SISTEMA DE REPARTO

-PROYECTO DIDACTICO-

SISTEMA DE REPARTO G6

Manual Técnico

Guatemala, 09 de agosto del 2024

INDICE

INTRODUCCIÓN

Este manual técnico está diseñado para proporcionar una descripción exhaustiva del sistema de reparto, incluyendo los programas **AdminRepartoApp**, **ClientesRepartoApp**, y **PilotosRepartoApp**. Este documento detalla los aspectos técnicos del desarrollo e implementación del sistema, abarcando desde la identificación del problema hasta la solución final, describiendo el diseño del software, su arquitectura, y las bibliotecas utilizadas. El objetivo principal es ofrecer una guía completa que permita comprender y mantener el sistema de manera eficiente.

OBJETIVOS

- **Proveer una solución integral de gestión de reparto** que permita la administración efectiva de pedidos, la interacción directa con los clientes y la optimización de las rutas de los pilotos.
- **Garantizar una experiencia de usuario óptima** mediante el desarrollo de aplicaciones especializadas para administradores, clientes y pilotos, que faciliten la operación y el uso del sistema.
- **Implementar un sistema escalable y mantenible** que pueda adaptarse a las necesidades futuras del negocio, incluyendo la integración con otras plataformas y la ampliación de funcionalidades.
- **Asegurar la integridad y seguridad de los datos** a través de una base de datos robusta y procedimientos de conexión seguros.

CONTENIDO

DESCRIPCIÓN DEL PROBLEMA

La necesidad planteada por la empresa es desarrollar un software integral que cubra las operaciones de gestión de un sistema de reparto, abarcando a todos los actores involucrados en el proceso: administradores, pilotos y clientes. El sistema propuesto debe ser capaz de proporcionar una solución eficaz y eficiente para la gestión de pedidos y entregas, optimizando la interacción entre estos tres tipos de usuarios:

1. **Aplicación Administrativa**:

2. Esta aplicación está dirigida a los administradores y usuarios con roles de supervisión. Se requiere una solución que permita realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los diferentes elementos del sistema, como usuarios, pedidos, rutas y pilotos. Además, la aplicación debe ser capaz de gestionar el flujo de trabajo diario, permitiendo la asignación de rutas a los pilotos, el seguimiento del estado de las entregas y la generación de reportes para la toma de decisiones.

3. **Aplicación de Pilotos**:

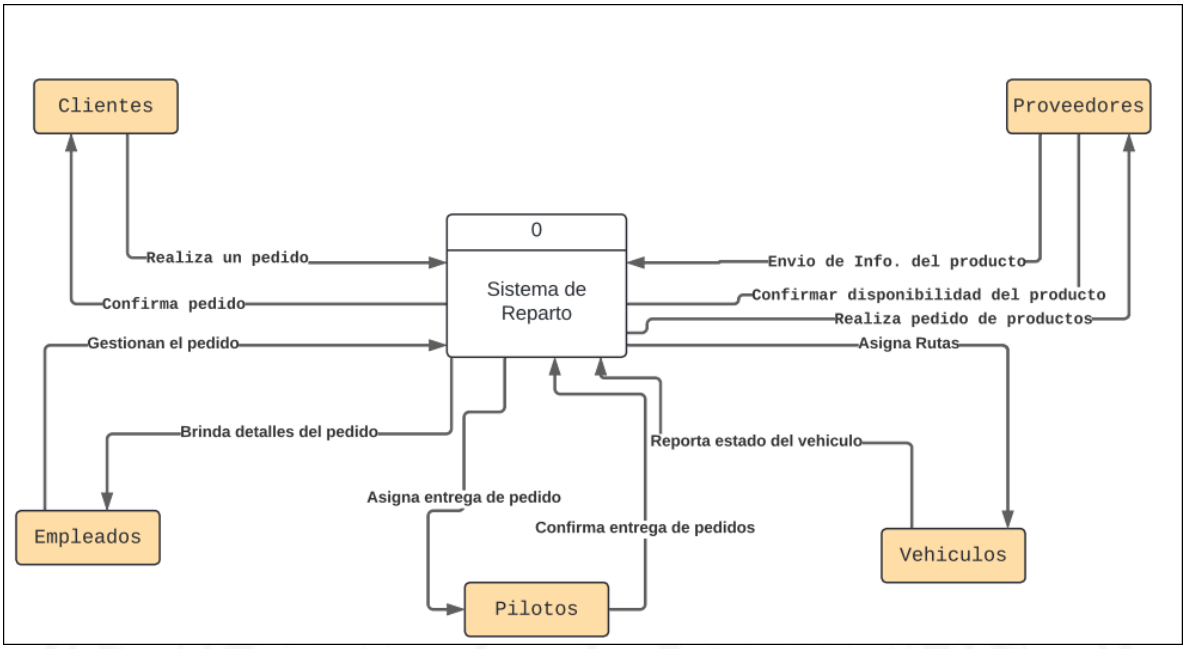
4. La aplicación destinada a los pilotos debe permitirles acceder de manera sencilla y rápida a la información sobre sus rutas asignadas. Es fundamental que los pilotos puedan verificar las rutas que les han sido asignadas, ya sean diarias, semanales o quincenales, y marcar las entregas que han realizado. La aplicación debe facilitar la navegación a través de las rutas, integrando la API de Google Maps para optimizar el trayecto y mejorar la eficiencia en las entregas.

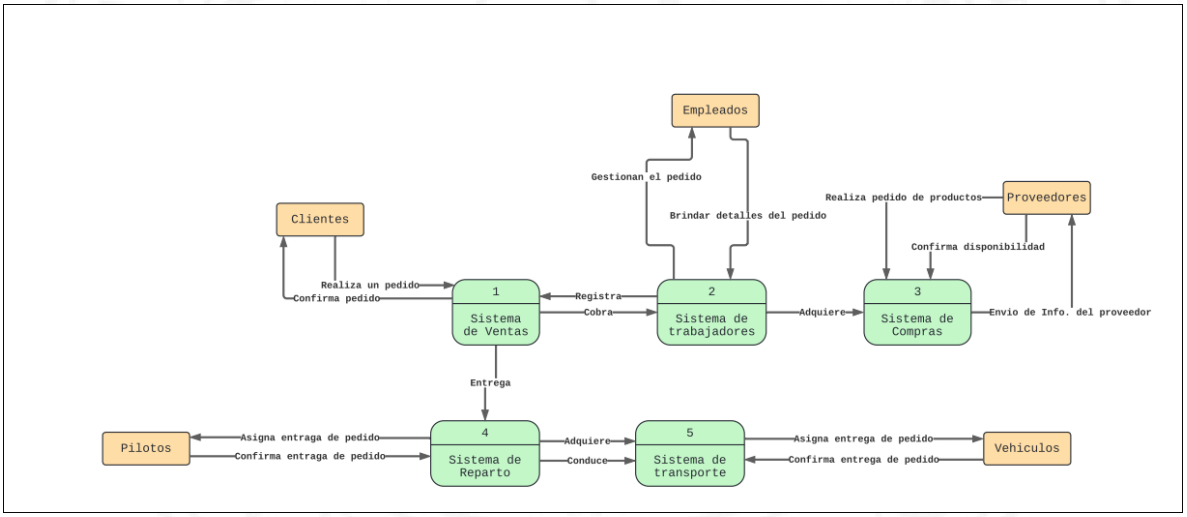
5. **Aplicación para Clientes**:

   Esta aplicación está enfocada en los clientes, quienes deben poder realizar pedidos de manera intuitiva, especificando los detalles de recogida y entrega. Además, los clientes necesitan tener acceso a la información sobre el estado de sus pedidos en tiempo real, permitiendo hacer un seguimiento detallado del proceso de entrega. Otra funcionalidad clave es la posibilidad de valorar la calidad del servicio y la entrega del producto, lo cual es esencial para mantener la satisfacción del cliente y mejorar el servicio continuamente.

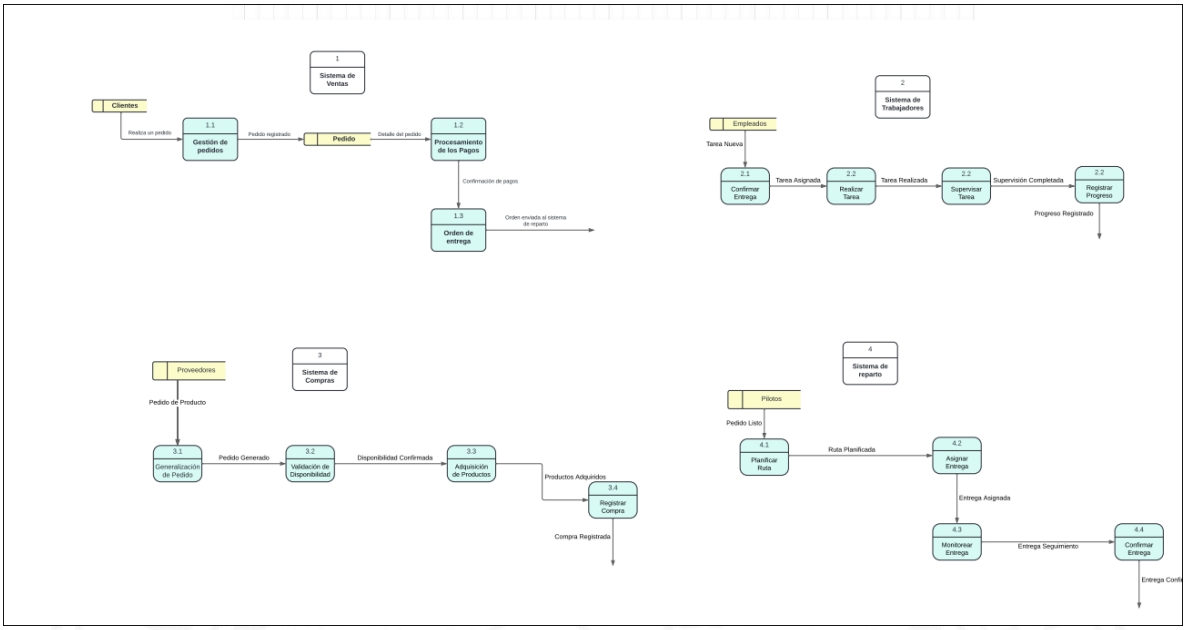
El problema global, por lo tanto, radica en la creación de un sistema cohesivo que integre estas tres aplicaciones, cada una de las cuales responde a las necesidades específicas de un tipo de usuario dentro del proceso de reparto. La solución debe ser robusta, segura y fácil de usar, asegurando una comunicación efectiva entre los administradores, pilotos y clientes para lograr un proceso de entrega eficiente y satisfactorio.

