzar a grabar la secuencia de acciones deseada.

En el marco de mi proyecto, planeo implementar Macros en Excel con el fin de desarrollar un botón de búsqueda que habilite la visualización de formularios en una página emergente. Esta página, creada dentro de Excel, albergará las bases de datos necesarias para facilitar el acceso y la búsqueda eficiente de cualquier producto dentro del sistema.

La incorporación de esta funcionalidad busca mejorar significativamente la experiencia del usuario al interactuar con la base de datos. En lugar de navegar a través de múltiples hojas de cálculo o bases de datos dispersas, los usuarios

podrán acceder a una interfaz simplificada y centralizada para realizar sus consultas.

El botón de búsqueda activará una macro diseñada para abrir una ventana emergente que contenga los formularios de búsqueda. Esta ventana proporcionará opciones para ingresar criterios de búsqueda, como nombre del producto, categoría, precio, etc. Una vez ingresados los criterios, la macro realizará una búsqueda en las bases de datos correspondientes y mostrará los resultados de manera clara y organizada en la misma ventana emergente.

Además de agilizar el proceso de búsqueda, esta funcionalidad también tiene como objetivo optimizar la eficiencia del sistema al reducir el tiempo dedicado a la navegación y la búsqueda manual de información.

¿Qué es Visual Basic for Application (VBA)? Recuperado de https://www.hostgator.mx/blog/que-es-y-como-usar-vba/ el 16 de Agosto del año 2023)

VBA, conocido como Visual Basic for Applications, destaca como una herramienta sumamente adaptable y potente dentro del contexto de Excel. Su capacidad para automatizar acciones en las hojas de cálculo y personalizar el análisis de datos de acuerdo a las necesidades particulares de los usuarios lo convierte en un recurso invaluable para optimizar el trabajo en esta plataforma.

Cuando se emplea VBA, se crean comandos personalizados que simplifican la resolución de problemas y mejoran los procesos de trabajo. Esta versatilidad es especialmente útil para aquellos usuarios que trabajan de manera extensa en una misma hoja de cálculo y buscan simplificar y automatizar tareas

recurrentes. La programación en VBA puede acelerar significativamente la ejecución de acciones repetitivas, reduciendo de manera notable el tiempo necesario para completarlas. Además, al minimizar la posibilidad de errores humanos, contribuye a una mayor eficiencia en el flujo de trabajo y a la mejora de la precisión en los resultados obtenidos.

Las posibilidades que ofrece VBA son prácticamente ilimitadas, ya que puede adaptarse a una amplia variedad de situaciones y necesidades. Desde la creación de complejas herramientas de análisis de datos hasta la automatización de procesos administrativos rutinarios, VBA ofrece una solución personalizada para cada caso.

En mi proyecto, aproveché las capacidades de VBA para desarrollar una aplicación con una interfaz gráfica intuitiva y funcional. Esta aplicación está diseñada para mejorar la experiencia del usuario al interactuar con una base de datos en Excel, facilitando la búsqueda de productos y la gestión de cotizaciones. La aplicación incluye elementos clave como textbox, listbox y botones que permiten al usuario ingresar criterios de búsqueda, seleccionar opciones y realizar acciones específicas con solo unos pocos clics. La integración de VBA garantiza que estos elementos funcionen de manera eficiente y precisa.

Por ejemplo, al ingresar el nombre o código de un producto en el texto y hacer clic en el botón de búsqueda, se activa una secuencia de comandos programada en VBA que busca el producto correspondiente en la base de datos de Excel. Una vez encontrado, la información se muestra en una ventana

emergente o en un área designada de la interfaz gráfica, proporcionando al usuario una vista clara y ordenada de los resultados.

Además de la función de búsqueda, también he implementado un botón diseñado en VBA para eliminar toda la cotización en caso de ser necesario. Esto brinda una forma rápida y conveniente de reiniciar el proceso en caso de errores o cambios de última hora en la selección de productos. Para mejorar aún más la experiencia del usuario, he incorporado hipervínculos que permiten navegar fácilmente entre diferentes hojas de Excel dentro de la base de datos. Esto facilita el acceso a información adicional y proporciona una visión más completa y organizada de los datos disponibles. En resumen, la integración de VBA en mi proyecto agrega un nivel adicional de funcionalidad y eficiencia, mejorando significativamente la experiencia del usuario y la gestión de datos en Excel.

¿Cómo Chat GPT nos ayudará con la creación de un software y la automatización de este? (Recuperado de https://www.mytaskpanel.com/inteligencia-artificial-en-el-desarrollo-de-software/ el 18 de Octubre del año 2023)

La inteligencia artificial (IA) está marcando un hito en el ámbito del desarrollo de software al ofrecer una gama diversa de ventajas significativas. Entre ellas se encuentra la capacidad de automatizar tareas repetitivas y aburridas que tradicionalmente consumen mucho tiempo, lo que libera a los desarrolladores para enfocarse en actividades más creativas y estratégicas. Esta capacidad de la IA para aligerar la carga de trabajo rutinario es fundamental para optimizar el

proceso de desarrollo de software y aumentar la productividad en general. Un aspecto impresionante de la IA es su capacidad para generar código de forma automática a partir de especificaciones o requisitos definidos. Esta capacidad no sólo acelera el desarrollo de software, sino que también reduce significativamente la probabilidad de errores humanos. Además, la IA puede mejorar la calidad del software al identificar patrones y anomalías en el código, lo que permite a los desarrolladores corregir problemas de manera más rápida y prever posibles desafíos en el futuro.

Las técnicas de IA, como el aprendizaje automático y el procesamiento del lenguaje natural (PLN), están transformando fundamentalmente la forma en que se desarrolla el software. El aprendizaje automático permite a los sistemas mejorar continuamente su precisión y adaptarse a escenarios nuevos y desconocidos, lo que amplía sus capacidades más allá de su programación inicial. Por otro lado, el PLN, potenciado por el aprendizaje automático, está allanando el camino para la creación de aplicaciones más inteligentes y conversacionales en una amplia variedad de áreas. Además, la IA está impulsando el desarrollo de aplicaciones más inteligentes e interactivas, lo que mejora drásticamente la experiencia del usuario y crea nuevas oportunidades comerciales. Sin embargo, para aprovechar al máximo el potencial de la IA en el desarrollo de software, es crucial abordar una serie de desafíos éticos y sociales, como garantizar la transparencia en el uso de datos, asegurar la aplicabilidad adecuada de los modelos de IA y proteger la privacidad del usuario.

El aprendizaje automático para la creación de cotizaciones

Las técnicas de aprendizaje automático han alcanzado un hito significativo en la industria de la construcción al abrir la puerta a una predicción más precisa de los costos futuros de materiales y mano de obra. Esta capacidad para anticipar con mayor exactitud los costos tiene un impacto transformador en la forma en que se elaboran las cotizaciones y se gestionan los proyectos dentro del sector de la construcción.

Cuando nos sumergimos en el uso de modelos de aprendizaje automático, nos encontramos con un proceso complejo pero sumamente efectivo. Implica el análisis exhaustivo de grandes volúmenes de datos históricos y actuales relacionados con los costos de materiales, la mano de obra y otros factores pertinentes para la construcción. A través de este análisis minucioso, los modelos son capaces de identificar patrones y tendencias en los datos, lo que les permite generar predicciones sólidas sobre la evolución de los costos en el futuro.

La mejora en la precisión de la estimación de costos representa una ventaja significativa para las empresas constructoras. Les permite optimizar sus presupuestos y recursos de manera más eficiente, reducir los riesgos asociados a sobrecostos y, en última instancia, mejorar la rentabilidad de sus proyectos. Esta capacidad predictiva también otorga a estas empresas una ventaja estratégica invaluable al proporcionarles información fundamental para la toma de decisiones en la planificación y ejecución de proyectos.

El análisis detallado de datos históricos y actuales permite a los modelos de aprendizaje automático identificar una amplia gama de factores que influyen en

los costos de construcción. Desde la disponibilidad de materiales hasta la demanda de mano de obra y las fluctuaciones del mercado, estos modelos pueden discernir los elementos clave que impactan en los costos. Esta comprensión más profunda permite a las empresas adaptar sus estrategias de manera más precisa y tomar medidas proactivas para mitigar riesgos y maximizar la eficiencia en la gestión de proyectos de construcción.

Creación de Prototipo

Cuando hemos terminado de comprender lo que conlleva realizar este proyecto procederemos a aplicar los conocimientos previos para crear nuestro prototipo. Este prototipo será un excel básico donde tendremos distintos apartados enlazados entre hojas de cálculo con hipervínculos:

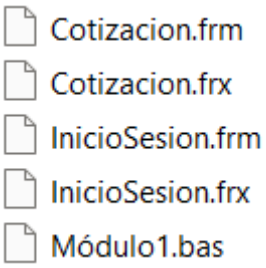
En nuestro primer apartado tendremos la lista de productos con un código especial el que nos servirá para poder contener los productos dentro de tablas (no declaradas) indicando si perteneces a la misma tabla a través del primer carácter que se encuentre en el código de serie del producto para poderlos agrupar en tablas, de nuevo no declaradas, junto al código una ligera descripción del producto y su precio unitario:

CODIGO	ARTICULO	PRECIO
TAIC0382	ABRAZADERA DE 9 PZS AJUSTABLE PIT BULL TOOLS	11.00
OC125	ACEITERA 125CC MAR-RAM FLEXIBLE	16.00
OC500	ACEITERA 500CC MAR-RAM FLEXIBLE	26.00
AM200MR	ACEITERA MAR-RAM 200CC PROFESIONAL	26.00
AM300MR	ACEITERA MAR-RAM 300CC PROFESIONAL	29.00
854	ADAPTADOR 3 PZS MAR-RAM	38.00
IA03-F	ADAPTADOR 3 PZS MAR-RAM IMPACTO	43.00
DR3412MR	ADAPTADOR 3/4 A 1/2 MAR-RAM REDUCCION	60.00
DR2MR	ADAPTADOR MAR-RAM 2 PZS 3/8A1/4 1/2A3/8	47.00
JA4MR	AFINACION 4 PZS 1/2 MAR-RAM PROFESIONAL	98.00
JA4MR-F	AFINACION 4 PZS FLEXIBLE MAR-RAM 1/2 MAT	115.00
DCS05-F	ARCO P/CALAR 5 SEGUETAS RAM-TOOLS	29.00
ARCO12MR	ARCO P/SEGUETA MAR-RAM CUADRADO M/AMARIL	58.00
1825	ARCO P/SEGUETA MINI ECONOMICO *	15.00
RS	ASPERSOR GRIRATORIO MAR-RAM	7.00
AEE1534MR	AUTOCLER 15 PZS MAR-RAM ENT 3/4 ASTRIADO	800.00
A1434MR	AUTOCLER 15 PZS MAR-RAM ENTRADA DE 3/4	800.00
A1612MR	AUTOCLER 16 PZS MAR-RAM PROF MILIMETRICO	189.00
A1714MMR	AUTOCLER 17 PZS 1/4" MAR-RAM MM. ROJO	85.00
SK17	AUTOCLER 17 PZS 1/4" MAR-RAM STD AZUL	58.00
A2134MR	AUTOCLER 21 PZS MAR-RAM ENTRADA DE 3/4"	1,000.00
A2512MR	AUTOCLER 25 PZS MAR-RAM PROF HOGAR/ CHIC	229.00
A2512MMR	AUTOCLER 25 PZS MAR-RAM PROFE USO TALLER	331.00

El segundo apartado tendremos los datos de los clientes estos se organizan por medio de un identificador único por cliente (del número 100 al 199) y para poder ingresar clientes que no son frecuentes no existen código, dentro del programa se colocara 000, así mismo se incluyen otros apartados para identificará el nombre completo (todo junto) del cliente, la dirección del mismo, su ciudad, el teléfono celular correspondiente al cliente y su dirección de correo electrónico para identificarlo de manera más sencilla. Un ejemplo de esta tabla será:

Id Cliente	Nombre de Cliente	Dirección	Ciudad	Teléfono	Email
100	José Pablo Juarez Perez	Av. 90, Calle 3, Bloc 1	Guatemala	7125-3225	mperez@gmail.com
101	Christofer Miguel Andubar	Av. 50, Calle 4, Bloc 20	Guatemala	7125-3225	cmiguel@gmail.com
102	Mario Aguilar Fuentes	Av. 10, Calle 5, Bloc10	Guatemala	2569-7841	maguliarpez123@gmail.com
103	Karen Montenegro Alvarez	Av. 8, Calle 8, Zona1	Guatemala	9863-1241	kmonal7854@gmail.com
104	Dara Flores Marroquin	Av. 7, Calle 9, Zona 10	Guatemala	5621-7489	daraflormar@gmail.com
105	Diego Cruz Maldonado	Av. 10, Calle 7A, Zona 24	Guatemala	4896-2014	diegocruzmal548@gmail.com
106	Sharon Gomez	NAC	GTM	6969-4224	g.sharon@ingeneria.edu.gt

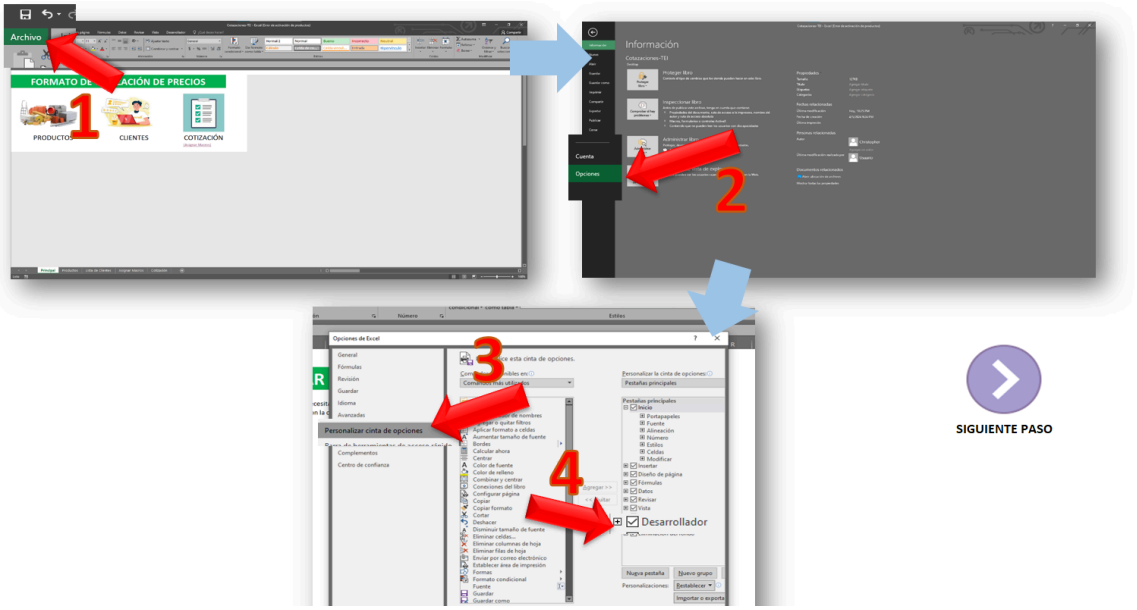
Dentro del tercer apartado tendremos esta la acción de cotizar, sin embargo este apartado no funciona sin macross y los mismo se deben de importar manualmente. Para poder ingresar los macros dentro del excel necesitaremos los archivos de los formularios de los macross:



Y para importarlos hay una pequeña guía dentro del mismo excel por debajo de la imagen para realizar la cotización, es un texto que dice “Asignar macros” que tiene un hipervínculos al tutorial express:

COMO ASIGNAR MACROS

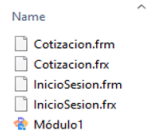
Primero debemos activar el modo desarrollador, para eso necesitamos ir a: archivos > opciones > personalizar cinta de opciones; y en la barra del lateral derecho habra un recuadro con la opcion de desarrollador



SIGUIENTE PASO

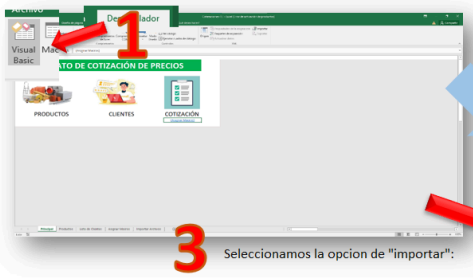
IMPORTAR ARCHIVOS NECESARIOS

El sistema de macros requiere de un formulario donde se mostrara graficamente la cotizacion conjunto al sistema.
Los archivos necesarios para que el sistema funcione correctamente son:

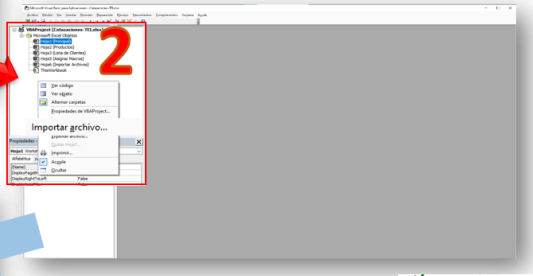


PASO ANTERIOR

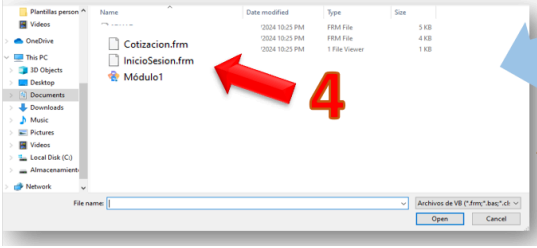
Una vez activado el modo desarrolladro ingresamos a "Visual Basic":



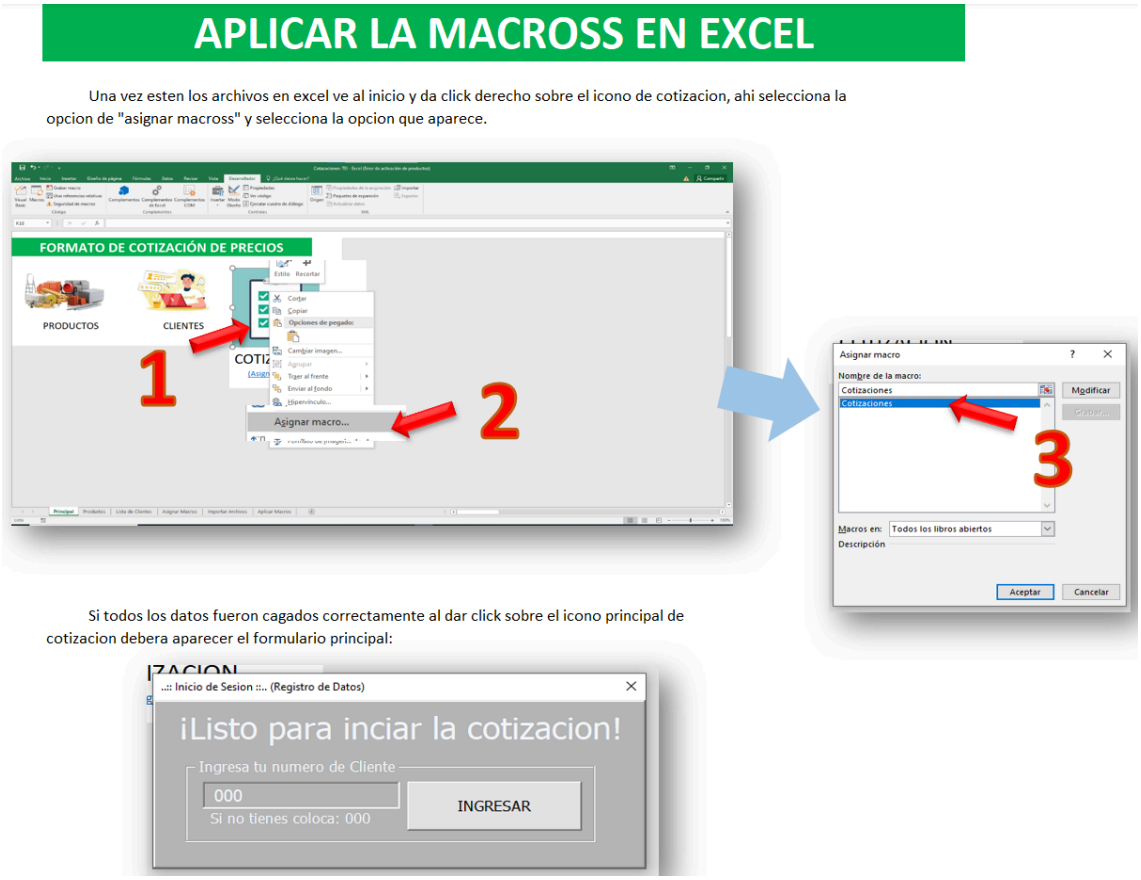
Una vez dentro de "visual basic" le daremos click derecho sobre los archivos que nos aparecen para desplegar un menu



3 Seleccionamos la opcion de "importar":



Aqui importaremos los archivos descargados (solo apareceran 3) para que queden mas o menos asi:



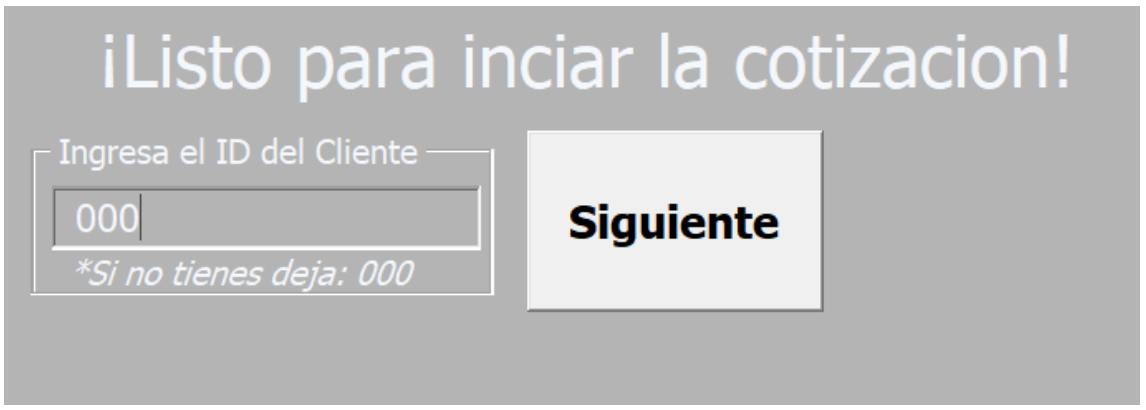
Sobre el macross para realizar la cotización hablaremos luego.