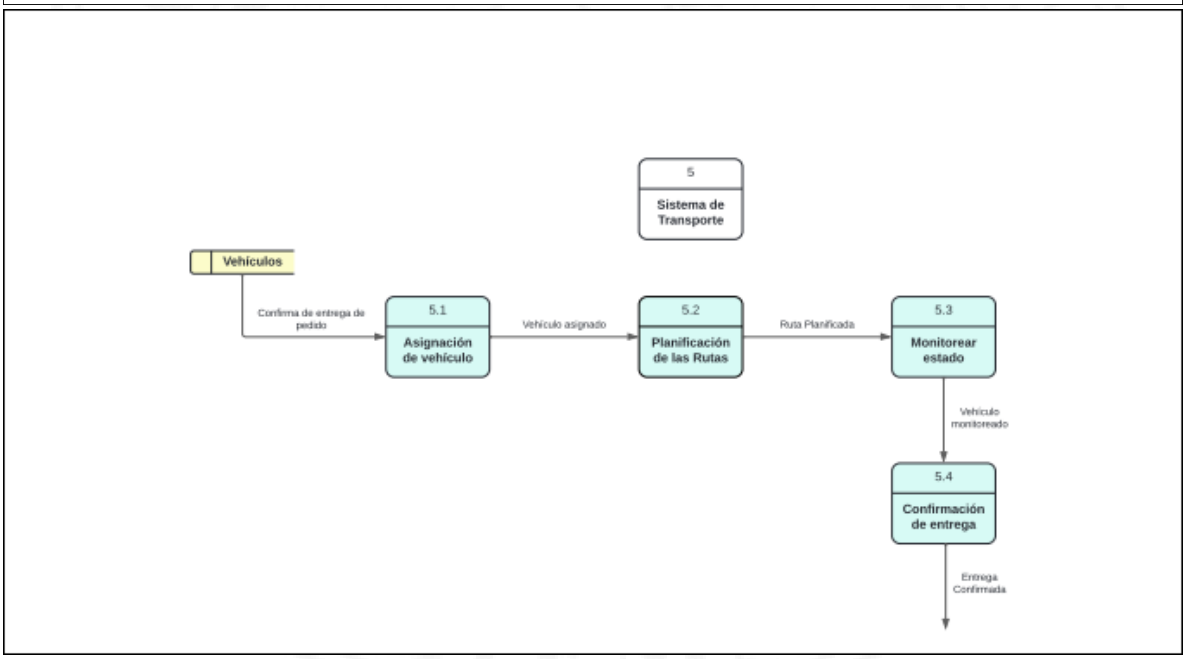
DIAGRAMAS DE CONTEXTO Y CERO



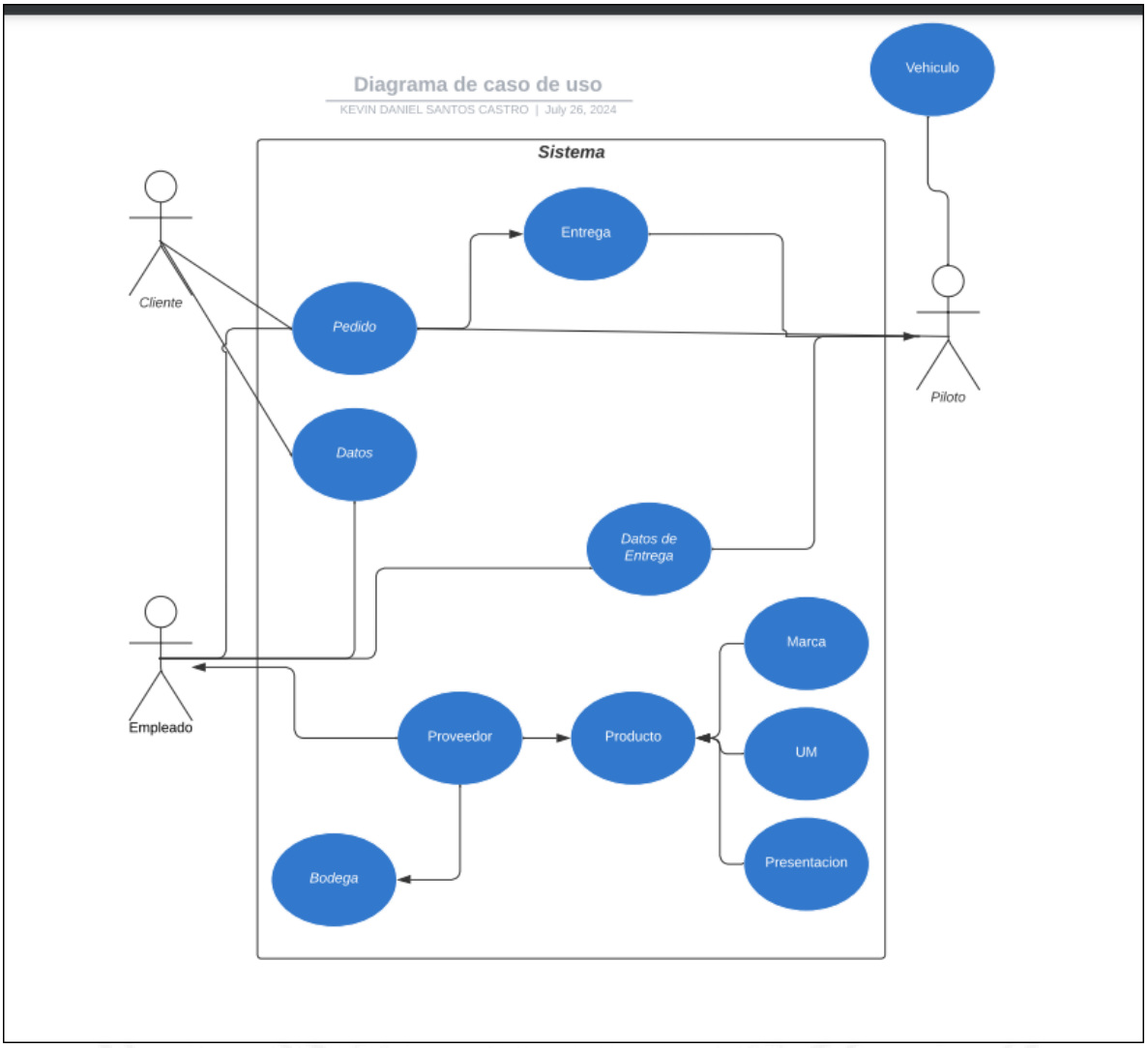


DIAGRAMAS HIJOS





DIAGRAMAS DE CASOS DE USO



ANÁLISIS DE REQUERIMIENTOS DEL USUARIO

REQUERIMIENTOS FUNCIONALES:

1. **Gestión de Pedidos**: El sistema debe permitir a los administradores crear, actualizar y eliminar pedidos, así como asignarlos a pilotos específicos. Los clientes deben poder realizar pedidos y ver el estado de sus entregas en tiempo real.

2. **Gestión de Usuarios**: Debe existir la funcionalidad para gestionar diferentes tipos de usuarios (administradores, clientes y pilotos), incluyendo el registro, autenticación y control de permisos.

3. **Optimización de Rutas**: La aplicación de pilotos debe permitir la visualización y optimización de las rutas para las entregas, utilizando la API de Google Maps para generar las mejores rutas posibles.

4. **Seguimiento en Tiempo Real**: Los clientes deben poder rastrear sus pedidos en tiempo real, desde la preparación hasta la entrega final.

5. **Generación de Reportes**: El sistema debe ser capaz de generar reportes detallados sobre el estado de los pedidos, la eficiencia de las entregas y otros aspectos operativos.

REQUERIMIENTOS NO FUNCIONALES:

1. **Escalabilidad**: El sistema debe ser escalable para manejar un número creciente de usuarios y pedidos sin pérdida de rendimiento.

2. **Seguridad**: Debe garantizarse la seguridad de los datos del usuario mediante autenticación segura y encriptación de la información sensible.

3. **Usabilidad**: La interfaz de usuario debe ser intuitiva y fácil de usar para todos los tipos de usuarios, minimizando la necesidad de formación.

4. **Disponibilidad**: El sistema debe estar disponible 24/7, con un tiempo de inactividad mínimo para mantenimientos y actualizaciones.

5. **Compatibilidad**: El sistema debe ser compatible con las versiones más recientes de los sistemas operativos Windows y las versiones específicas de .NET Framework y MySQL.

PLANTEAMIENTO INICIAL DE LA SOLUCIÓN

ARQUITECTURA DEL SISTEMA:

El sistema de reparto está compuesto por tres aplicaciones principales: **AdminRepartoApp**, **ClientesRepartoApp**, y **PilotosRepartoApp**. Estas aplicaciones se conectan a una base de datos centralizada en MySQL y están desarrolladas en .NET Framework.

- **AdminRepartoApp**: Es la aplicación principal para los administradores, diseñada para gestionar los pedidos, usuarios y generar reportes. Esta aplicación también gestiona las configuraciones globales del sistema.
- **ClientesRepartoApp**: Es la aplicación orientada a los clientes, permitiéndoles realizar pedidos, gestionar su perfil y hacer seguimiento de sus compras.
- **PilotosRepartoApp**: Es la aplicación diseñada para los pilotos, donde pueden ver sus rutas, actualizar el estado de las entregas y comunicarse con los clientes.

TECNOLOGÍAS UTILIZADAS:

1. **MySQL**: Se utilizó MySQL para la gestión de la base de datos por su eficiencia y compatibilidad con aplicaciones de .NET. Esta tecnología permite una gestión robusta de los datos, incluyendo la autenticación de usuarios y el almacenamiento de pedidos y rutas.
2. **.NET Framework**: La elección de .NET Framework permite el desarrollo de aplicaciones ricas en funcionalidades con interfaces de usuario atractivas y la capacidad de integrar diversas bibliotecas y APIs, como Google Maps para la optimización de rutas.
3. **Google Maps API**: Utilizada en la aplicación de pilotos para calcular y mostrar las rutas más eficientes para las entregas.
4. **Visual Studio 2019**: Herramienta de desarrollo utilizada para construir, depurar y mantener las aplicaciones.