Y en el último apartado tendremos a los asesores de ventas. Este como es un prototipo tendrá los mismos datos que los clientes, con la diferencia de las posiciones de los datos es diferente y que en el id del asesor de ventas su rango es de 200 al 299, así es como tiene esta tabla nuestro prototipo:

Id Asesor	Nombre de Asesor	Dirección	Ciudad	Teléfono	Email
200	Christofer Miguel Andubar	Av. 90, Calle 3, Bloc 1	Guatemala	7125-3225	mperez@gmail.com
201	Mario Aguilar Fuentes	Av. 50, Calle 4, Bloc 20	Guatemala	7125-3225	cmiguel@gmail.com
202	José Pablo Juarez Perez	Av. 10, Calle 5, Bloc10	Guatemala	2569-7841	maguliarpez123@gmail.com
203	Dara Flores Marroquin	Av. 8, Calle 8, Zona1	Guatemala	9863-1241	kmonal7854@gmail.com
204	Diego Cruz Maldonado	Av. 7, Calle 9, Zona 10	Guatemala	5621-7489	daraflormar@gmail.com
205	Karen Montenegro Alvarez	Av. 10, Calle 7A, Zona 24	Guatemala	4896-2014	diegocruzmal548@gmail.com
206	Heidy Garcia	La roosvelt	Quiche	4224-3530	g.heidy@ingeneria.edu.gt

El funcionamiento del macross para poder realizar con acotizacion de los productos es:

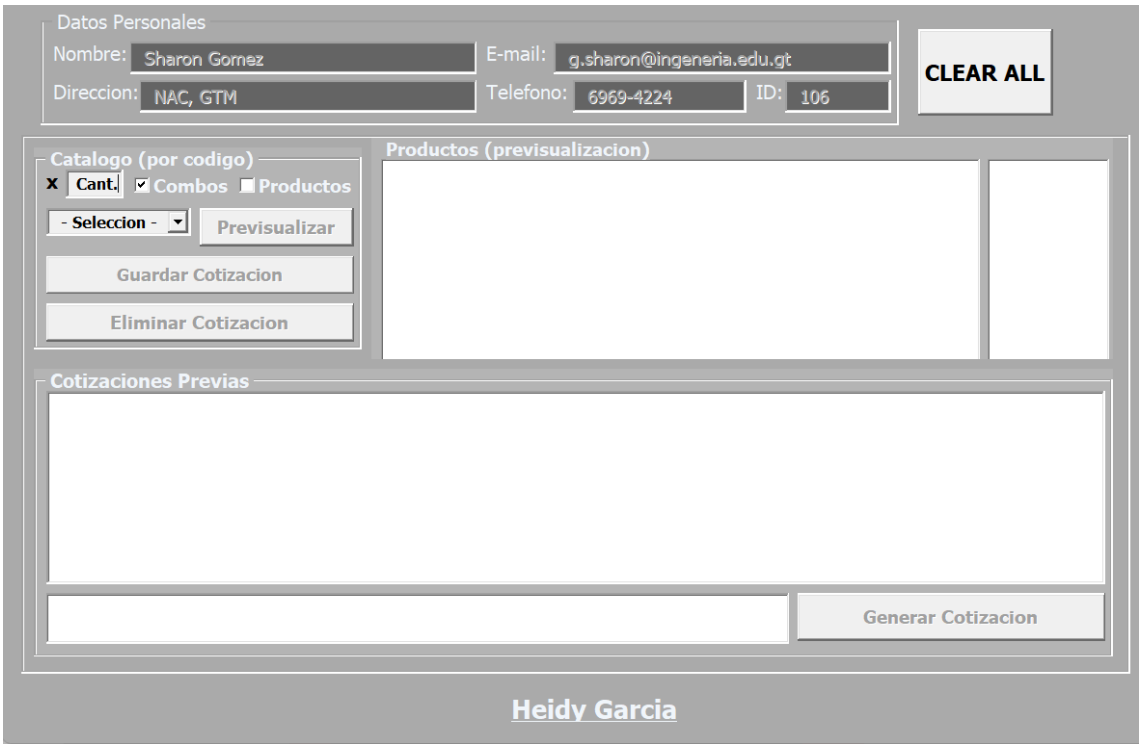
1. Al darle click a la ejecución del programa verás una interfaz sencilla donde tendrás que ingresar el id del cliente (en caso no tenga se deja con 000) y después debes ingresar el nombre del asesor de ventas a través de un selector. Ejemplo de la ejecución:



2. Luego de procesar los datos se abrirá un formulario que contendrá, inicialmente, un botón para poder cargar los archivos dentro del programa, este botón solo se usa para estética. Luego de que los datos se hayan cargado correctamente se mostrarán los datos del cliente en la parte superior y el nombre del asesor de ventas en la parte inferior para poder guiarse, así mismo el programa tendrá la opción de poder agregar productos o los conjuntos/combos para poder mantenerlas dentro de la

cotización. Algunos botones se habilitan únicamente al ejecutar otras acciones previamente para evitar errores dentro del sistema, siempre es necesario que contengan datos existentes dentro de la cotización.

Muestra gráfica del sistema:



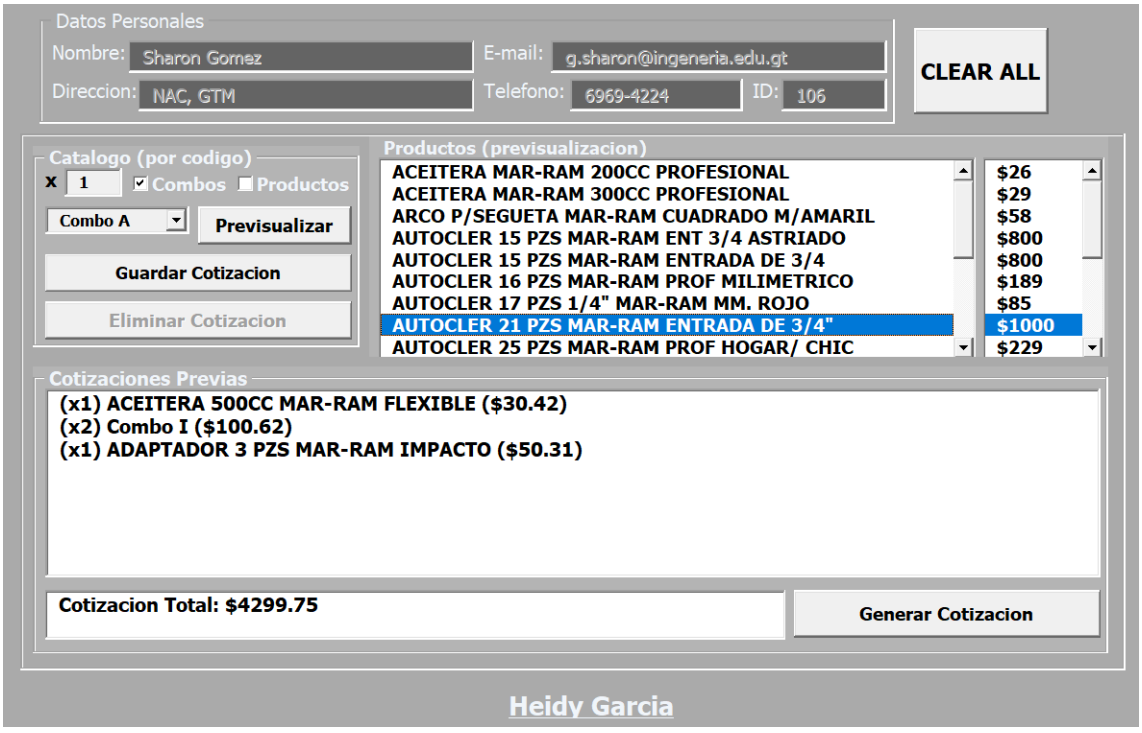
3. El funcionamiento de la interfaz gráfica es:
    a. Ubicándonos en la esquina superior derecha se encuentra un botón para limpiar todos los datos de todos los datos.
    b. En el apartado de “catálogo” están las opciones de cantidad, selector de compra, producto/combo de compra, botón de previsualizar para cargar los datos ingresados al apartado de “productos” y saber el contenido del combo junto a los precios del mismo, también se encuentra el botón de “guardar cotización” donde cargamos los datos de la previsualización al apartado de “cotización previa” para ir guardando y acumulando los datos de manera local así mismo esta el boton de eliminar cotización que

se habilita al seleccionar sobre una cotización guardad en “cotización previa” que, asu vez, re cargará los datos de la cotización previa a “productos” para rectificar los datos que contiene la cotización.

c. En el apartado de “Productos” solo sirve para previsualizar datos de los productos/combos donde al darle clic se seleccionará en ambas partes (precio y descripción/producto) para poder visualizar correctamente los productos.

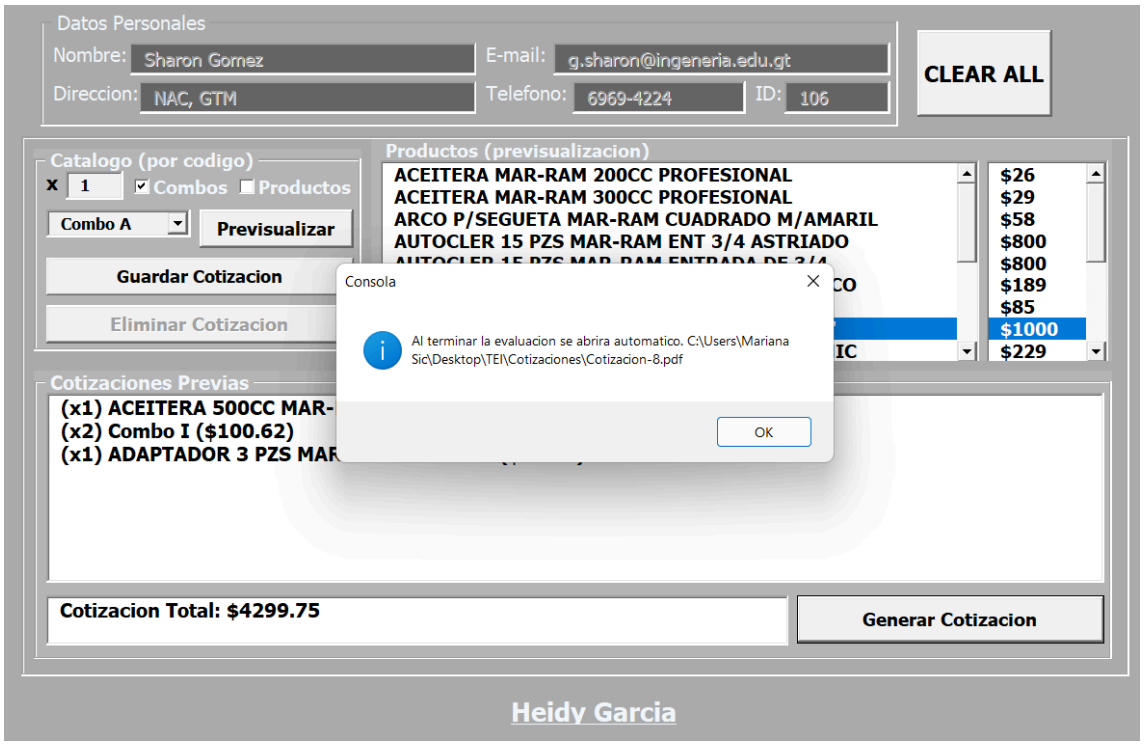
d. Por último está el apartado de “cotización previa” donde encontramos un espacio en blanco para ir guardando los datos de la cotización junto a un botón que generará un pdf donde podrá visualizar los datos completos de la cotización incluyendo los datos del cliente y el asesor de ventas.

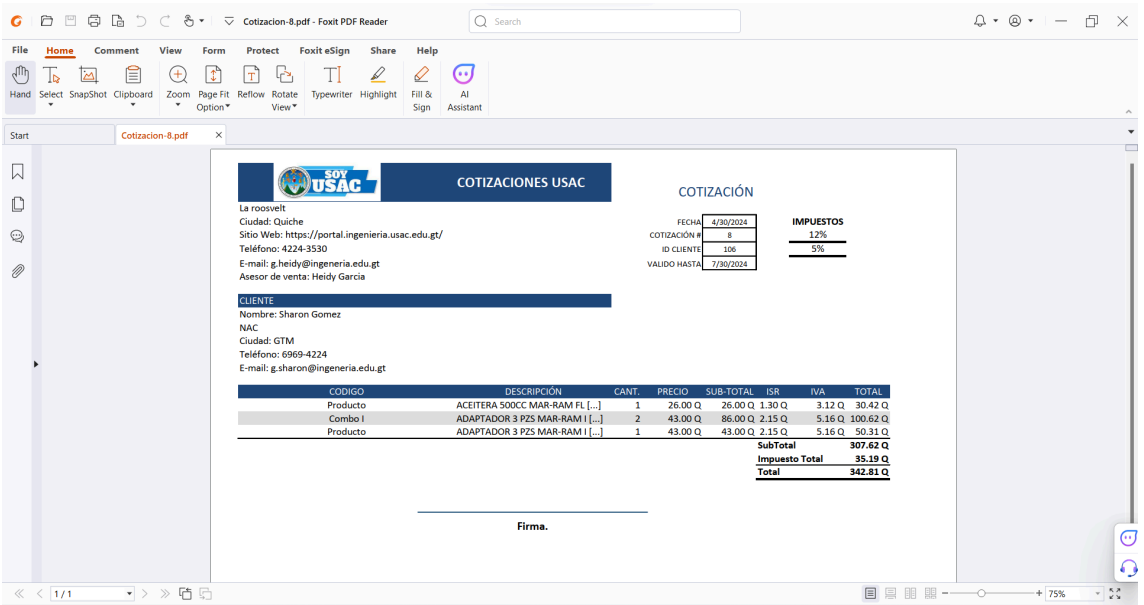
e. Ejemplo visual sobre los datos:



4. Cuando se da click sobre el boton de generar cotizacion el programa

lanzara un mensaje donde avisara al usuario que se generara un pdf y una vez termine de procesar los datos el pdf se abrira para poder visualizar la cotizacion en pdf, ejemplo para el pdf:





Hipótesis

La implementación de una IA Generativa mediante el uso de prompts para la creación de cotizaciones de productos de construcción representa una innovación significativa en el proceso de cotización en el sector de la construcción. Se postula que al emplear esta tecnología, se logrará una mejora sustancial en la eficiencia, precisión y competitividad de las cotizaciones, transformando fundamentalmente la forma en que se planifican y ejecutan los proyectos de construcción. Se espera que al proporcionar a la IA instrucciones específicas a través de prompts bien diseñados, esta sea capaz de generar cotizaciones detalladas y exhaustivas que incluyen información precisa sobre los materiales necesarios, los costos asociados, la mano de obra requerida y otros aspectos relevantes para la ejecución del proyecto. Esto permitiría a los profesionales de la construcción obtener cotizaciones más precisas en menos tiempo, reduciendo la carga de trabajo administrativo y liberando recursos para tareas más estratégicas.