DECISIONES CLAVE DE DISEÑO:

1. **Modularización**: Cada funcionalidad principal del sistema está aislada en su propio módulo o aplicación, lo que facilita la escalabilidad y el mantenimiento.
2. **Uso de Bibliotecas**: Se eligieron bibliotecas como MySql.Data para la manipulación de la base de datos y System.Windows.Forms para las interfaces de usuario. Estas elecciones permiten un desarrollo más rápido y confiable.
3. **Interfaz de Usuario**: Se decidió utilizar un enfoque de diseño simple y coherente para todas las aplicaciones, garantizando que los usuarios, independientemente de su rol, puedan operar el sistema de manera eficiente.
4. **Seguridad**: Se implementaron medidas de seguridad tanto en el acceso a las aplicaciones como en la gestión de los datos, utilizando cifrado y autenticación segura.

ADMINSYSTEMAPP EXPLICACIÓN

AdminSystemApp

Form1(login)

Descripción General

El formulario frmLogin es la pantalla inicial de la aplicación AdminRepartoApp, diseñada para gestionar la autenticación de usuarios mediante la verificación de sus credenciales en una base de datos MySQL. Este formulario no solo permite el acceso seguro a la aplicación, sino que también registra las actividades de inicio de sesión para auditoría y seguridad.

Componentes y Funcionalidades Clave

1. **Inicialización del Formulario:**
    - **Constructor frmLogin:**
        - Se configura el formulario y se asocian manejadores de eventos para los campos de texto del correo y la contraseña. Estos manejadores permiten que el usuario pueda presionar "Enter" para intentar iniciar sesión, mejorando la usabilidad del formulario.
2. **Bienvenida Personalizada:**
    - **Método frmLogin_Load:**
        - Al cargar el formulario, se genera y muestra un mensaje de bienvenida que varía según la hora del día, proporcionando un toque personalizado a la experiencia del usuario.
3. **Validación de Credenciales:**
    - **Método btnLogin_Click_1:**
        - **Validación Previa:** Comprueba que el correo y la contraseña no estén vacíos y que el formato del correo sea correcto usando una expresión regular.
        - **Conexión a MySQL:** Se establece una conexión con la base de datos utilizando una cadena de conexión definida.
        - **Consulta y Verificación:** Se ejecuta una consulta SQL para verificar si las credenciales del usuario coinciden con las almacenadas en la base de datos.
        - **Acceso Concedido:** Si las credenciales son correctas, el formulario muestra un mensaje de bienvenida y abre el formulario principal (frmMain), pasando la información necesaria sobre el usuario (como permisos de edición y tipo de empleado).
4. **Manejo de Intentos Fallidos:**
    - Se incrementa un contador cada vez que las credenciales son incorrectas.
    - Tras tres intentos fallidos, se registra un intento fallido en la base de datos, incluyendo detalles como la dirección IP del usuario y el nombre del equipo.
5. **Registro de Actividades:**

- **Método RegistrarLog:** Registra los inicios de sesión exitosos en una tabla LogActividades, proporcionando una traza de auditoría de quién ha accedido al sistema y cuándo.
- **Método RegistrarIntentoFallido:** Similar al anterior, pero registra intentos fallidos, lo cual es crucial para detectar posibles intentos de acceso no autorizados.

6. **Funciones Auxiliares:**
   - **ObtenerDireccionIP:** Recupera la dirección IP local del equipo para su inclusión en los registros de actividad.
   - **ObtenerMensajeBienvenida:** Genera el mensaje de bienvenida de acuerdo a la hora del día.
7. **Interacción del Usuario:**
   - **Manejo de Eventos de Teclado:** Permite una interacción fluida al permitir que el usuario presione "Enter" para iniciar sesión.
   - **Botón de Salida (btnExit_Click):** Solicita confirmación antes de cerrar la aplicación, evitando cierres accidentales.

frmMain - Kevin

El formulario frmMain de la aplicación AdminRepartoApp actúa como el panel principal de control una vez que el usuario ha iniciado sesión. Está diseñado para proporcionar acceso a diversas funcionalidades y gestionar diferentes aspectos del sistema de reparto. Aquí tienes un desglose detallado de su funcionamiento:

Descripción General

El formulario frmMain es la interfaz principal que aparece después del inicio de sesión exitoso. Proporciona enlaces a diferentes áreas de la aplicación, como la gestión de productos, usuarios, pedidos, reportes, etc. La visibilidad y funcionalidad de ciertos elementos se personalizan según el rol del usuario (administrador o no).

Componentes y Funcionalidades Clave

1. **Inicialización del Formulario:**
    - **Constructores frmMain:**
        - El constructor principal recibe varios parámetros (correo del usuario, permisos de edición, si es administrador, y el ID del empleado) que son utilizados para personalizar la experiencia del usuario.
        - Hay un constructor por defecto que inicializa el componente gráfico.
2. **Carga del Formulario (frmMain_Load):**
    - **Etiqueta de Usuario:** Se muestra el correo del usuario que ha iniciado sesión en una etiqueta (lblUsuarioIniciado).
    - **Fecha y Hora:** Inicia un temporizador (timerFechaHora) para mostrar continuamente la fecha y hora actual.
    - **Visibilidad del Botón de Logs:** El botón para acceder a los logs se muestra solo si el usuario es un administrador, restringiendo así el acceso a esta función.
3. **Funcionalidades de Botones:**
    - Cada botón en el formulario abre otro formulario relacionado con una funcionalidad específica de la aplicación, como la gestión de productos, usuarios, pedidos, y más.
    - **Ejemplos:**
        - **btnProductos_Click:** Abre el formulario para gestionar productos (frmProducto).
        - **btnLogs_Click:** Abre el formulario de logs (frmLogs), visible solo para administradores.
        - **btnCerrarSesion_Click:** Cierra la sesión actual y muestra el formulario de inicio de sesión (frmLogin).
4. **Manejo de Errores:**

- Se utilizan bloques try-catch al intentar abrir cada formulario para capturar y mostrar mensajes de error en caso de que ocurran problemas al abrir cualquier formulario.

5. **Consulta a la Base de Datos:**
    - **ObtenerCorreosUsuarios:** Método privado que obtiene una lista de correos electrónicos de todos los usuarios en la base de datos, lo que podría ser útil para tareas administrativas o validaciones.
6. **Actualización de Fecha y Hora:**
    - **timerFechaHora_Tick:** Actualiza la etiqueta de fecha y hora (lblFechaHora) cada vez que el temporizador hace "tick", mostrando la hora y fecha actuales en el formato "HH dd/MM/yyyy".

frmGestionPedidos - Anika

El formulario frmGestionPedidos permite a los usuarios ver, buscar, y actualizar la información relacionada con los pedidos. Los datos de los pedidos se extraen de una base de datos MySQL y se presentan al usuario a través de una interfaz de usuario intuitiva que incluye elementos como DataGridView, ComboBox, y varios campos de texto.

Componentes y Funcionalidades Clave

1. **Inicialización del Formulario:**
    - **Constructor frmGestionPedidos:**
        - Inicializa los componentes gráficos del formulario.
        - Carga los pedidos en un ComboBox para facilitar su selección.
        - Llena un DataGridView con los datos de todos los pedidos existentes.
        - Deshabilita los campos de entrada de pedidos y clientes inicialmente, hasta que se seleccione un pedido.
2. **Carga de Datos:**
    - **CargarComboboxPedidos:** Llena el ComboBox (cmbBuscarPedido) con los números de factura de los pedidos, permitiendo al usuario seleccionar un pedido específico.
    - **CargarDatosPedidos:** Rellena un DataGridView (dgvPedidos) con todos los pedidos existentes, proporcionando una visión general de la información de los pedidos almacenados en la base de datos.
3. **Interacción del Usuario:**
    - **cmbBuscarPedido_SelectedIndexChanged:** Cuando el usuario selecciona un número de factura del ComboBox, este método carga y muestra los detalles del pedido y la información del cliente relacionada en los campos correspondientes.
    - **btnActualizarPedido_Click:** Permite al usuario actualizar la información de un pedido seleccionado en la base de datos. Antes de proceder, verifica que un pedido esté seleccionado y actualiza los registros de la base de datos con los valores proporcionados.
4. **Manejo de la Base de Datos:**
    - Utiliza conexiones a MySQL para ejecutar comandos SQL y obtener o actualizar datos de la base de datos. Los métodos MySqlConnection, MySqlCommand, y MySqlDataAdapter son empleados para manejar estas operaciones.
5. **Manejo de Errores:**
    - Se utilizan bloques try-catch para capturar y manejar excepciones que puedan ocurrir durante las operaciones con la base de datos, mostrando mensajes de error informativos a los usuarios.
6. **Habilitación y Limpieza de Campos:**

- **HabilitarCamposPedido y HabilitarCamposCliente:** Controlan la habilitación de los campos de entrada, activándolos o desactivándolos según las necesidades.
- **LimpiarCamposPedido y LimpiarCamposCliente:** Limpian los campos de entrada después de que se actualiza un pedido, preparando la interfaz para una nueva operación.

Resumen

El formulario frmGestionPedidos proporciona una interfaz para visualizar y actualizar la información de los pedidos dentro del sistema. Facilita la gestión eficiente de los pedidos al permitir a los usuarios acceder rápidamente a los detalles de cada pedido y modificarlos según sea necesario. Esta funcionalidad es esencial para mantener actualizada la base de datos con información precisa y relevante sobre los pedidos.

frmVerProductos - Pablo

El formulario frmVerProductos está diseñado para gestionar la información de productos en una aplicación de gestión de reparto. Ofrece funcionalidades para visualizar, actualizar y eliminar productos, y para manejar las imágenes asociadas a cada producto.

Funcionalidades Principales

1. **Inicialización y Configuración:**
    - **Constructor frmVerProductos:**
        - Establece el modo de ajuste de la imagen del producto a StretchImage para asegurar que cualquier imagen cargada se ajuste al tamaño del PictureBox.
        - Desactiva todos los campos de entrada al inicio para evitar cambios accidentales hasta que un producto sea seleccionado.
2. **Carga de Datos desde la Base de Datos:**
    - **CargarProductos:**
        - Conecta a la base de datos MySQL y ejecuta una consulta para obtener los nombres de todos los productos.
        - Rellena el ComboBox (cmbBuscarProducto) con los nombres de los productos, facilitando al usuario seleccionar un producto para visualizar o editar.
        - Muestra un mensaje si no se encuentran productos, alertando al usuario de una posible falta de datos.
3. **Visualización de Detalles del Producto:**
    - **cmbBuscarProducto_SelectedIndexChanged_1:**
        - Detecta el cambio de selección en el ComboBox y carga los detalles del producto seleccionado.
    - **CargarProducto:**
        - Recupera y muestra la información detallada del producto, incluyendo nombre, detalles, precio, IVA, comentarios, stock, marca, y otros campos relevantes.
        - Carga y muestra la imagen del producto si existe en la base de datos, manejando el byte[] de la imagen mediante un MemoryStream.
4. **Actualización de Productos:**
    - **btnUpdate_Click_1:**
        - Valida que los datos del producto sean correctos y completos antes de actualizar la base de datos.