Además, se anticipa que el uso de una IA Generativa para la creación de cotizaciones aumentará la consistencia y la calidad del proceso de cotización, al minimizar la posibilidad de errores humanos y garantizar la coherencia en los criterios de cálculo y estimación. Esta mayor precisión y consistencia no solo mejoraría la confiabilidad de las cotizaciones, sino que también fortalecería la credibilidad y la reputación de los profesionales de la construcción en el mercado.

Análisis de resultados de experimentos

1. Recopilación de datos: Tenemos paredes con diferentes materiales para cotizar:

*Medidas de materiales y construcción

*Tipos de paredes

*Precios

2. Análisis descriptivo:

Como se muestra nuestro programa almacena los datos en las tablas y este presenta código, descripción y unidades de medidas. Con los datos proporcionados en el menú de productos hacer una cotización.

3. Análisis estadístico:

Medimos y nuestro proyecto ahorra un 70% del tiempo que una persona manualmente ingresando datos, ya que este tiene que calcular producto por producto y ponerlo en una hoja de cotización, nuestro programa automatiza todo este proceso para hacerlo más rápido.

4. Interpretación de resultados:

Basándonos en los resultados del análisis vemos que el uso de nuestro proyecto es muy eficiente y puede resultar provechoso para cualquier empresa porque es un Excel muy fácil de entender además de ser muy intuitivo. Esto les puede ayudar a ser más rápido todo tipo de cotizaciones y automatizar más el proceso

5. Conclusiones:

Nuestro proyecto es uno que aporta mucho a las empresas además de ser fácil y se puede emplear a cualquier cotización, además de contar con menús y botón para dirigirnos a las diferentes partes para poder añadir, quitar o modificar los productos. Los macros son muy amigables para todo tipo de usuario y práctico para que un programador lo pueda modificar. Según los datos esto hace que el proyecto sea un éxito y sea de mucha utilidad.

Conclusiones

- La implementación de una IA Generativa para la creación de cotizaciones en el sector de la construcción representa una innovación significativa que promete mejorar la eficiencia, precisión y competitividad en el proceso de cotización. Esto podría transformar fundamentalmente la forma en que se planifican y ejecutan los proyectos de construcción, liberando recursos para tareas más estratégicas.

- El desarrollo de software que permita la creación automatizada de cotizaciones para obras de ingeniería civil, con funcionalidades completas para gestionar una amplia variedad de datos y generar cotizaciones detalladas, es fundamental para optimizar el proceso de cotización y garantizar la transparencia en la información.

- Los resultados de experimentos realizados con el software de cotización automatizada muestran un ahorro significativo de tiempo en comparación con el proceso manual de ingreso de datos. Esto sugiere que la automatización del proceso de cotización no solo mejora la eficiencia, sino que también puede ser una herramienta valiosa para las empresas al agilizar y simplificar el proceso de cotización

Egrafía

- https://enciclopediaeconomica.com/cotizacion/
- https://www.xataka.com/basics/chatgpt-que-como-usarlo-que-puedes-hacer-este-chat-inteligencia-artificial
- https://www.searchenginejournal.com/how-to-write-chatgpt-prompts/479324/
- https://www.techtarget.com/searchenterprisedesktop/definition/Excel#:~:text=Excel%20is%20a%20spreadsheet%20program,calculate%20data%20in%20a%20spreadsheet. https://excelyvba.com/funciones-de-excel/
- https://www.hostgator.mx/blog/que-es-y-como-usar-vba/
- https://www.mytaskpanel.com/inteligencia-artificial-en-el-desarrollo-de-software/

Codigo Comentado

Modulo1.bas

```
' Subrutina Cotizaciones: Esta subrutina se encarga de mostrar el formulario
InicioSesion.
Sub Cotizaciones()
    InicioSesion.Show ' Muestra el formulario InicioSesion.
End Sub

' Subrutina Custom: Esta subrutina se encarga de mostrar el formulario Customizar.
Sub Custom()
    Customizar.Show ' Muestra el formulario Customizar.
End Sub
```

InicioSesion.frm

```
' Declaración de variables para almacenar nombres de usuario ingresados y datos de
asesores de ventas.
Private USUARIO_INGRESADO() As String
Private ASESOR_VENTAS() As String

' Declaración de variables para tablas y rangos.
Dim ws As Worksheet
Dim tblClientes As ListObject
Dim tblAsesores As ListObject
Dim row As ListRow
Dim cell As Range

' Botón para iniciar sesión.
Dim texto As String
Dim ClienteID As Double

' Manejo del evento de hacer clic en el botón "BtnIngresar".
Private Sub BtnIngresar_Click()
    ' Variable para verificar si el asesor existe en la tabla.
    Dim AsesorExistente As Boolean
    AsesorExistente = False

    ' Buscar coincidencias en la tabla de asesores.
    Set ws = ThisWorkbook.Sheets("Asesores de Venta")
    Set tblAsesores = ws.ListObjects("Asesores")

    ' Verificar si tblAsesores es válido antes de usarlo.
    If Not tblAsesores Is Nothing Then
        Set col = tblAsesores.ListColumns(2)
```

```
        ' Iterar sobre los datos de la columna y verificar si el asesor seleccionado
está en la lista.
        For Each cell In col.DataBodyRange.Cells
            If cbNombres.Value = cell.Value Then
                AsesorExistente = True
            End If
        Next cell
    Else
        MsgBox "La tabla de Clientes no se encontró.", vbCritical, "Consola"
    End If

    ' Mostrar mensaje de error si el asesor no existe, de lo contrario, ocultar el
formulario de inicio de sesión y mostrar el formulario de cotización.
    If Not AsesorExistente Then
        MsgBox "El Asesor de Ventas no es valido.", vbCritical, "Consola"
    Else
        InicioSesion.Hide
        Cotizacion.Show
    End If
End Sub

' Manejo del evento de hacer clic en el botón "btnNext".
Private Sub btnNext_Click()
    ' Verificar que se haya ingresado un número en el campo "NumCliente".
    ' Convertir el texto a un valor numérico.
    If Not IsNumeric(NumCliente.Text) Then ' Verificar que es un número
        MsgBox "No ingresaste un Dato Numerico", vbCritical, "Consola"
    Else
        ClienteID = Val(NumCliente.Text) ' Convertir el texto a un valor numérico
    End If

    ' Verificar caracteres existentes.
    If Len(Trim(NumCliente.Text)) = 0 Then
        texto = "No ingresaste nada"
    ElseIf Len(texto) = 0 Then
        RecorerAsesores
        BuscarCliente
    End If

    ' Actualizar la interfaz gráfica con la información del cliente.
    frmCliente.Caption = "Cliente Registrado"
    lblCliente.Caption = "ID: " & NumCliente.Text
    NumCliente.Text = "Name: " & InicioSesion.MiVariable(2)
    frmAsesor.Visible = True
    frmCliente.Enabled = False
    BtnIngresar.Visible = True
    btnNext.Visible = False
End Sub
```

```
' Subrutina para buscar un cliente en la tabla.
Sub BuscarCliente()
    ' Buscar coincidencias en la tabla de clientes.
    Set ws = ThisWorkbook.Sheets("Lista de Clientes")
    Set tblClientes = ws.ListObjects("Clientes")

    ' Verificar si tblClientes es válido antes de usarlo.
    If Not tblClientes Is Nothing Then
        Set col = tblClientes.ListColumns(1)

        ' Iterar sobre los datos de la columna y buscar el cliente con el ID
correspondiente.
        For Each cell In col.DataBodyRange.Cells
            ReDim Preserve USUARIO_INGRESADO(1 To 6)
            If ClienteID = cell.Value Then
                ' Actualizar el texto con la información del cliente.
                texto = ws.Cells(cell.row, col.Index + 1).Value
                For i = 1 To 6 ' Recorre las 6 columnas
                    USUARIO_INGRESADO(i) = ws.Cells(cell.row, col.Index + i -
1).Value
                Next i
                Exit For
            Else
                ' Si no se encuentra el cliente, establecer valores predeterminados.
                For i = 1 To 6
                    USUARIO_INGRESADO(1) = "000"
                    USUARIO_INGRESADO(2) = "Usuario"
                    USUARIO_INGRESADO(i) = "-"
                Next i
            End If
        Next cell
    Else
        MsgBox "La tabla de Clientes no se encontró.", vbCritical, "Consola"
    End If

    ' Si no se encuentra ningún cliente, establecer el texto predeterminado como
"Usuario".
    If texto = "" Then
        texto = "Usuario"
    End If
End Sub

' Subrutina para recorrer los asesores y agregarlos a un combo box.
Sub RecorerAsesores()
    ' Buscar coincidencias en la tabla de asesores.
    Set ws = ThisWorkbook.Sheets("Asesores de Venta")
    Set tblAsesores = ws.ListObjects("Asesores")

    ' Verificar si tblAsesores es válido antes de usarlo.
```

```
    If Not tblAsesores Is Nothing Then
        Set col = tblAsesores.ListColumns(2)

        ' Iterar sobre los datos de la columna y agregar los nombres de los asesores
al combo box.
        For Each cell In col.DataBodyRange.Cells
            cbNombres.AddItem cell.Value
        Next cell
    Else
        MsgBox "La tabla de Clientes no se encontró.", vbCritical, "Consola"
    End If
End Sub

' Manejo del evento de cambio en el combo box "cbNombres".
Private Sub cbNombres_Change()
    ' Buscar coincidencias en la tabla de asesores.
    Set ws = ThisWorkbook.Sheets("Asesores de Venta")
    Set tblAsesores = ws.ListObjects("Asesores")

    ' Verificar si tblAsesores es válido antes de usarlo.
    If Not tblAsesores Is Nothing Then
        Set col = tblAsesores.ListColumns(2)

        ' Iterar sobre los datos de la columna y actualizar el campo de texto
"txtAsesor" con la información del asesor seleccionado.
        For Each cell In col.DataBodyRange.Cells
            If cbNombres.Value = cell.Value Then
                txtAsesor.Text = Left(cell.Value, 1) & ws.Cells(cell.row, col.Index -
1).Value
                ReDim Preserve ASESOR_VENTAS(1 To 6)
                For i = 1 To 6 ' Recorre las 6 columnas
                    ASESOR_VENTAS(i) = ws.Cells(cell.row, col.Index + i - 1).Value
                Next i
                Exit For
            End If
        Next cell
    Else
        MsgBox "La tabla de Clientes no se encontró.", vbCritical, "Consola"
    End If
End Sub

' ------- Otros Eventos -------

' Manejo del evento de hacer clic en el campo "NumCliente".
Private Sub NumCliente_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    ' Limpia el campo "NumCliente" al hacer clic.
    NumCliente.Text = ""
End Sub
```

```
' Manejo del evento de hacer clic en el campo "txtAsesor".
Private Sub txtAsesor_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    ' Limpia el campo "txtAsesor" al hacer clic.
    txtAsesor.Text = ""
End Sub

' Propiedades para acceder y modificar los datos de usuario ingresados y de asesores
de ventas.
Public Property Get MiVariable(Index As Integer) As String
    ' Obtiene los datos de usuario ingresados.
    MiVariable = USUARIO_INGRESADO(Index)
End Property

Public Property Let MiVariable(Index As Integer, ByVal Value As String)
    ' Establece los datos de usuario ingresados.
    USUARIO_INGRESADO(Index) = Value
End Property

Public Property Get MisDatos(Index As Integer) As String
    ' Obtiene los datos de asesores de ventas.
    MisDatos = ASESOR_VENTAS(Index)
End Property

Public Property Let MisDatos(Index As Integer, ByVal Value As String)
    ' Establece los datos de asesores de ventas.
    ASESOR_VENTAS(Index) = Value
End Property
```

Customizar.frm

```
' Declaración de variables para hojas de cálculo y tablas.
Dim ws As Worksheet
Dim wsa As Worksheet
Dim tbl As ListObject
Dim col As ListColumn
Dim colum As ListColumn
Dim cell As Range
Dim celda As Range

' Colección para almacenar productos seleccionados.
Dim productos As New Collection

' Manejo del evento de hacer clic en el botón "btnAgregar".
Private Sub btnAgregar_Click()
    ' Verificar si se ha seleccionado un producto.
    If Not cbSelect.Text = "- Seleccionar -" Then
        ' Verificar si el valor seleccionado no es nulo y agregarlo a la lista.
```

```
        If Not IsNull(cbSelect.Value) Then
            lbDatosCombo.AddItem cbSelect.Text
        End If
        ' Habilitar el botón "btnGuardar".
        btnGuardar.Enabled = True
        ' Cargar el producto seleccionado en el combo box.
        CargarSelect cbSelect.Text
    Else
        MsgBox "Debes seleccionar un producto.", vbExclamation, "Consola"
    End If
End Sub

' Manejo del evento de hacer clic en el botón "btnCargar".
Private Sub btnCargar_Click()
    ' Ordenar la lista de productos por código.
    OrdenarPorCodigo
    ' Definir la hoja de trabajo y la tabla de asesores.
    Set ws = ThisWorkbook.Sheets("Asesores de Venta")
    Set tbl = ws.ListObjects("Asesores")
    ' Variable para contar coincidencias.
    Dim Coincidencias As Double
    ' Variable para verificar si se debe verificar en Excel.
    Dim VerificarEnExcel As Integer
    Coincidencias = False
    ' Verificar si la tabla de asesores existe.
    If Not tbl Is Nothing Then
        Set col = tbl.ListColumns(1)

        ' Iterar sobre los datos de la primera columna y verificar si hay
coincidencias.
        For Each cell In col.DataBodyRange.Cells
            If txtCodigo.Text = cell.Value Then
                If ws.Cells(cell.row, col.Index + 4).Value = txtTel.Text Then
                    ' Mostrar formulario de customización y mensaje de bienvenida.
                    frmCustom.Visible = True
                    MsgBox "Bienvenido/a " & ws.Cells(cell.row, col.Index + 1).Value
& "."
                    Frame1.Caption = "Asesor: " & ws.Cells(cell.row, col.Index +
1).Value
                    Coincidencias = True
                End If
            End If
        Next cell
    Else
        MsgBox "La tabla de Clientes no se encontró.", vbCritical, "Consola"
    End If

    ' Si no hay coincidencias, preguntar si se debe verificar en Excel.
    If Not Coincidencias Then
```

```
        frmCustom.Visible = False
        VerificarEnExcel = MsgBox("No coincidieron los datos, ¿Quieres verificarlos
en Excel?", vbYesNo, "Consola")
        If VerificarEnExcel = vbYes Then
            Sheets("Asesores de Venta").Activate
            Unload Customizar
        End If
    Else
        CargarSelect ""
    End If
    ' Detectar letras iniciales.
    DetectarLetrasIniciales
End Sub

' Manejo del evento de hacer clic en el botón "btnEliminar".
Private Sub btnEliminar_Click()
    ' Borrar filas por código.
    BorrarFilasPorCodigo (Right(cbCombos.Value, 2))
    ' Ocultar formulario de customización y mostrar mensaje.
    Customizar.Hide
    MsgBox "Todo listo para continuar"
    Unload Customizar
End Sub

' Manejo del evento de hacer clic en el botón "btnGuardar".
Private Sub btnGuardar_Click()
    ' Definir la hoja de trabajo y la tabla de productos.
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    ' Definir la columna de códigos.
    Set col = tbl.ListColumns("CODIGO")
    ' Variable para verificar si el combo ya existe.
    Dim ComboExistente As Boolean
    ComboExistente = False

    ' Verificar si hay elementos en el combo box.
    If lbDatosCombo.ListCount = 0 Then
        MsgBox "Deben existir datos para el combo que vas a crear", vbExclamation,
"Consola"
    ElseIf Left(txtCodigoCombo.Text, 2) = "XY" And Not chbExistente.Value Then
        MsgBox "Coloca un código diferente al genérico.", vbExclamation, "Consola"
        ComboExistente = True
    Else
        Set letrasCombo = cbSelect
        letrasCombo.Clear

        ' Inicializar una colección para almacenar temporalmente las letras
iniciales.
        Dim letras As New Collection
```

```
        For Each cell In col.DataBodyRange.Cells
            ' Obtener la letra inicial de cada celda.
            Dim letra As String
            letra = Left(cell.Value, 2)

            ' Verificar si la letra inicial coincide con el código del combo.
            If letra = Left(txtCodigoCombo.Text, 2) Then
                ComboExistente = True
            End If
        Next cell
    End If

    ' Borrar filas por código si se ha seleccionado la opción correspondiente.
    If chbExistente.Value Then
        BorrarFilasPorCodigo Right(cbCombos.Value, 2)
    End If

    ' Mostrar mensaje si el combo ya existe, de lo contrario, agregar un nuevo combo.
    If ComboExistente Then
        MsgBox "El Combo ya fue creado con anterioridad, por favor verifica los
primeros caracteres en el código.", vbExclamation, "Consola"
    Else
        Dim nuevaFila As ListRow
        Set nuevaFila = tbl.ListRows.Add
        Set col = tbl.ListColumns("ARTICULO")

        For Each cell In col.DataBodyRange.Cells
            Dim rowPos As Long
            Dim colPos As Long
            rowPos = cell.row - tbl.HeaderRowRange.row + 1
            colPos = cell.Column - tbl.HeaderRowRange.Column + 1
            If colPos = 2 Then
                For i = 0 To lbDatosCombo.ListCount - 1
                    If cell.Value = lbDatosCombo.List(i) Then ' Buscar en el combo
seleccionado
                        nuevaFila.Range(i, 1).Value = txtCodigoCombo.Text
                        nuevaFila.Range(i, 2).Value = cell.Value
                        nuevaFila.Range(i, 3).Value = ws.Cells(cell.row, col.Index +
1).Value
                    End If
                Next i
            End If
        Next cell
        MsgBox "El nuevo combo `" & txtCodigoCombo.Text & "` ha sido creado con
éxito.", vbInformation, "Consola"
        ' Ocultar formulario de customización y mostrar mensaje.
        Customizar.Hide
        MsgBox "Todo listo para continuar"
        Unload Customizar
```

```
    End If
    ' Cargar productos en el combo box.
    CargarSelect ""
End Sub

' Manejo del evento de hacer clic en el botón "btnQuitar".
Private Sub btnQuitar_Click()
    ' Obtener el índice del producto seleccionado.
    Dim cotizacionSelect As Integer
    cotizacionSelect = lbDatosCombo.ListIndex

    ' Eliminar el producto seleccionado del combo box.
    lbDatosCombo.RemoveItem cotizacionSelect
    ' Cargar productos en el combo box.
    CargarSelect ""
    btnQuitar.Enabled = False
End Sub

' Manejo del evento de cambio en el combo box "cbCombos".
Private Sub cbCombos_Change()
    ' Definir la hoja de trabajo y la tabla de productos.
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    ' Definir la columna de códigos.
    Set col = tbl.ListColumns("CODIGO")
    ' Limpiar la lista de productos.
    lbDatosCombo.Clear

    ' Iterar sobre los datos y agregar productos que coincidan con el código del
combo.
    For Each cell In col.DataBodyRange.Cells
        If Left(cell.Value, 2) = Right(cbCombos.Value, 2) Then
            lbDatosCombo.AddItem ws.Cells(cell.row, col.Index + 1).Value
        End If
    Next cell
    ' Establecer el texto del cuadro de texto "txtCodigoCombo".
    txtCodigoCombo.Text = Right(cbCombos.Value, 2)
    ' Cargar productos en el combo box.
    CargarSelect ""
    ' Habilitar el botón "btnGuardar".
    btnGuardar.Enabled = True
End Sub

' Manejo del evento de clic en la casilla de verificación "chbExistente".
Private Sub chbExistente_Click()
    ' Desactivar la casilla de verificación "chbNuevo" si se ha seleccionado
"chbExistente".
    If chbNuevo.Value Then
        chbNuevo.Value = False
```

```
    Else
        chbExistente.Value = True
        cbCombos.Visible = True
        txtCodigoCombo.Visible = False
        btnEliminar.Enabled = True
    End If
End Sub

' Manejo del evento de clic en la casilla de verificación "chbNuevo".
Private Sub chbNuevo_Click()
    ' Desactivar la casilla de verificación "chbExistente" si se ha seleccionado
"chbNuevo".
    If chbExistente.Value Then
        chbExistente.Value = False
    Else
        chbNuevo.Value = True
        cbCombos.Visible = False
        txtCodigoCombo.Visible = True
        btnGuardar.Enabled = False
        btnEliminar.Enabled = False
    End If
End Sub

' Manejo del evento de clic en la lista de productos "lbDatosCombo".
Private Sub lbDatosCombo_Click()
    ' Habilitar el botón "btnQuitar".
    btnQuitar.Enabled = True
End Sub

' Manejo del evento de cambio en el cuadro de texto "txtTel".
Private Sub txtTel_Change()
    ' Validar el dato ingresado en el cuadro de texto "txtTel".
    ValidarDatosDeTexto txtTel.Text, txtTel
End Sub

' Manejo del evento de cambio en el cuadro de texto "txtCodigo".
Private Sub txtCodigo_Change()
    ' Validar el dato ingresado en el cuadro de texto "txtCodigo".
    ValidarDatosDeTexto txtCodigo.Text, txtCodigo
End Sub

' Manejo del evento de clic en el cuadro de texto "txtTel".
Private Sub txtTel_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X
As Single, ByVal Y As Single)
    ' Limpiar el cuadro de texto "txtTel" si contiene el valor por defecto.
    If txtTel.Text = "0000-0000" Then
        txtTel.Text = ""
    End If
End Sub
```

```
' Manejo del evento de clic en el cuadro de texto "txtCodigoCombo".
Private Sub txtCodigoCombo_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    ' Limpiar el cuadro de texto "txtCodigoCombo" si contiene el valor por defecto.
    If txtCodigoCombo.Text = "XYZ-123" Then
        txtCodigoCombo.Text = ""
    End If
End Sub

' Manejo del evento de cambio en el cuadro de texto "txtCodigoCombo".
Private Sub txtCodigoCombo_Change()
    ' Convertir las letras ingresadas en el cuadro de texto "txtCodigoCombo" a
mayúsculas.
    Dim texto As String
    Dim i As Integer
    texto = txtCodigoCombo.Text
    For i = 1 To Len(texto)
        If Mid(texto, i, 1) Like "[a-zA-Z]" Then
            Mid(texto, i, 1) = UCase(Mid(texto, i, 1))
        End If
    Next i
    txtCodigoCombo.Text = texto
End Sub

' Manejo del evento de clic en el cuadro de texto "txtCodigo".
Private Sub txtCodigo_MouseDown(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    ' Limpiar el cuadro de texto "txtCodigo" si contiene el valor por defecto.
    If txtCodigo.Text = "000" Then
        txtCodigo.Text = ""
    End If
End Sub

' Función para validar los datos ingresados en un cuadro de texto.
Sub ValidarDatosDeTexto(Dato As String, Cuadro As Object)
    ' Verificar el formato del dato ingresado y reemplazarlo si es incorrecto.
    If InStr(Dato, "-") = 5 Then
        Cuadro.Text = Cuadro.Text
    ElseIf Not IsNumeric(Dato) Then
        Cuadro.Text = ""
    End If
    If Val(Dato) <= 0 Then
        Cuadro.Text = ""
    End If
End Sub

' Subrutina para cargar productos en el combo box.
Sub CargarSelect(Coincidencia As String)
```

```
    ' Ordenar la lista de productos por código.
    OrdenarPorCodigo
    Dim letrasCombo As ComboBox
    Dim item As Variant
    ' Definir la hoja de trabajo y la tabla de productos.
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    Set col = tbl.ListColumns("ARTICULO")
    Set letrasCombo = cbSelect
    letrasCombo.Clear

    ' Limpiar la colección de productos.
    For Each item In productos
        productos.Remove 1
    Next item

    ' Agregar productos al combo box.
    If Not lbDatosCombo.ListCount = 0 Then
        For i = 0 To lbDatosCombo.ListCount - 1
            productos.Add lbDatosCombo.List(i)
        Next i
    End If

    ' Iterar sobre los datos y agregar productos al combo box.
    For Each cell In col.DataBodyRange.Cells
        ' Verificar si el producto ya está en la colección y agregarlo si no está.
        If Not Contiene(productos, cell.Value) Then
            cbSelect.AddItem cell.Value
        End If
    Next cell

    letrasCombo.Value = "- Seleccionar -"
End Sub

' Subrutina para detectar letras iniciales.
Sub DetectarLetrasIniciales()
    ' Definir la hoja de trabajo y la tabla de productos.
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    Set col = tbl.ListColumns("CODIGO")
    Set letrasCombo = cbCombos
    letrasCombo.Clear

    ' Inicializar una colección para almacenar temporalmente las letras iniciales.
    Dim letras As New Collection
    For Each cell In col.DataBodyRange.Cells
        ' Obtener la letra inicial de cada celda.
        Dim letra As String
        letra = Left(cell.Value, 2)
```

```
        ' Verificar si la letra inicial no está en la colección y agregarla si no
está.
        If Not Contiene(letras, letra) Then
            letras.Add letra
            letrasCombo.AddItem "Combo " & letra
        End If
    Next cell
End Sub

' Función para verificar si una colección contiene un valor específico.
Function Contiene(col As Collection, valor As Variant) As Boolean
    Dim elem As Variant
    For Each elem In col
        If elem = valor Then
            Contiene = True
            Exit Function
        End If
    Next elem
    Contiene = False
End Function

' Subrutina para ordenar la lista de productos por código.
Sub OrdenarPorCodigo()
    Dim ws As Worksheet
    Dim rng As Range
    Dim LastRow As Long

    ' Definir la hoja de trabajo.
    Set ws = ThisWorkbook.Sheets("Productos") ' Reemplaza "NombreDeTuHoja" con el
nombre real de tu hoja

    ' Encontrar la última fila con datos en la columna A.
    LastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    ' Definir el rango que quieres ordenar.
    Set rng = ws.Range("A1:C" & LastRow) ' Asumiendo que tus datos comienzan en la
fila 2 y que la columna C contiene los precios

    ' Ordenar el rango por los caracteres iniciales del código.
    With ws.Sort
        .SortFields.Clear
        .SortFields.Add Key:=rng.Columns(1), SortOn:=xlSortOnValues,
Order:=xlAscending, DataOption:=xlSortNormal
        .SetRange rng
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
```

```
        .Apply
    End With
End Sub

' Función para borrar filas por código.
Function BorrarFilasPorCodigo(ByVal codigoABorrar As String)
    Dim rng As Range
    Dim LastRow As Long
    Dim i As Long

    ' Definir la hoja de trabajo.
    Set ws = ThisWorkbook.Sheets("Productos") ' Reemplaza "NombreDeTuHoja" con el
nombre real de tu hoja

    ' Encontrar la última fila con datos en la columna A.
    LastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).row

    ' Definir el rango que quieres recorrer.
    Set rng = ws.Range("A1:A" & LastRow) ' Asumiendo que tus datos comienzan en la
fila 2 y la columna A contiene los códigos

    ' Recorrer el rango y borrar las filas que coincidan con el código.
    For i = rng.Rows.Count To 1 Step -1
        If Left(rng.Cells(i, 1).Value, 2) = codigoABorrar Then
            ws.Rows(i).EntireRow.Delete ' Sumamos 1 porque A2 corresponde a la fila 1
en la hoja de cálculo
        End If
    Next i
End Function
```