        - Actualiza los detalles del producto en la base de datos MySQL, informando al usuario si la operación fue exitosa o no.

5. **Eliminación de Productos:**
    - **btnEliminarProducto_Click_1:**
        - Solicita confirmación al usuario antes de eliminar el producto seleccionado.
        - Elimina el producto de la base de datos si el usuario confirma la acción, actualizando la interfaz para reflejar la eliminación.
6. **Carga de Imágenes:**
    - **btnCargarImagen_Click_1:**
        - Permite al usuario seleccionar una nueva imagen para el producto a través de un OpenFileDialog.
        - Actualiza la imagen en la base de datos, utilizando un arreglo de bytes para almacenar la imagen.
7. **Validación de Datos:**
    - **ValidarCampos:**
        - **Verifica Campos Vacíos:** Asegura que ningún campo requerido esté vacío.
        - **Validaciones Numéricas:** Confirma que el precio, IVA y stock son números válidos y no negativos.
        - **Validaciones de Texto:** Usa expresiones regulares para asegurar que los nombres de productos y comentarios contienen solo caracteres permitidos:
            - **Nombre de Producto:** Permite letras, números, espacios, tildes y la letra ñ.
            - **Comentarios del Producto:** Permite letras, números, espacios, tildes, puntos y comas.
            - **Contacto de Marca:** Permite caracteres comunes en direcciones de correo electrónico y descripciones de contacto (incluyendo @).

Resumen de Características

- **Validaciones Rigurosas:** Protege la integridad de los datos mediante validaciones exhaustivas antes de cualquier modificación.
- **Gestión de Imágenes:** Proporciona una manera de actualizar visualmente los productos mediante la gestión de imágenes.
- **Interfaz Intuitiva:** Los campos solo se habilitan una vez que se ha seleccionado un producto, guiando al usuario a través del proceso de gestión de productos de manera clara y eficiente.

frmBodegasTransportes - Emanuel

El formulario frmBodegasTransportes en la aplicación AdminRepartoApp está diseñado para gestionar las operaciones logísticas relacionadas con la asignación de pedidos a pilotos, el manejo de envíos, y la generación de órdenes de compra. Aquí tienes una descripción detallada de su funcionamiento:

Descripción General

El formulario frmBodegasTransportes permite a los usuarios asignar pedidos sin piloto a pilotos disponibles, gestionar los estados de los envíos, y crear órdenes de compra para productos. Proporciona una interfaz para controlar estos aspectos logísticos de manera eficiente y precisa.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmBodegasTransportes:**
        - Llama a métodos para cargar las listas desplegables con valores necesarios al iniciar el formulario, como estados de envío, pedidos no asignados, pilotos disponibles, y productos.
2. **Carga de Datos en ComboBox:**
    - **CargarComboboxValores:** Llena el ComboBox cmbEstadoEnvio con estados predefinidos de los envíos, como "En Tránsito", "Entregado", etc.
    - **CargarComboboxPedidosNoAsignados:** Recupera pedidos de la base de datos que aún no han sido asignados a ningún piloto y los carga en el ComboBox cmbBuscarPedidoNoAsignado.
    - **CargarComboboxPilotos:** Llena el ComboBox cmbBuscarPiloto con pilotos disponibles que pueden ser asignados a pedidos.
    - **CargarComboboxProductos:** Llena el ComboBox cmbProductoOrden con productos disponibles para la generación de órdenes de compra.
3. **Interacción del Usuario:**
    - **cmbBuscarPedidoNoAsignado_SelectedIndexChanged:**
        - Cuando se selecciona un pedido sin asignar, este método carga los detalles del pedido y del cliente asociado, permitiendo al usuario ver y completar la información necesaria para la asignación.
    - **btnGuardarEnvio_Click:**
        - Asigna un pedido seleccionado a un piloto, actualiza la información del envío en la base de datos, y recarga la lista de pedidos no asignados.
    - **btnGuardarOrden_Click:**
        - Genera una orden de compra para un producto seleccionado con la cantidad especificada, insertando los detalles en la base de datos.

4. **Manejo de la Base de Datos:**
    - **Asignación de Pedidos:**
        - Actualiza la base de datos para reflejar la asignación de un piloto a un pedido mediante la ejecución de una sentencia SQL UPDATE.
        - Inserta un nuevo registro en la tabla de envíos (Envio) con los detalles de dirección y estado del envío.
    - **Generación de Órdenes de Compra:**
        - Inserta un nuevo registro en la tabla de órdenes de compra (Orden_Compra) con detalles sobre el producto, cantidad, fecha, y monto.
5. **Manejo de Errores:**
    - Uso de bloques try-catch para manejar excepciones durante las operaciones de base de datos, asegurando que cualquier error se muestre al usuario con mensajes informativos.
6. **Clases Auxiliares:**
    - **ComboBoxItem:** Clase interna utilizada para almacenar tanto el valor (ID) como el texto (nombre o descripción) de los elementos del ComboBox, permitiendo acceder fácilmente al identificador del elemento seleccionado.

Resumen

El formulario frmBodegasTransportes es una herramienta clave en la gestión de operaciones logísticas, facilitando la asignación de pilotos a pedidos, el seguimiento de envíos, y la creación de órdenes de compra. Su interfaz permite a los usuarios realizar estas tareas de manera rápida y eficiente, asegurando que los datos logísticos se gestionen correctamente en la base de datos.

frmClientesPilotos - Anika

El formulario frmClientesPilotos en la aplicación AdminRepartoApp está diseñado para gestionar la información de clientes y pilotos, permitiendo a los usuarios realizar operaciones como agregar, actualizar y eliminar registros en la base de datos. A continuación, se presenta una descripción detallada de su funcionamiento:

Descripción General

El formulario frmClientesPilotos proporciona una interfaz para la administración de datos personales tanto de clientes como de pilotos. Incluye funcionalidades para buscar, visualizar, añadir, actualizar y eliminar clientes y pilotos de la base de datos.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmClientesPilotos:**
        - Inicializa el formulario y desactiva los campos de entrada y botones de acción para evitar modificaciones accidentales antes de que se seleccione un cliente o piloto.
2. **Carga de Datos en ComboBox:**
    - **CargarComboboxClientes:** Llena el ComboBox (cmbBuscarCliente) con los correos electrónicos de todos los clientes registrados en la base de datos.
    - **CargarComboboxPilotos:** Llena el ComboBox (cmbBuscarPiloto) con los correos electrónicos de todos los pilotos registrados en la base de datos.
3. **Interacción del Usuario:**
    - **Búsqueda y Selección:**
        - **cmbBuscarCliente_SelectedIndexChanged:** Carga la información del cliente seleccionado en los campos correspondientes para su visualización o edición.
        - **cmbBuscarPiloto_SelectedIndexChanged_1:** Similar a la anterior, pero para pilotos.
    - **Agregar Nuevos Registros:**
        - **btnAgregarCliente_Click:** Habilita los campos de cliente para la introducción de un nuevo cliente.
        - **btnAgregarPiloto_Click:** Habilita los campos de piloto para la introducción de un nuevo piloto.
    - **Guardar Registros:**
        - **btnGuardarCliente_Click:** Valida y guarda la información de un nuevo cliente en la base de datos.

        - **btnGuardarPiloto_Click_1:** Valida y guarda la información de un nuevo piloto en la base de datos.
    - **Actualización de Registros:**
        - **btnActualizarCliente_Click:** Permite actualizar la información de un cliente existente en la base de datos.
        - **btnActualizarPiloto_Click_1:** Permite actualizar la información de un piloto existente.
    - **Eliminación de Registros:**
        - **btnEliminarCliente_Click:** Elimina un cliente seleccionado de la base de datos después de confirmar la acción.
        - **btnEliminarPiloto_Click_1:** Elimina un piloto seleccionado de la base de datos.
4. **Visualización de Datos:**
    - **btnVerClientes_Click:** Carga y muestra en un DataGridView (dgvClientes) todos los clientes registrados en la base de datos.
    - **btnVerPilotos_Click_1:** Similar al anterior, pero muestra todos los pilotos.
5. **Validación de Datos:**
    - **ValidarCamposCliente y ValidarCamposPiloto:** Asegura que todos los campos requeridos estén completos y validos antes de permitir la inserción o actualización de datos. Incluye verificaciones para que las contraseñas coincidan y que los valores numéricos sean válidos.
6. **Manejo de la Base de Datos:**
    - Utiliza conexiones a MySQL para ejecutar consultas que obtienen, insertan, actualizan y eliminan registros en la base de datos.
    - Maneja las transacciones para garantizar la integridad de los datos durante las operaciones críticas.
7. **Manejo de Errores:**
    - Emplea bloques try-catch para manejar excepciones que puedan ocurrir durante las operaciones de base de datos, proporcionando mensajes informativos al usuario.
8. **Habilitación y Limpieza de Campos:**
    - **HabilitarCamposCliente y HabilitarCamposPiloto:** Controlan la habilitación de los campos de entrada para clientes y pilotos, respectivamente.
    - **LimpiarCamposCliente y LimpiarCamposPiloto:** Limpian los campos de entrada después de completar una operación, preparándolos para nuevas entradas.