Cotizacion.frm

```
' Variables de Entorno
Dim ws As Worksheet ' Declaración de la variable para una hoja de cálculo
Dim wsa As Worksheet ' Declaración de la variable para otra hoja de cálculo
Dim tbl As ListObject ' Declaración de la variable para una tabla
Dim col As ListColumn ' Declaración de la variable para una columna de la tabla
Dim cell As Range ' Declaración de la variable para una celda
Dim celda As Range ' Declaración de la variable para otra celda
Dim precioIVA As Double ' Declaración de la variable para el precio con IVA
Dim precioISR As Double ' Declaración de la variable para el precio con ISR
Dim totalPrecio As Double ' Declaración de la variable para el precio total
Dim valorPrecio As Double ' Declaración de la variable para el valor del precio
Dim cotizacionTot As Double ' Declaración de la variable para el total de la
cotización
Dim letrasCombo As MSForms.ComboBox ' Declaración de la variable para un ComboBox

' Subrutina para limpiar los campos del formulario al hacer clic en un botón
Private Sub btnClear_Click()
```

```
    ' Esta subrutina se activa cuando se hace clic en el botón "Limpiar".

    ' Limpia el contenido de la lista de descripciones.
    lbDescipcion.Clear

    ' Limpia el contenido de la lista de precios.
    lbPrecios.Clear

    ' Limpia el contenido de la lista de precios totales.
    lbTotal.Clear

    ' Limpia el contenido del historial de cotizaciones.
    lbHistorial.Clear

    ' Desactiva el botón "Eliminar" del formulario.
    btnEliminar.Enabled = False

    ' Desactiva el botón "Generar" del formulario.
    btnGenerar.Enabled = False

    ' Establece el valor predeterminado del ComboBox a "- Seleccion -".
    cbSelect.Value = "- Seleccion -"

    ' Establece el texto predeterminado del cuadro de cantidad a "Cant.".
    txtCantidad.Text = "Cant."
End Sub

' Subrutina para eliminar una cotización del historial
Private Sub btnEliminar_Click()
    ' Esta subrutina se activa cuando se hace clic en el botón "Eliminar".

    ' Declaración de variables locales para almacenar la cotización seleccionada y su
precio.
    Dim cotizacionSelect As Integer
    Dim itemCotizacion As String
    Dim itemPrecio As String

    ' Almacena el índice de la cotización seleccionada en la lista de historial.
    cotizacionSelect = lbHistorial.ListIndex

    ' Almacena el valor de la cotización seleccionada.
    itemCotizacion = lbHistorial.Value

    ' Extrae el precio de la cotización seleccionada.
    itemPrecio = Mid(itemCotizacion, InStr(1, itemCotizacion, "($") + 2,
Len(itemCotizacion) - InStr(1, itemCotizacion, "($") - 2)

    ' Resta el precio de la cotización seleccionada del total de la cotización.
    cotizacionTot = cotizacionTot - Val(itemPrecio)
```

```
    ' Reinicia el valor del precio total.
    totalPrecio = 0

    ' Actualiza el total de la cotización.
    totCotizacion

    ' Elimina la cotización seleccionada del historial.
    lbHistorial.RemoveItem cotizacionSelect

    ' Desactiva el botón "Eliminar".
    btnEliminar.Enabled = False
End Sub

' Subrutina para generar una cotización en formato PDF y cerrar los formularios de inicio de sesión y cotización.
Private Sub btnGenerar_Click()
    ' Esta subrutina se activa cuando se hace clic en el botón "Generar".

    ' Llama a la subrutina para generar una cotización en formato PDF.
    CotizacionPDF

    ' Cierra el formulario de inicio de sesión.
    Unload InicioSesion

    ' Cierra el formulario de cotización.
    Unload Cotizacion
End Sub

' Subrutina para guardar los datos en el historial local.
Private Sub btnGuardar_Click()
    ' Esta subrutina se activa cuando se hace clic en el botón "Guardar".

    ' Verifica si se ha seleccionado un artículo.
    If cbSelect.Value = "- Seleccion -" Then
        MsgBox "Argumentos no válidos", vbCritical, "Consola"

    ' Verifica si la cantidad ingresada es mayor que cero.
    ElseIf Not Val(txtCantidad.Text) > 0 Then
        MsgBox "Argumentos no válidos", vbCritical, "Consola"

    ' Verifica si el precio total es igual a cero.
    ElseIf totalPrecio = 0 Then
        MsgBox "Argumentos no válidos", vbCritical, "Consola"

    ' Si todas las condiciones se cumplen, agrega la cotización al historial.
    Else
        Prevista
```

```
        ' Agrega la cotización al historial con el formato "(x Cantidad) Nombre ($
Precio)".
        lbHistorial.AddItem "(x" & txtCantidad.Text & ") " & cbSelect.Text & " ($" &
totalPrecio & ")"

        ' Activa el botón "Generar".
        btnGenerar.Enabled = True

        ' Actualiza el total de la cotización.
        totCotizacion
    End If
End Sub

' Subrutina para mostrar la vista previa de la cotización.
Private Sub btnPreView_Click()
    ' Esta subrutina se activa cuando se hace clic en el botón "PreView".

    ' Llama a la subrutina para mostrar la vista previa de la cotización.
    Prevista
End Sub

' Subrutina para controlar el evento Click del CheckBox "chbCombo".
Private Sub chbCombo_Click()
    ' Si el CheckBox "chbProductos" está marcado, lo desmarca.
    If chbProductos.Value Then
        chbProductos.Value = False
    ' Si no, marca el CheckBox "chbCombo".
    Else
        chbCombo.Value = True
    End If

    ' Limpia el ComboBox "cbSelect" y establece el valor predeterminado.
    cbSelect.Clear
    cbSelect.Value = "- Seleccion -"

    ' Llama a la subrutina para detectar las letras iniciales.
    DetectarLetrasIniciales
End Sub

' Subrutina para controlar el evento Click del CheckBox "chbProductos".
Private Sub chbProductos_Click()
    ' Si el CheckBox "chbCombo" está marcado, lo desmarca.
    If chbCombo.Value Then
        chbCombo.Value = False
    ' Si no, marca el CheckBox "chbProductos".
    Else
        chbProductos.Value = True
    End If
```

```
    ' Limpia el ComboBox "cbSelect" y establece el valor predeterminado.
    cbSelect.Clear
    cbSelect.Value = "- Seleccion -"

    ' Llama a la subrutina para detectar los códigos.
    DetectarCodigos
End Sub

' Subrutina para controlar el evento Click del CommandButton1 (botón inicial).
Private Sub CommandButton1_Click()
    ' Rellena los campos de texto con los datos almacenados en el formulario de
inicio de sesión.
    txtNombre.Text = InicioSesion.MiVariable(2)
    txtDireccion.Text = InicioSesion.MiVariable(3) & ", " &
InicioSesion.MiVariable(4)
    txtMail.Text = InicioSesion.MiVariable(6)
    txtTel.Text = InicioSesion.MiVariable(5)
    txtID.Text = InicioSesion.MiVariable(1)
    lblAsesor.Caption = InicioSesion.MisDatos(1)

    ' Llama a la subrutina para detectar las letras iniciales.
    DetectarLetrasIniciales

    ' Hace visible el Frame1 y el Frame2, oculta el CommandButton1 y muestra el botón
btnClear.
    Frame1.Visible = True
    Frame2.Visible = True
    CommandButton1.Visible = False
    btnClear.Visible = True
End Sub

' Subrutina para controlar el evento Change del ComboBox "cbSelect" (al seleccionar
un combo).
Private Sub cbSelect_Change()
    ' Verifica si el texto en el campo txtCantidad es igual a "Cant.".
    If txtCantidad.Text = "Cant." Then
        ' Si es así, establece el valor predeterminado a "1".
        txtCantidad.Text = "1"
    ' Verifica si el texto en el campo txtCantidad no es un número.
    ElseIf Not IsNumeric(txtCantidad.Text) Then
        ' Si es así, establece el valor predeterminado a "1".
        txtCantidad.Text = "1"
    End If

    ' Habilita los botones btnGuardar y btnPreView.
    btnGuardar.Enabled = True
    btnPreView.Enabled = True

    ' Llama a la subrutina para mostrar la vista previa.
```

```
    Prevista
End Sub

' Subrutina para controlar el evento Click del ListBox "lbDescipcion".
Private Sub lbDescipcion_Click()
    ' Declara una variable para almacenar el índice seleccionado.
    Dim indexDesc As Integer

    ' Obtiene el índice seleccionado en el ListBox "lbDescipcion".
    indexDesc = lbDescipcion.ListIndex

    ' Selecciona el elemento correspondiente en el ListBox "lbPrecios".
    lbPrecios.Selected(indexDesc) = True
End Sub

' Subrutina para controlar el evento Change del ListBox "lbHistorial" (al seleccionar
un dato del historial).
Private Sub lbHistorial_Change()
    ' Verifica si el valor seleccionado en el ListBox "lbHistorial" contiene la
palabra "Combo" (ignorando mayúsculas y minúsculas).
    If InStr(1, lbHistorial.Value, "Combo", vbTextCompare) > 0 Then
        ' Si contiene "Combo", marca el CheckBox "chbCombo" y desmarca el CheckBox
"chbProductos".
        chbCombo.Value = True
        chbProductos.Value = False
        ' Llama a la subrutina para desglosar la cadena.
        DesglosarCadena
    Else
        ' Si no contiene "Combo", desmarca el CheckBox "chbCombo" y marca el CheckBox
"chbProductos".
        chbCombo.Value = False
        chbProductos.Value = True
        ' Llama a la subrutina para desglosar la cadena.
        DesglosarCadena
    End If
    ' Habilita el botón btnEliminar.
    btnEliminar.Enabled = True
End Sub

' Subrutina para controlar el evento Click del ListBox "lbPrecios".
Private Sub lbPrecios_Click()
    ' Declara una variable para almacenar el índice seleccionado.
    Dim indexDesc As Integer
    ' Obtiene el índice seleccionado en el ListBox "lbPrecios".
    indexDesc = lbPrecios.ListIndex
    ' Selecciona el elemento correspondiente en el ListBox "lbDescipcion".
    lbDescipcion.Selected(indexDesc) = True
End Sub
```

```
' Subrutina para controlar el evento MouseDown del TextBox "txtCantidad" (al editar el cuadro de texto 'cantidad').
Private Sub txtCantidad_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
    ' Verifica si el texto en el campo txtCantidad es igual a "Cant.".
    If txtCantidad.Text = "Cant." Then
        ' Si es así, establece el valor predeterminado a "1".
        txtCantidad.Text = "1"
    End If
End Sub

' Subrutina para calcular y mostrar una vista previa de la cotización.
Sub Prevista() ' funcion prevista
    ' Limpia los ListBox "lbDescipcion" y "lbPrecios".
    lbDescipcion.Clear
    lbPrecios.Clear
    ' Reinicia el valor de las variables totalPrecio y valorPrecio a cero.
    totalPrecio = 0
    valorPrecio = 0

    ' Itera a través de cada fila en la tabla "tbl".
    For Each row In tbl.ListRows
        ' Itera a través de cada celda en la fila.
        For Each cell In row.Range.Cells