Resumen

El formulario frmClientesPilotos es una herramienta crucial para la administración de los datos de clientes y pilotos en el sistema. Proporciona una interfaz amigable y eficiente para gestionar estas entidades, permitiendo operaciones como búsqueda, adición, modificación y eliminación, todo mientras asegura la integridad y seguridad de los datos mediante validaciones y manejo de errores robusto.

frmProducto - Maty

El formulario frmProducto de la aplicación AdminRepartoApp está diseñado para permitir a los usuarios agregar nuevos productos a la base de datos. Proporciona funcionalidades para introducir los detalles de un producto, cargar una imagen asociada y validar los datos antes de almacenarlos. Aquí tienes una descripción detallada de su funcionamiento:

Descripción General

El formulario frmProducto es una interfaz para la creación de productos en la base de datos, asegurando que los datos sean válidos y completos antes de ser guardados. Se centra en la gestión de la información esencial del producto, incluyendo su imagen y características clave.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmProducto:**
        - Inicializa los componentes gráficos y ajusta el PictureBox (pbImagenProducto) para que las imágenes se escalen correctamente cuando son cargadas.
2. **Carga de Datos Iniciales:**
    - **frmProducto_Load:** Método preparado para cargar datos iniciales o configuraciones al abrir el formulario, aunque en este caso no se está utilizando.
3. **Carga de Imágenes:**
    - **btnCargarImagen_Click:**
        - Abre un diálogo de archivo para permitir al usuario seleccionar una imagen de producto desde su computadora.
        - Actualiza el PictureBox con la imagen seleccionada y guarda la ruta de la imagen en un TextBox oculto para referencia futura.
4. **Guardado de Productos:**
    - **btnGuardar_Click:**
        - Primero, valida que todos los campos requeridos sean correctos y estén completos.
        - Conecta a la base de datos MySQL y prepara la consulta SQL para insertar un nuevo registro de producto.
        - Convierte la imagen a un arreglo de bytes para almacenarla en la base de datos junto con los otros datos del producto.
        - Si la operación es exitosa, muestra un mensaje de confirmación y cierra el formulario.
5. **Validación de Datos:**
    - **ValidarCampos:**

        - Asegura que todos los campos necesarios estén llenos antes de permitir la inserción.
        - **Validaciones Numéricas:**
            - Verifica que el precio y el IVA sean valores decimales válidos y positivos.
            - Confirma que el stock sea un número entero positivo.
        - **Validaciones de Texto:**
            - Usa expresiones regulares para asegurarse de que el nombre del producto, los comentarios y los contactos de marca solo contengan caracteres permitidos, evitando errores de formato o de entrada no intencionada.

6. **Manejo de Errores:**
    - Uso de bloques try-catch para manejar errores potenciales durante las operaciones de base de datos o durante la validación de datos, asegurando que el usuario sea notificado de cualquier problema.

Detalles de Validación de Datos

El formulario implementa varias validaciones específicas para garantizar que los datos ingresados sean correctos y cumplan con los requisitos del sistema:

- **Campos de Texto:** Asegura que no estén vacíos y que contengan solo caracteres válidos según las necesidades de cada campo.
- **Campos Numéricos:** Verifica que los valores sean numéricos donde sea necesario (precio, IVA, stock) y asegura que sean positivos.
- **Campos de Comentarios y Contactos:** Permite caracteres especiales como tildes, ñ, y símbolos comunes en comentarios y contactos, utilizando expresiones regulares para definir estas reglas.

Resumen

El formulario frmProducto facilita la adición de nuevos productos a la base de datos de manera controlada y segura. Su enfoque en la validación exhaustiva y en el manejo eficiente de imágenes asegura que los productos se registren con precisión, mejorando la consistencia y la calidad de los datos en la base de datos del sistema.

frmGestionPersonal - Emanuel

El formulario frmGestionPersonal en la aplicación AdminRepartoApp está diseñado para gestionar la información de empleados, permitiendo realizar operaciones de búsqueda, actualización y eliminación de registros en la base de datos. Además, permite asignar roles y permisos a los empleados. A continuación, se presenta una descripción detallada de su funcionamiento:

Descripción General

El formulario frmGestionPersonal proporciona una interfaz para la administración de datos personales de los empleados y la configuración de sus permisos y roles dentro del sistema. Permite a los usuarios realizar operaciones como búsqueda, actualización y eliminación de registros, así como la asignación de roles y permisos.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmGestionPersonal:**
        - Inicializa el formulario y desactiva los campos de entrada y botones de acción para evitar modificaciones accidentales antes de seleccionar un empleado.
2. **Carga de Datos Iniciales:**
    - **frmGestionPersonal_Load:**
        - Establece la conexión a la base de datos y carga los correos electrónicos de los empleados, los tipos de personal, roles, estados y géneros disponibles en sus respectivos ComboBox.
3. **Interacción del Usuario:**
    - **Búsqueda y Selección:**
        - **cmbBuscarPersonal_SelectedIndexChanged_1:**
            - Carga la información del empleado seleccionado en los campos correspondientes para su visualización o edición.
        - **cmbTipoPersonal_SelectedIndexChanged:**
            - Filtra y carga en la vista de datos los empleados según el tipo seleccionado, actualizando el ComboBox de búsqueda con los correos de los empleados filtrados.
    - **Actualización de Registros:**
        - **btnActualizar_Click_1:**
            - Valida y actualiza la información del empleado seleccionado en la base de datos, incluyendo sus permisos y roles.
    - **Eliminación de Registros:**
        - **btnEliminar_Click_1:**

        - Elimina un empleado seleccionado de la base de datos después de confirmar la acción con el usuario.

4. **Manejo de la Base de Datos:**
    - **Conexión y Consultas:**
        - Utiliza conexiones a MySQL para ejecutar consultas que obtienen, insertan, actualizan y eliminan registros en las tablas Empleado y Datos_Personales.
        - Incluye transacciones para asegurar la integridad de los datos durante las operaciones críticas.
5. **Manejo de Logs:**
    - **RegistrarLog:**
        - Inserta un registro en la tabla LogActividades cada vez que se actualiza o elimina un usuario, manteniendo un seguimiento de las acciones realizadas por los empleados.
6. **Validación de Datos:**
    - **ValidarCampos:**
        - Verifica que todos los campos requeridos estén completos y sean válidos antes de permitir la actualización.
        - **Validaciones de Formato:**
            - Asegura que el correo electrónico y el teléfono tengan un formato válido utilizando expresiones regulares.
7. **Manejo de Errores:**
    - Emplea bloques try-catch para manejar excepciones durante las operaciones de base de datos, proporcionando mensajes informativos al usuario en caso de errores.
8. **Habilitación y Limpieza de Campos:**
    - **HabilitarCampos:** Controla la habilitación de los campos de entrada para los empleados.
    - **LimpiarCampos:** Limpia los campos de entrada después de completar una operación, preparándolos para nuevas entradas.

Resumen

El formulario frmGestionPersonal es una herramienta crucial para la administración de empleados dentro del sistema, permitiendo gestionar de manera eficiente los datos personales, roles y permisos de los empleados. Facilita la supervisión y modificación de la estructura organizativa, asegurando que los empleados tengan los permisos adecuados para realizar sus tareas. La integración de un sistema de logs permite mantener un registro detallado de las acciones realizadas, mejorando la transparencia y seguridad del sistema.

frmReportesBodegasTransportes - Pablo

El formulario frmReportesBodegasTransportes en la aplicación AdminRepartoApp está diseñado para permitir la generación de reportes sobre los pedidos y sus estados de envío desde la base de datos. Utiliza la biblioteca ClosedXML para exportar los datos a archivos de Excel. A continuación, se presenta una descripción detallada de su funcionamiento:

Descripción General

El formulario frmReportesBodegasTransportes ofrece una interfaz para seleccionar pedidos y generar reportes detallados en formato Excel. Estos reportes incluyen información relevante sobre los pedidos, clientes y estados de envío.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmReportesBodegasTransportes:**
        - Inicializa los componentes del formulario y carga los valores iniciales en los ComboBox para tipos de pedidos, estados y estados de envío.
2. **Carga de Datos Iniciales:**
    - **CargarComboboxValores:**
        - Rellena los ComboBox con opciones predeterminadas para tipos de pedidos y estados de envío.
    - **CargarComboboxPedidos:**
        - Conecta a la base de datos y carga los números de factura de los pedidos disponibles, evitando duplicados mediante DISTINCT.
3. **Interacción del Usuario:**
    - **Selección de Pedido:**
        - **cmbBuscarPedido_SelectedIndexChanged:**
            - Al seleccionar un pedido en el ComboBox, carga los detalles del pedido, cliente y envío asociados al número de factura seleccionado.
    - **Generación de Reportes:**
        - **btnGenerarReporte_Click_1:**
            - Genera un reporte en formato Excel del pedido seleccionado. Utiliza ClosedXML para crear y guardar el archivo Excel.
4. **Generación de Reportes:**
    - **Exportación a Excel:**
        - Utiliza la biblioteca ClosedXML para crear una hoja de cálculo en Excel con los datos del pedido seleccionado. Guarda el archivo mediante un diálogo de

guardar archivo, permitiendo al usuario elegir la ubicación de almacenamiento.

5. **Manejo de Errores y Logs:**
    - **Registro de Errores:**
        - **LogError:**
            - Inserta un registro en la tabla logactividades para cada error ocurrido durante las operaciones de carga de datos y generación de reportes, con detalles sobre la función y el mensaje de error.
    - **Registro de Acciones:**
        - **LogAction:**
            - Inserta un registro en la tabla logactividades cuando se genera un reporte, registrando la acción y detalles del pedido.
6. **Manejo de la Base de Datos:**
    - **Conexión y Consultas:**
        - Utiliza conexiones a MySQL para ejecutar consultas que obtienen los datos necesarios para los reportes desde las tablas Pedido, Datos_Personales y Envio.
        - Usa comandos SQL para asegurar que los datos sean precisos y estén actualizados.

Resumen

El formulario frmReportesBodegasTransportes es una herramienta efectiva para la generación de reportes detallados sobre los pedidos y sus envíos, permitiendo a los usuarios exportar fácilmente la información a archivos Excel. El uso de ClosedXML para la creación de documentos Excel proporciona una manera simple y eficiente de gestionar la generación de reportes. Además, el formulario incluye un robusto manejo de errores y logs para asegurar la integridad y trazabilidad de las operaciones.

frmReportesPersonal - Jorge

El formulario frmReportesPersonal en la aplicación AdminRepartoApp permite la visualización, búsqueda y exportación de los datos de los empleados de la empresa. Utiliza la biblioteca ClosedXML para exportar la información a un archivo Excel. A continuación, se detalla su funcionamiento:

Descripción General

El formulario está diseñado para mostrar información detallada de los empleados, incluyendo sus datos personales y permisos dentro del sistema. Además, permite exportar esta información a Excel para su análisis y archivado.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmReportesPersonal:**
        - Inicializa los componentes del formulario y desactiva todos los campos de entrada al inicio.
2. **Carga de Datos Iniciales:**
    - **frmReportesPersonal_Load:**
        - Se conecta a la base de datos al cargar el formulario y llama al método CargarComboboxEmpleados para cargar los correos de los empleados en el ComboBox.
3. **Interacción del Usuario:**
    - **Selección de Empleado:**
        - **cmbBuscarEmpleado_SelectedIndexChanged:**
            - Cuando se selecciona un empleado en el ComboBox, carga sus datos desde la base de datos y los muestra en los campos de texto del formulario.
    - **Visualización de Empleados:**
        - **btnVerEmpleados_Click:**
            - Llama al método CargarDatosEmpleados para mostrar la lista completa de empleados en un DataGridView.
4. **Exportación de Datos:**
    - **btnExportarExcel_Click:**
        - Permite al usuario exportar los datos de los empleados mostrados en el DataGridView a un archivo Excel usando ClosedXML.
5. **Manejo de la Base de Datos:**
    - **Conexiones y Consultas:**

- Utiliza conexiones a MySQL para ejecutar consultas que obtienen la información necesaria desde las tablas Datos_Personales y Empleado.
- Asegura la precisión de los datos mostrados en el formulario.