            ' Declara variables para almacenar la posición de la fila y columna.
            Dim rowPos As Long
            Dim colPos As Long
            ' Calcula la posición de la fila y columna en relación con la tabla.
            rowPos = cell.row - tbl.HeaderRowRange.row + 1
            colPos = cell.Column - tbl.HeaderRowRange.Column + 1
            ' Verifica si se ha seleccionado la opción "Combo".
            If chbCombo.Value Then
                ' Si la columna es la primera (código de producto) y el código coincide con la selección del ComboBox.
                If colPos = 1 Then
                    If Left(cell.Value, 2) = Right(cbSelect.Text, 2) Then ' en el combo seleccionado buscar
                        ' Agrega la descripción del producto al ListBox "lbDescipcion".
                        lbDescipcion.AddItem ws.Cells(cell.row, col.Index + 1).Value ' Descripcion
                        ' Calcula y agrega el precio del producto al ListBox "lbPrecios".
                        lbPrecios.AddItem "$" & Val(ws.Cells(cell.row, col.Index + 2).Value) * Val(txtCantidad.Text) ' Precio
                        ' Calcula el precio total acumulado.
                        valorPrecio = Val(ws.Cells(cell.row, col.Index + 2).Value) * Val(txtCantidad.Text) ' acumuladores (precio)
                        totalPrecio = totalPrecio + valorPrecio
```

```
                    End If
                End If
            Else
                ' Si no se ha seleccionado la opción "Combo".
                ' Si la columna es la segunda (nombre del producto) y el nombre
coincide con la selección del ComboBox.
                If colPos = 2 Then
                    If cell.Value = cbSelect.Value Then ' en el combo seleccionado
buscar
                        ' Agrega la descripción del producto al ListBox
"lbDescipcion".
                        lbDescipcion.AddItem cell.Value ' Descripcion
                        ' Calcula y agrega el precio del producto al ListBox
"lbPrecios".
                        lbPrecios.AddItem "$" & Val(ws.Cells(cell.row, col.Index +
1).Value) * Val(txtCantidad.Text) ' Precio
                        ' Calcula el precio total.
                        totalPrecio = Val(ws.Cells(cell.row, col.Index + 1).Value) *
Val(txtCantidad.Text)
                    End If
                End If
            End If
        Next cell
    Next row

    ' ------------------- Decoracion -------------------
    ' Agrega líneas de separación y conceptos de impuestos al ListBox "lbDescipcion".
    lbDescipcion.AddItem "----"
    lbDescipcion.AddItem "IVA   - (12%)"
    lbDescipcion.AddItem "ISR   - (5%)"
    lbDescipcion.AddItem "Total"

    ' Calcula los impuestos (IVA y ISR) y actualiza el precio total.
    precioIVA = totalPrecio * 0.12
    precioISR = totalPrecio * 0.05
    totalPrecio = totalPrecio + precioIVA + precioISR

    ' Agrega los impuestos y el precio total al ListBox "lbPrecios".
    lbPrecios.AddItem "----"
    lbPrecios.AddItem "$" & precioIVA
    lbPrecios.AddItem "$" & precioISR
    lbPrecios.AddItem "$" & totalPrecio
End Sub

' Función para detectar las letras iniciales de los códigos de serie y acomodar los
combos en consecuencia.
Sub DetectarLetrasIniciales() ' Funcion para acomodar los combos segun el codigo de
serie
    ' Establece la hoja de cálculo y la tabla de datos.
```

```
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    Set col = tbl.ListColumns("CODIGO")
    Set letrasCombo = cbSelect
    letrasCombo.Clear

    ' Inicializar una colección para almacenar temporalmente las letras iniciales
    Dim letras As New Collection
    For Each cell In col.DataBodyRange.Cells
        ' Obtener la letra inicial de cada celda
        Dim letra As String
        letra = Left(cell.Value, 2)

        ' Verificar si la letra inicial no está en la colección y agregarla si no
está
        If Not Contiene(letras, letra) Then
            letras.Add letra
            ' Agrega la letra inicial como un elemento del ComboBox
            letrasCombo.AddItem "Combo " & letra
        End If
    Next cell
End Sub

' Función para detectar los códigos de productos y acomodar los combos en
consecuencia.
Sub DetectarCodigos() ' Funcion para acomodar los productos
    ' Establece la hoja de cálculo y la tabla de datos.
    Set ws = ThisWorkbook.Sheets("Productos")
    Set tbl = ws.ListObjects("Catalogo")
    Set col = tbl.ListColumns("ARTICULO")
    Set letrasCombo = cbSelect
    letrasCombo.Clear

    ' Itera a través de cada celda en la columna "ARTICULO".
    For Each cell In col.DataBodyRange.Cells
        ' Agrega el valor de la celda como un elemento del ComboBox.
        letrasCombo.AddItem cell.Value
    Next cell
End Sub

' Subrutina para desglosar la cadena seleccionada del historial y actualizar la
previsualización.
Sub DesglosarCadena() ' re acomodar datos del historial a la previsualizacion
    Dim cadena As String
    Dim cantidad As String
    Dim nombreCombo As String

    ' Verifica si hay un elemento seleccionado en el historial
    If Not IsNull(lbHistorial.Value) Then
```

```
        ' Obtiene la cadena seleccionada del historial
        cadena = lbHistorial.Value
        ' Extrae la cantidad del combo
        cantidad = Mid(cadena, 3, InStr(1, cadena, ")") - 3)
        ' Extrae el nombre del combo
        nombreCombo = Mid(cadena, InStr(1, cadena, ")") + 2, InStr(1, cadena, "($") -
InStr(1, cadena, ")") - 3)

        ' Actualiza el valor de la cantidad y el nombre del combo en la interfaz
        txtCantidad.Text = cantidad
        cbSelect.Value = nombreCombo
        ' Actualiza la previsualización
        Prevista
    End If
End Sub

' Subrutina para actualizar el total de la cotización en el registro local.
Sub totCotizacion() ' Cambios sobre el total de la cotizacion (registro local)
    ' Actualiza el total de la cotización sumando el precio total actual
    cotizacionTot = cotizacionTot + totalPrecio
    ' Limpia el ListBox de total y agrega el nuevo total formateado
    lbTotal.Clear
    lbTotal.AddItem "Cotizacion Total: $" & Format(cotizacionTot, "0.00")
End Sub

' Función para verificar si una colección contiene un valor específico.
Function Contiene(col As Collection, valor As Variant) As Boolean ' colector para las
letras iniciales de los numeros en serie
    Dim elem As Variant
    ' Itera a través de los elementos de la colección
    For Each elem In col
        ' Comprueba si el elemento actual es igual al valor buscado
        If elem = valor Then
            Contiene = True
            Exit Function
        End If
    Next elem
    ' Si no se encuentra el valor, devuelve False
    Contiene = False
End Function

' Subrutina para generar una cotización en formato PDF.
Sub CotizacionPDF()
    Dim ruta As String ' Ruta de archivo para guardar el PDF
    Dim i As Integer ' Variable de iteración
    Dim NumeroCotizacion As String ' Número de cotización
    Dim rng As Range ' Rango de celdas
    Dim cell As Range ' Variable de celda
    Dim nombreHoja As String ' Nombre de la hoja de cálculo
```

```
    ' Agrega una hoja de cálculo oculta y obtiene su nombre
    Set ws = ThisWorkbook.Sheets.Add
    ws.Visible = xlSheetHidden
    nombreHoja = ws.Name

    ' Extrae el número de cotización de la hoja de cálculo
    For i = 1 To Len(nombreHoja)
        If IsNumeric(Mid(nombreHoja, i, 1)) Then
            NumeroCotizacion = NumeroCotizacion & Mid(nombreHoja, i, 1)
        End If
    Next i

    ' Define la ruta de archivo para guardar el PDF
    ruta = ThisWorkbook.Path & "\Cotizaciones\" & "Cotizacion-" & NumeroCotizacion &
"-" & Day(Date) & "." & Month(Date) & "." & Year(Date) & ".pdf"

    ' Muestra un mensaje con la ruta de archivo
    MsgBox "Al terminar la evaluacion se abrira automatico. " & ruta, vbInformation,
"Consola"

    ' Formatea y añade el título "COTIZACIÓN" a la hoja de cálculo
    Set rng = ws.Range("D1:E1")
    rng.Merge
    With rng
        .Value = "COTIZACIÓN"
        .HorizontalAlignment = xlHAlignRight
        .VerticalAlignment = xlVAlignBottom
        .Font.Color = RGB(31, 70, 120)
        .Font.Size = 20
        .RowHeight = rng.RowHeight * 2
        .Columns.AutoFit
    End With

    ' Añade una imagen a la hoja de cálculo
    On Error GoTo ManejarError
    Dim IMG As Shape
    With ws.Range("A1")
        Set IMG =
ws.Shapes.AddPicture(Filename:="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9G
cT41OXjDy-UdA42TTCttqmLlbkQ65wlVN10MGPAjJJujQ&s", _
                             LinkToFile:=False, _
                             SaveWithDocument:=True, _
                             Left:=ws.Range("A1").Left + ws.Range("A1").Width - 1,
_
                             Top:=ws.Range("A1").Top, _
                             Width:=-1, _
                             Height:=-1)
        IMG.LockAspectRatio = msoTrue
```

```
        IMG.Height = .Height
    End With
ManejarError:
    Err.Clear

    ' Formatea la sección "CLIENTE" de la cotización
    Set rng = ws.Range("A9:B9")
    rng.Merge
    With rng
        .Value = "CLIENTE"
        .Font.Color = RGB(255, 255, 255)
        .Interior.Color = RGB(31, 70, 120)
        .Font.Size = 14
    End With

    ' Completa la información del cliente y la cotización
    Set rng = ws.Range("E3:E6")
    For Each cell In rng
        Select Case cell.Address
            Case "$E$3":
                cell.Value = Date
            Case "$E$4":
                cell.Value = NumeroCotizacion
            Case "$E$5":
                cell.Value = InicioSesion.MiVariable(1)
            Case "$E$6":
                cell.Value = DateAdd("m", 3, Date)
            Case Else:
                cell.Value = ""
        End Select
        With cell
            .HorizontalAlignment = xlHAlignCenter
            .Borders.LineStyle = xlContinuous
            .Borders.Color = RGB(0, 0, 0)
            .Borders.Weight = xlThin
            .Columns.AutoFit
        End With
    Next cell

    ' Define los encabezados de las columnas de la tabla de productos
    Set rng = ws.Range("A16:H16")
    For Each cell In rng
        Select Case cell.Address
            Case "$A$16":
                cell.Value = "CODIGO"
            Case "$B$16":
                cell.Value = "DESCRIPCIÓN"