6. **Manejo de Errores:**
    - Implementa mensajes de error para informar al usuario sobre problemas de conexión o fallos durante el proceso de exportación.

Funciones Detalladas

- **CargarComboboxEmpleados:**
    - Rellena el ComboBox con los correos electrónicos de los empleados desde la base de datos, mostrando un mensaje si no hay empleados.
- **CargarDatosEmpleados:**
    - Ejecuta una consulta SQL para obtener los datos de todos los empleados y los muestra en un DataGridView.
- **HabilitarCampos:**
    - Activa o desactiva los campos de entrada del formulario para permitir la edición de datos de los empleados seleccionados.
- **LimpiarCampos:**
    - Limpia los campos de entrada después de mostrar los datos de un empleado o después de exportar los datos.
- **btnExportarExcel_Click:**
    - Utiliza un diálogo de guardar archivo para permitir al usuario elegir la ubicación del archivo Excel exportado.

Resumen

El formulario frmReportesPersonal proporciona una manera eficiente de gestionar y visualizar los datos de los empleados, facilitando la exportación de esta información a Excel para fines de análisis y reportes. Su integración con ClosedXML asegura una exportación sencilla y confiable de los datos en formato Excel. Además, la implementación de manejo de errores mejora la experiencia del usuario al interactuar con la aplicación.

FrmLogs - Jorge

El formulario frmLogs en la aplicación AdminRepartoApp está diseñado para mostrar, filtrar y exportar los registros de actividad (logs) de los empleados. Aquí te explico cómo está estructurado y cómo funciona el código:

Descripción General

Este formulario proporciona una interfaz para visualizar los logs de actividades de los empleados almacenados en la base de datos. Permite filtrar los registros por empleado específico y exportar los logs a un archivo CSV para análisis o archivado.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmLogs:**
        - Inicializa los componentes del formulario.
2. **Carga de Datos Iniciales:**
    - **frmLogs_Load:**
        - Llama a los métodos CargarLogs y CargarEmpleados para cargar los registros y empleados desde la base de datos al abrir el formulario.
3. **Carga de Logs y Empleados:**
    - **CargarLogs:**
        - Conecta a la base de datos y obtiene todos los registros de la tabla LogActividades, mostrándolos en un DataGridView.
    - **CargarEmpleados:**
        - Obtiene y carga los IDs de empleados que tienen logs registrados en un ComboBox, permitiendo seleccionar entre todos los empleados o un empleado específico para el filtrado.
4. **Filtrado de Logs:**
    - **btnAplicarFiltro_Click_1:**
        - Filtra los logs basándose en el empleado seleccionado del ComboBox. Llama a FiltrarLogsPorEmpleado para mostrar únicamente los registros del empleado seleccionado.
    - **FiltrarLogsPorEmpleado:**
        - Ejecuta una consulta SQL para obtener los registros de actividad de un empleado específico y actualiza el DataGridView con estos registros.
5. **Gestión de Filtros:**
    - **btnLimpiarFiltro_Click_1:**

        - Restablece el filtro al estado inicial, mostrando todos los logs disponibles.

6. **Exportación de Logs:**
    - **btnExportarLogs_Click_1:**
        - Permite exportar los registros mostrados en el DataGridView a un archivo CSV, proporcionando un diálogo para seleccionar el destino del archivo.

Manejo de Errores

- Se implementan mensajes de error para informar al usuario de problemas con la conexión a la base de datos o durante la exportación de los logs.

frmRegister - Maty

El formulario frmRegister en la aplicación AdminRepartoApp se utiliza para registrar nuevos empleados en el sistema. Aquí tienes un resumen de su funcionalidad y estructura:

Descripción General

El formulario permite a los usuarios registrar nuevos empleados, asegurándose de que los datos ingresados son válidos y que no hay duplicados en la base de datos. Los registros incluyen información personal y detalles del rol de empleado.

Funcionalidades Principales

1. **Inicialización del Formulario:**
    - **Constructor frmRegister:**
        - Inicializa los componentes del formulario.
2. **Registro de Nuevo Empleado:**
    - **btnRegisterCliente_Click:**
        - Valida los campos de entrada utilizando el método ValidarCampos.
        - Verifica que el correo electrónico y el DPI no existan ya en la base de datos.
        - Inserta los datos del nuevo empleado en las tablas Empleado y Datos_Personales.
        - Registra la acción en la tabla de logs LogActividades.
3. **Validación de Campos:**
    - **ValidarCampos:**
        - Asegura que todos los campos requeridos están completos.
        - Comprueba que los nombres y apellidos contengan solo letras.
        - Valida el formato del correo electrónico.
        - Asegura que la dirección personal tiene una longitud mínima aceptable.

Manejo de Errores

- El código implementa mensajes de error que informan al usuario sobre problemas durante el proceso de registro, como errores de conexión a la base de datos o entrada de datos no válida.

Proceso de Registro

- **Verificación de Unicidad:**
    - El código verifica que el correo y el DPI no existan ya en la base de datos antes de permitir el registro.
- **Inserción de Datos:**

- Los datos del nuevo empleado se insertan en las tablas Empleado y Datos_Personales.

- **Registro en Logs:**
  - La acción de registro se documenta en la tabla LogActividades para auditoría y seguimiento.

Resumen

El formulario frmRegister asegura que los empleados se registran de manera correcta y segura, validando los datos de entrada y previniendo duplicados en la base de datos. Proporciona un sistema robusto para el manejo de nuevas entradas de empleados, manteniendo la integridad de los datos y registrando las acciones para el seguimiento administrativo. Si necesitas más detalles sobre alguna funcionalidad específica o ajuste, estaré encantado de ayudarte.

CLIENTESREPARTOAPP

Form1(Login) - Kevin

El formulario frmLogin es la puerta de entrada a la aplicación de reparto para clientes. Su principal función es autenticar a los usuarios clientes antes de permitirles el acceso a la aplicación, y maneja el proceso de registro y cierre de sesión.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos para el funcionamiento de la aplicación.
- **System.Linq**: Ofrece funcionalidad para consultas LINQ, que se utilizan para manipular colecciones de datos.
- **System.Net**: Permite el acceso a información de red, utilizada aquí para obtener la dirección IP del usuario.
- **System.Windows.Forms**: Ofrece clases para la creación de aplicaciones de Windows Forms, facilitando la creación de la interfaz de usuario.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL desde .NET, esencial para la conexión y ejecución de consultas en la base de datos.

Estructura de Clases y Métodos

Clase frmLogin

- **Constructor frmLogin()**: Inicializa el formulario de inicio de sesión. Llama al método InitializeComponent() que configura los controles del formulario.

Métodos Principales

1. **btnIniciarSesion_Click**:
    - **Propósito**: Gestiona el proceso de inicio de sesión.
    - **Funcionamiento**:
        - Establece una conexión con la base de datos MySQL.
        - Ejecuta una consulta para verificar que el correo electrónico y la contraseña proporcionados correspondan a un cliente registrado.
        - Si la autenticación es exitosa, redirige al usuario al formulario principal de la aplicación (frmMenu), pasando su ID y correo.
        - Si la autenticación falla, muestra un mensaje de error y registra el intento fallido en los registros de actividad.
2. **btnRegistro_Click**:
    - **Propósito**: Abre el formulario de registro (frmRegistro).
    - **Funcionamiento**:
        - Verifica si el formulario de registro ya está abierto.

        - Si no está abierto, instancia y muestra un nuevo formulario de registro.

3. **btnExit_Click**:
    - **Propósito**: Cierra la aplicación.
    - **Funcionamiento**: Llama a Application.Exit() para terminar la aplicación.
4. **RegistrarLogActividad**:
    - **Propósito**: Registra acciones y eventos en la base de datos para auditoría y seguimiento.
    - **Funcionamiento**:
        - Conecta a la base de datos y registra la acción proporcionada junto con detalles como el nombre del equipo y la dirección IP del cliente.
        - Este registro se utiliza principalmente para auditar intentos fallidos de inicio de sesión y errores.
5. **ObtenerIPv4**:
    - **Propósito**: Obtiene la dirección IPv4 del equipo del usuario.
    - **Funcionamiento**:
        - Itera sobre las direcciones IP del host hasta encontrar una dirección IPv4, que luego devuelve como cadena.

Detalles de Funcionalidad

- **Autenticación**: El formulario utiliza un enfoque básico para autenticar usuarios mediante la comparación de correos electrónicos y contraseñas sin cifrar almacenados en la base de datos.
- **Registro de Actividades**: Cada acción importante (especialmente fallos en el inicio de sesión y errores) se registra en la base de datos, proporcionando un rastro de auditoría valioso.
- **Navegación de Formularios**: Al inicio de sesión exitoso, el formulario principal se oculta mientras se muestra el formulario de menú, mejorando la experiencia de usuario al mantener la navegación dentro de la aplicación sin interrupciones.
- **Gestión de Formularios**: El formulario se asegura de que las instancias duplicadas del formulario de registro no se creen innecesariamente, optimizando el uso de memoria y recursos.

frmRegistro - Pablo

El formulario frmRegistro permite a los nuevos usuarios registrarse en la aplicación de reparto para clientes. Su funcionalidad incluye la validación de campos de entrada, la inserción de nuevos registros en la base de datos, y la navegación de regreso al formulario de inicio de sesión tras un registro exitoso.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos para el funcionamiento de la aplicación.
- **System.Data**: Ofrece clases para la manipulación de datos, particularmente útil para las operaciones con bases de datos.
- **System.Linq**: Permite consultas y operaciones sobre colecciones de datos.
- **System.Text.RegularExpressions**: Proporciona funcionalidad para trabajar con expresiones regulares, utilizadas para validar entradas de texto.
- **System.Windows.Forms**: Ofrece clases para la creación de aplicaciones de Windows Forms, que facilitan la creación de interfaces de usuario.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL desde .NET.
- **System.Net**: Se utiliza para obtener información de red, como la dirección IP del usuario.

Estructura de Clases y Métodos

Clase frmRegistro

- **Constructor frmRegistro()**: Inicializa el formulario de registro. Llama al método InitializeComponent() que configura los controles del formulario.

Métodos Principales

1. **btnRegistro_Click**:
    - **Propósito**: Gestiona el proceso de registro de un nuevo cliente.
    - **Funcionamiento**:
        - Valida los campos de entrada para asegurar que están completos y son correctos.
        - Verifica que no exista ya un usuario con el mismo correo o DPI en la base de datos.
        - Comprueba que las contraseñas ingresadas coinciden.
        - Inserta un nuevo registro en la tabla cliente y obtiene el ID_Cliente.
        - Inserta los datos personales del nuevo cliente en la tabla datos_personales usando el ID_Cliente recién obtenido.
        - Muestra un mensaje de confirmación de registro exitoso y registra la actividad en el sistema de logs.

        - Cierra el formulario de registro y abre o enfoca el formulario de inicio de sesión.

2. **ValidarCampos**:
    - **Propósito**: Valida que todos los campos de entrada del formulario estén correctamente completados.
    - **Funcionamiento**:
        - Verifica que no haya campos vacíos.
        - Usa expresiones regulares para validar el formato del correo electrónico, DPI y teléfono.
        - Retorna true si todos los campos son válidos, de lo contrario muestra un mensaje de error y retorna false.
3. **lnklblRegresar_LinkClicked**:
    - **Propósito**: Permite al usuario regresar al formulario de inicio de sesión.
    - **Funcionamiento**: Cierra el formulario de registro y abre o enfoca el formulario de inicio de sesión.
4. **RegistrarLogActividad**:
    - **Propósito**: Registra actividades y eventos en la base de datos para auditoría.
    - **Funcionamiento**:
        - Conecta a la base de datos y registra la acción proporcionada, junto con detalles como el nombre del equipo y la dirección IP del cliente.
5. **ObtenerIPv4**:
    - **Propósito**: Obtiene la dirección IPv4 del equipo del usuario.
    - **Funcionamiento**:
        - Itera sobre las direcciones IP del host hasta encontrar una dirección IPv4, que luego devuelve como cadena.

Detalles de Funcionalidad

- **Registro de Usuarios**: El formulario permite registrar nuevos usuarios asegurándose de que no existan duplicados por correo o DPI.
- **Validación de Datos**: Los campos de entrada son validados usando expresiones regulares para asegurar que cumplen con los formatos esperados.
- **Gestión de Navegación**: Después del registro, el usuario es redirigido al formulario de inicio de sesión para que pueda iniciar sesión con las nuevas credenciales.
- **Registro de Actividades**: Las actividades importantes, como registros exitosos y errores, se registran en una tabla de logs en la base de datos, proporcionando un historial de eventos.

frmMenu - Anika

El formulario frmMenu actúa como el panel de control principal al que acceden los usuarios después de iniciar sesión. Desde este menú, los usuarios pueden realizar acciones clave como gestionar sus pedidos y editar su perfil. También ofrece la funcionalidad de cerrar sesión y volver al formulario de inicio de sesión.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos que incluyen la manipulación de fechas y horas.
- **System.Linq**: Facilita la búsqueda y manipulación de colecciones de datos, utilizada aquí para buscar formularios abiertos.
- **System.Windows.Forms**: Ofrece clases para la creación y gestión de interfaces gráficas de usuario (GUI) en Windows Forms.

Estructura de Clases y Métodos

Clase frmMenu

- **Atributos**:
    - idPersona: Almacena el identificador único de la persona que ha iniciado sesión.
    - correoUsuario: Guarda el correo electrónico del usuario que ha iniciado sesión.
- **Constructor frmMenu(int idPersona, string correo)**:
    - Inicializa una nueva instancia del formulario de menú principal.
    - Llama al método InicializarMenu para configurar los detalles del usuario en la interfaz.

Métodos Principales

1. **InicializarMenu(int idPersona, string correo)**:
    - **Propósito**: Configura los detalles del usuario en la interfaz del menú.
    - **Funcionamiento**:
        - Valida que el correo del usuario no sea nulo o vacío.
        - Asigna los valores de idPersona y correoUsuario.
        - Actualiza las etiquetas del formulario con el correo del usuario y la fecha y hora actuales.
2. **frmMenu_Load**:
    - **Propósito**: Maneja eventos de carga del formulario.
    - **Funcionamiento**: Se utiliza para cargar recursos necesarios cuando el formulario se inicia.
3. **btnLogOut_Click**:

- **Propósito**: Cierra la sesión del usuario y regresa al formulario de inicio de sesión.
- **Funcionamiento**:
  - Oculta el formulario actual.
  - Abre una nueva instancia del formulario de inicio de sesión, o enfoca una existente.

4. **btnRealizarPedidos_Click**:
   - **Propósito**: Permite al usuario acceder al formulario para realizar pedidos.
   - **Funcionamiento**:
     - Crea y muestra una nueva instancia del formulario frmRealizarPedidos, pasando el idPersona para mantener la sesión del usuario.
5. **btnVerPedidos_Click**:
   - **Propósito**: Abre el formulario para que el usuario vea sus pedidos actuales.
   - **Funcionamiento**:
     - Crea y muestra una nueva instancia del formulario frmVerPedidos, también pasando el idPersona.
6. **btnPerfil_Click**:
   - **Propósito**: Permite al usuario acceder a su perfil personal.
   - **Funcionamiento**:
     - Crea y muestra una nueva instancia del formulario frmPerfil, pasando el idPersona para mostrar y permitir la edición de los detalles del perfil del usuario.

Detalles de Funcionalidad

- **Interfaz de Usuario**: El formulario actúa como el panel de navegación central de la aplicación, facilitando el acceso a otras funcionalidades esenciales como la gestión de pedidos y la edición del perfil del usuario.
- **Gestión de Sesión**: El menú maneja la información del usuario actualmente logueado (ID y correo), permitiendo que otros formularios accedan a esta información para personalizar la experiencia del usuario.
- **Navegación**: Implementa una navegación fluida entre el menú principal y otros formularios dentro de la aplicación, asegurando que las sesiones de usuario se mantengan mientras se cambia de una funcionalidad a otra.
- **Manejo de Errores**: Cada acción de botón incluye un manejo de excepciones básico, que muestra mensajes de error en caso de que ocurra un problema durante la ejecución de la operación correspondiente.

frmPerfil - Jorge

El formulario frmPerfil permite a los usuarios visualizar y actualizar su información personal almacenada en la base de datos. Este formulario ofrece una interfaz donde los usuarios pueden ver sus datos personales y, si lo desean, editarlos tras autenticarse con una contraseña válida.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos que incluyen excepciones y operaciones matemáticas.
- **System.Windows.Forms**: Ofrece clases para la creación y gestión de interfaces gráficas de usuario (GUI) en Windows Forms, permitiendo la interacción del usuario con la aplicación.
- **MySql.Data.MySqlClient**: Proporciona las clases necesarias para conectar y manipular bases de datos MySQL desde una aplicación C#.

Estructura de Clases y Métodos

Clase frmPerfil

- **Atributos**:
    - idPersona: Almacena el identificador único de la persona que ha iniciado sesión.
    - connectionString: Una cadena que contiene la información necesaria para conectarse a la base de datos MySQL.
    - isEditing: Un indicador booleano para determinar si el formulario está en modo de edición.
- **Constructor frmPerfil(int idPersona)**:
    - Inicializa una nueva instancia del formulario de perfil.
    - Carga los datos del cliente desde la base de datos y desactiva la edición de campos.
    - Configura los campos de contraseña para que muestren caracteres ocultos.

Métodos Principales

1. **CargarDatosCliente()**:
    - **Propósito**: Recupera y muestra la información personal del usuario desde la base de datos.
    - **Funcionamiento**:
        - Establece una conexión a la base de datos.
        - Ejecuta una consulta SQL para obtener los datos personales del usuario con el ID_Persona dado.
        - Asigna los datos recuperados a los campos de texto correspondientes en el formulario.

2. **HabilitarCampos(bool habilitar)**:
    - **Propósito**: Controla si los campos de texto son editables.
    - **Funcionamiento**:
        - Basado en el valor del parámetro habilitar, activa o desactiva la edición de los campos.
        - También controla la visibilidad de los campos de confirmación de contraseña.
3. **frmPerfil_Load**:
    - **Propósito**: Maneja eventos de carga del formulario.
    - **Funcionamiento**: Se utiliza para ejecutar cualquier acción necesaria cuando el formulario se carga.
4. **btnActualizarDatos_Click**:
    - **Propósito**: Permite al usuario actualizar su información personal.
    - **Funcionamiento**:
        - Si no está en modo edición, activa la edición de campos y cambia el texto del botón.
        - Si está en modo edición, verifica que las contraseñas coincidan y solicita confirmación para guardar los cambios.
        - Actualiza la base de datos con los nuevos datos del usuario si se confirma la actualización.
        - Maneja el estado del modo edición y la visibilidad de los campos de confirmación de contraseña.

Detalles de Funcionalidad

- **Visualización y Edición de Datos**: Permite a los usuarios ver su información personal y realizar cambios cuando están autenticados correctamente.
- **Seguridad**: Maneja las contraseñas de usuario con cuidado, ocultando los caracteres y asegurando que la confirmación de la contraseña coincida antes de realizar cambios.
- **Interfaz de Usuario**: Proporciona un mecanismo fácil de usar para alternar entre los modos de visualización y edición.
- **Manejo de Errores**: Utiliza bloques try-catch para manejar y mostrar errores que puedan ocurrir durante las operaciones de base de datos.

Este formulario es esencial para permitir que los usuarios gestionen su información personal dentro de la aplicación, asegurando tanto la accesibilidad como la seguridad de los datos.

frmRealizarPedidos - Emanuel

El formulario frmRealizarPedidos permite a los usuarios seleccionar productos y realizar pedidos. Proporciona una interfaz para elegir un producto, ver sus detalles como el precio y el stock, seleccionar la cantidad deseada y finalmente confirmar el pedido con un método de pago.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos, incluyendo operaciones matemáticas y manejo de excepciones.
- **System.Data**: Ofrece clases para la manipulación de datos, especialmente en entornos desconectados.
- **System.Linq**: Proporciona funcionalidad de consulta para objetos en memoria.
- **System.Drawing**: Contiene clases para el manejo de gráficos y dibujo de objetos.
- **System.Windows.Forms**: Ofrece clases para la creación y gestión de interfaces gráficas de usuario (GUI) en Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona las clases necesarias para conectar y manipular bases de datos MySQL desde una aplicación C#.