            Case "$C$16":
                cell.Value = "CANT."
```

```
            Case "$D$16":
                cell.Value = "PRECIO"
            Case "$E$16":
                cell.Value = "SUB-TOTAL"
            Case "$F$16":
                cell.Value = "ISR"
            Case "$G$16":
                cell.Value = "IVA"
            Case "$H$16":
                cell.Value = "TOTAL"
            Case Else:
                cell.Value = ""
        End Select
        With cell
            .HorizontalAlignment = xlHAlignCenter
            .Font.Color = RGB(255, 255, 255)
            .Interior.Color = RGB(31, 70, 120)
            .Font.Size = 14
            .Columns.AutoFit
        End With
    Next cell

    ' Completa la información adicional del cliente y la cotización
    Set rng = ws.Range("A2:A7")
    For Each cell In rng
        Select Case cell.Address
            Case "$A$2":
                cell.Value = InicioSesion.MisDatos(2)
            Case "$A$3":
                cell.Value = "Ciudad: " & InicioSesion.MisDatos(3)
            Case "$A$4":
                cell.Value = "Sitio Web: https://portal.ingenieria.usac.edu.gt/"
            Case "$A$5":
                cell.Value = "Teléfono: " & InicioSesion.MisDatos(4)
            Case "$A$6":
                cell.Value = "E-mail: " & InicioSesion.MisDatos(5)
            Case "$A$7":
                cell.Value = "Asesor de venta: " & InicioSesion.MisDatos(1)
                cell.Columns.AutoFit
            Case Else:
                cell.Value = ""
        End Select
        With cell
            .Font.Size = 14
        End With
    Next cell

    ' Completa la información del cliente adicional
    Set rng = ws.Range("A10:A14")
```

```
For Each cell In rng
    Select Case cell.Address
        Case "$A$10":
            cell.Value = "Nombre: " & InicioSesion.MiVariable(2)
        Case "$A$11":
            cell.Value = InicioSesion.MiVariable(3)
        Case "$A$12":
            cell.Value = "Ciudad: " & InicioSesion.MiVariable(4)
        Case "$A$13":
            cell.Value = "Teléfono: " & InicioSesion.MiVariable(5)
        Case "$A$14":
            cell.Value = "E-mail: " & InicioSesion.MiVariable(6)
        Case Else:
            cell.Value = ""
    End Select
    With cell
        .Font.Size = 14
    End With
Next cell

' Formatea la sección de impuestos
Set rng = ws.Range("G4:G5")
For Each cell In rng
    ws.Range("G4").Value = "12%"
    ws.Range("G5").Value = "5%"
    With cell
        .HorizontalAlignment = xlHAlignCenter
        .Font.Size = 14
        .Borders(xlEdgeBottom).Weight = xlThick
    End With
Next cell

' Añade encabezados a la sección de impuestos
With ws.Range("G3")
    .Font.Size = 14
    .Value = "IMPUESTOS"
    .HorizontalAlignment = xlHAlignCenter
    .Font.Bold = True
    .Columns.AutoFit
End With

' Completa la información adicional de la cotización
Set rng = ws.Range("D3:D6")
For Each cell In rng
    Select Case cell.Address
        Case "$D$3":
            cell.Value = "FECHA"
        Case "$D$4":
            cell.Value = "COTIZACIÓN #"
```

```
            Case "$D$5":
                cell.Value = "ID CLIENTE"
            Case "$D$6":
                cell.Value = "VALIDO HASTA"
            Case Else:
                cell.Value = ""
        End Select
        With cell
            .HorizontalAlignment = xlHAlignRight
            .Columns.AutoFit
            .Font.Size = 12
        End With
    Next cell

    ' Variables para la información de la cotización
    Dim Rango As String
    Dim cadena As String
    Dim cantidad As String
    Dim nombreCombo As String
    Dim precioString As String
    Dim CODIGO As String
    Dim DESCRIPCION As String
    Dim precioNeto As Double
    Dim TotalBruto As Double
    Dim TotalImpuesto As Double
    Dim TotalNeto As Double

    ' Itera sobre los elementos en el historial para crear la cotización
    For i = 0 To lbHistorial.ListCount - 1
        If Not IsNull(lbHistorial.List(i)) Then
            ' Extrae la información del historial
            cadena = lbHistorial.List(i)
            cantidad = Mid(cadena, 3, InStr(1, cadena, ")") - 3)
            nombreCombo = Mid(cadena, InStr(1, cadena, ")") + 2, InStr(1, cadena,
"($") - InStr(1, cadena, ")") - 3)
            precioString = Mid(cadena, InStr(InStr(InStr(InStr(InStr(cadena, "("),
cadena, " "), cadena, " "), cadena, "("), cadena, "$") + 1)
        End If
        ' Calcula el precio neto
        precioNeto = (Val(precioString) / (1 + 0.05 + 0.12)) / Val(cantidad)
        Rango = "$A$" & 17 + i & ":$H$" & 16 + Val(lbHistorial.ListCount)
        Set rng = ws.Range(Rango)
        CODIGO = ""
        DESCRIPCION = ""
        ' Completa la tabla de productos
        For Each cell In rng
            ' Formatea las filas alternas de la tabla
            If cell.row Mod 2 = 0 Then
                cell.EntireRow.Interior.Color = RGB(220, 220, 220)
```

```
        Else
            cell.EntireRow.Interior.colorIndex = xlNone
        End If
        ' Verifica si es un combo de productos o un producto individual
        If InStr(1, nombreCombo, "Combo", vbTextCompare) > 0 Then
            Set wsa = ThisWorkbook.Sheets("Productos")
            Set tbl = wsa.ListObjects("Catalogo")
            Set col = tbl.ListColumns("CODIGO")

            Dim Articulos As New Collection
            For Each celda In col.DataBodyRange.Cells
                Dim articulo As String
                articulo = wsa.Cells(celda.row, celda.Column + 1).Value
                If Left(celda.Value, 1) = Right(nombreCombo, 1) Then
                    If Not Contiene(Articulos, articulo) Then
                        Articulos.Add articulo
                        If Articulos.Count > 1 Then
                            DESCRIPCION = DESCRIPCION & " (-) " & articulo
                        Else
                            DESCRIPCION = DESCRIPCION & articulo
                        End If
                    End If
                End If
            Next celda
            Set Articulos = New Collection
            CODIGO = nombreCombo
        Else
            DESCRIPCION = nombreCombo
            CODIGO = "Producto"
        End If
        ' Ajusta la longitud de la descripción
        If Len(DESCRIPCION) < 31 Then
            DESCRIPCION = DESCRIPCION
        Else
            DESCRIPCION = Left(DESCRIPCION, 25) & " [...]"
        End If
        ' Completa las celdas de la tabla
        Select Case cell.Column
            Case 1:
                cell.Value = CODIGO
                cell.HorizontalAlignment = xlHAlignCenter
            Case 2:
                cell.Value = DESCRIPCION
            Case 3:
                cell.Value = cantidad
            Case 4:
                cell.Value = precioNeto
                cell.NumberFormat = "#,##0.00 Q"
            Case 5:
```

```
                cell.Value = precioNeto * Val(cantidad)
                cell.NumberFormat = "#,##0.00 Q"
                TotalBruto = TotalBruto + Val(precioString)
            Case 6:
                cell.Value = precioNeto * 0.05
                cell.NumberFormat = "#,##0.00 Q"
                TotalImpuesto = TotalImpuesto + precioNeto * 0.05
            Case 7:
                cell.Value = precioNeto * 0.12
                cell.NumberFormat = "#,##0.00 Q"
                TotalImpuesto = TotalImpuesto + precioNeto * 0.12
            Case 8:
                cell.Value = Val(precioString)
                cell.NumberFormat = "#,##0.00 Q"
                TotalNeto = TotalNeto + Val(precioString)
            Case Else:
                cell.Value = "N/A"
        End Select
        cell.Font.Size = 14
    Next cell
Next i

' Obtiene la última fila de la tabla
Dim LastRow As Long
LastRow = rng.Rows(rng.Rows.Count).row + 1

' Añade totales a la tabla
With ws.Range("F" & LastRow)
    .Value = "SubTotal"
    .Font.Bold = True
    .Font.Size = 14
End With
With ws.Range("F" & LastRow + 1)
    .Value = "Impuesto Total"
    .Font.Bold = True
    .Font.Size = 14
End With
With ws.Range("F" & LastRow + 2)
    .Value = "Total"
    .Font.Bold = True
    .Font.Size = 14
    .Borders(xlEdgeTop).Weight = xlThin
    .Borders(xlEdgeBottom).Weight = xlThick
End With

' Completa los valores de los totales
With ws.Range("H" & LastRow)
    .Value = TotalBruto - TotalImpuesto
    .Font.Bold = True
```

```
        .NumberFormat = "#,##0.00 Q"
        .Font.Size = 14
    End With
    With ws.Range("H" & LastRow + 1)
        .Value = TotalImpuesto
        .Font.Bold = True
        .NumberFormat = "#,##0.00 Q"
        .Font.Size = 14
    End With
    With ws.Range("H" & LastRow + 2)
        .Value = TotalBruto
        .Font.Bold = True
        .NumberFormat = "#,##0.00 Q"
        .Font.Size = 14
    End With

    ' Ajusta el ancho de las columnas
    ws.Columns("A:H").AutoFit

    ' Guarda la hoja de cálculo como PDF
    ws.ExportAsFixedFormat Type:=xlTypePDF, Filename:=ruta,
Quality:=xlQualityStandard, IncludeDocProperties:=True, IgnorePrintAreas:=False,
OpenAfterPublish:=True

    ' Elimina la hoja de cálculo temporal
    Application.DisplayAlerts = False
    ws.Delete
    Application.DisplayAlerts = True
End Sub
```

Explicación de CotizacionPDF()

```
Este código realiza las siguientes tareas
1. Crea una nueva hoja de cálculo oculta para construir la cotización en formato PDF.
2. Extrae el número de cotización de la hoja de cálculo.
3. Define la ruta de archivo para guardar el PDF.
4. Formatea y añade el título "COTIZACIÓN" a la hoja de cálculo.
5. Añade una imagen a la hoja de cálculo.
6. Formatea la sección "CLIENTE" de la cotización.
7. Completa la información del cliente y la cotización.
8. Define los encabezados de las columnas de la tabla de productos.
9. Completa la información adicional del cliente y la cotización.
10. Completa la información del cliente adicional.
11. Formatea la sección de impuestos.
12. Añade encabezados a la sección de impuestos.
13. Completa la información adicional de la cotización.
14. Itera sobre los elementos en el historial para crear la cotización.
15. Añade totales a la tabla.
16. Completa los valores de los totales.
```

17. Ajusta el ancho de las columnas.
18. Guarda la hoja de cálculo como PDF.
19. Elimina la hoja de cálculo temporal.