Estructura de Clases y Métodos

Clase frmRealizarPedidos

- **Atributos**:
    - connectionString: Almacena la cadena de conexión a la base de datos MySQL.
    - idPersona: Guarda el identificador único de la persona que está realizando el pedido.
- **Constructor frmRealizarPedidos(int idPersona)**:
    - Inicializa una nueva instancia del formulario de pedidos.
    - Configura el modo de visualización de la imagen del producto.
    - Asigna el idPersona recibido para identificar al usuario que realiza el pedido.

Métodos Principales

1. **frmRealizarPedidos_Load(object sender, EventArgs e)**:
    - **Propósito**: Maneja el evento de carga del formulario.
    - **Funcionamiento**: Llama al método CargarProductos() para llenar el comboBox de productos al cargar el formulario.
2. **CargarProductos()**:
    - **Propósito**: Llena el comboBox con los productos disponibles en la base de datos.
    - **Funcionamiento**:

        - Se conecta a la base de datos y recupera los nombres y IDs de los productos.
        - Agrega cada producto como un ComboBoxItem al comboBox.

3. **cmbProductos_SelectedIndexChanged(object sender, EventArgs e)**:
    - **Propósito**: Muestra los detalles del producto seleccionado.
    - **Funcionamiento**:
        - Recupera y muestra el precio, IVA, stock, descripción, y la imagen del producto seleccionado.
4. **btnAumentarCantidad_Click(object sender, EventArgs e)**:
    - **Propósito**: Incrementa la cantidad del producto seleccionada.
    - **Funcionamiento**: Aumenta la cantidad en uno, siempre que no exceda el stock disponible.
5. **btnDisminuirCantidad_Click(object sender, EventArgs e)**:
    - **Propósito**: Disminuye la cantidad del producto seleccionada.
    - **Funcionamiento**: Disminuye la cantidad en uno, siempre que sea mayor que cero.
6. **btnRealizarPedido_Click(object sender, EventArgs e)**:
    - **Propósito**: Procesa el pedido y lo registra en la base de datos.
    - **Funcionamiento**:
        - Valida que se haya seleccionado un producto, cantidad y método de pago.
        - Calcula el total del pedido y lo inserta en las tablas detalle y pedido.
        - Genera un número de factura único y lo verifica.
        - Actualiza el stock del producto en la base de datos.
        - Muestra un mensaje de éxito al usuario.
7. **GenerarHexadecimal(int length)**:
    - **Propósito**: Genera un número hexadecimal aleatorio.
    - **Funcionamiento**: Usa caracteres hexadecimales para crear una cadena aleatoria de la longitud especificada.
8. **ExisteNoFactura(MySqlConnection connection, string noFactura)**:
    - **Propósito**: Verifica si un número de factura ya existe en la base de datos.
    - **Funcionamiento**: Ejecuta una consulta SQL para contar los registros con el número de factura dado.

Clase ComboBoxItem

- **Atributos**:

    - Value: Almacena el valor del elemento (ID del producto).
    - Text: Almacena el texto del elemento (nombre del producto).
- **Constructor ComboBoxItem(string value, string text)**:
    - Inicializa un nuevo objeto ComboBoxItem con los valores proporcionados.
- **Método ToString()**:
    - **Propósito**: Retorna el texto del elemento.
    - **Funcionamiento**: Sobre-escribe el método ToString() para devolver el nombre del producto.

Detalles de Funcionalidad

- **Interacción con la Base de Datos**: Realiza consultas para obtener detalles de productos, registrar pedidos y actualizar el stock.
- **Gestión de Pedidos**: Facilita la creación de pedidos al permitir al usuario seleccionar productos, cantidades y métodos de pago.
- **Validación de Datos**: Asegura que el usuario haya seleccionado opciones válidas antes de permitir que el pedido se procese.
- **Interfaz de Usuario**: Proporciona un mecanismo para navegar y gestionar productos de manera intuitiva para los usuarios.

Este formulario es crucial para permitir a los usuarios realizar compras, reflejando una experiencia de compra en línea dentro de la aplicación.

frmVerPedidos - Maty

El formulario frmVerPedidos permite a los usuarios ver y gestionar sus pedidos realizados. Proporciona funcionalidades para ver los pedidos en diferentes estados, consultar los detalles de cada pedido y factura, así como calificar a los pilotos encargados de las entregas.

Librerías Utilizadas

- **System**: Proporciona clases y métodos básicos, incluyendo operaciones matemáticas y manejo de excepciones.
- **System.Data**: Ofrece clases para la manipulación de datos, especialmente en entornos desconectados.
- **System.Windows.Forms**: Ofrece clases para la creación y gestión de interfaces gráficas de usuario (GUI) en Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona las clases necesarias para conectar y manipular bases de datos MySQL desde una aplicación C#.

Estructura de Clases y Métodos

Clase frmVerPedidos

- **Atributos**:
    - connectionString: Almacena la cadena de conexión a la base de datos MySQL.
    - idPersona: Guarda el identificador único de la persona cuyas órdenes se están visualizando.
- **Constructor frmVerPedidos(int idPersona)**:
    - Inicializa una nueva instancia del formulario para ver pedidos.
    - Asigna el idPersona recibido para identificar al usuario.

Métodos Principales

1. **frmVerPedidos_Load(object sender, EventArgs e)**:
    - **Propósito**: Maneja el evento de carga del formulario.
    - **Funcionamiento**: Llama a los métodos CargarEstadoPedidos() y CargarFacturas() para cargar los estados y facturas al iniciar el formulario.
2. **CargarEstadoPedidos()**:
    - **Propósito**: Llena el comboBox de estados de pedidos con opciones predeterminadas.
    - **Funcionamiento**: Añade opciones como "En Tránsito", "Entregado", "Retrasado", "Cancelado" y "Pendiente".
3. **CargarFacturas()**:
    - **Propósito**: Llena el comboBox de facturas con facturas disponibles para el usuario.

- **Funcionamiento**: Realiza una consulta a la base de datos para obtener números de factura para pedidos con estado "Entregado".

4. **cmbEstadoPedidos_SelectedIndexChanged(object sender, EventArgs e)**:
   - **Propósito**: Muestra pedidos filtrados por el estado seleccionado.
   - **Funcionamiento**: Llama al método CargarPedidosPorEstado() para actualizar la vista de pedidos según el estado elegido.
5. **CargarPedidosPorEstado(string estado)**:
   - **Propósito**: Muestra los pedidos que corresponden a un estado específico.
   - **Funcionamiento**: Realiza una consulta a la base de datos para obtener detalles de pedidos en un estado específico y los muestra en un DataGridView.
6. **cmbFactura_SelectedIndexChanged(object sender, EventArgs e)**:
   - **Propósito**: Muestra los detalles del pedido y del piloto basados en la factura seleccionada.
   - **Funcionamiento**: Llama a CargarDetallesPedido() y CargarDetallesPilotoPorFactura() para obtener y mostrar información detallada.
7. **CargarDetallesPedido(string noFactura)**:
   - **Propósito**: Muestra detalles del pedido basado en el número de factura.
   - **Funcionamiento**: Realiza una consulta a la base de datos y muestra los detalles en un DataGridView.
8. **CargarDetallesPilotoPorFactura(string noFactura)**:
   - **Propósito**: Muestra detalles del piloto encargado de un pedido específico.
   - **Funcionamiento**: Realiza una consulta para obtener información del piloto y actualiza campos de texto con los datos.
9. **btnAumentarCalificacion_Click(object sender, EventArgs e)**:
   - **Propósito**: Incrementa la calificación del piloto.
   - **Funcionamiento**: Aumenta el valor numérico en el campo de calificación, sin exceder un límite máximo.
10. **btnDisminuirCalificacion_Click(object sender, EventArgs e)**:
    - **Propósito**: Disminuye la calificación del piloto.
    - **Funcionamiento**: Reduce el valor numérico en el campo de calificación, sin caer por debajo de un mínimo.
11. **btnCalificarPiloto_Click_1(object sender, EventArgs e)**:
    - **Propósito**: Registra la calificación del piloto para un pedido específico.

- **Funcionamiento**: Valida la selección de factura y calificación, y actualiza la base de datos con la nueva calificación.

Detalles de Funcionalidad

- **Interacción con la Base de Datos**: Realiza múltiples consultas para obtener y actualizar información de pedidos y pilotos.
- **Gestión de Pedidos**: Proporciona una interfaz para que los usuarios filtren y vean detalles de sus pedidos según el estado.
- **Calificación de Pilotos**: Permite a los usuarios calificar la entrega de los pilotos, proporcionando retroalimentación para mejorar el servicio.
- **Interfaz de Usuario**: Facilita la navegación y revisión de pedidos, con opciones para filtrar y ver detalles adicionales.

Este formulario es una parte esencial de la aplicación para permitir a los usuarios gestionar sus pedidos y proporcionar calificaciones a los pilotos.

PILOTOSREPARTOAPP

Form1(Login) - Maty

Descripción General

El formulario frmLogin se utiliza para gestionar el proceso de inicio de sesión de los pilotos en la aplicación PilotosRepartoApp. Permite a los pilotos ingresar sus credenciales para acceder a sus cuentas. Este formulario también se encarga de registrar los intentos de inicio de sesión en una base de datos para mantener un registro de actividad.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Linq**: Ofrece funciones para manipular colecciones de datos.
- **System.Net**: Permite el acceso a datos de red, utilizado aquí para obtener el nombre del host y la dirección IP.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.
- **MySql.Data.MySqlClient**: Permite la interacción con bases de datos MySQL.

Estructura de la Clase

Variables y Constructores

- **frmLogin**: El constructor del formulario que inicializa los componentes de la interfaz y añade manejadores de eventos para las teclas presionadas en los campos de texto, permitiendo el inicio de sesión al presionar Enter.

Métodos Principales

- **IniciarSesion()**:
    - Establece una conexión con la base de datos comercioelectronico para verificar las credenciales del piloto.
    - Si las credenciales son correctas, se abre el formulario frmMenu, pasando el ID de la persona y su correo.
    - Si las credenciales son incorrectas o si la persona no es un piloto, muestra un mensaje de error y registra un intento fallido de inicio de sesión.
- **RegistrarLogActividad(string accion)**:
    - Registra un evento en la base de datos logactividades, incluyendo información sobre la acción, el equipo y la dirección IP.
- **ObtenerIPv4()**:
    - Devuelve la dirección IPv4 del equipo desde donde se ejecuta la aplicación.

Eventos

- **btnIniciarSesion_Click(object sender, EventArgs e)**: Llama a IniciarSesion() cuando se hace clic en el botón de iniciar sesión.
- **TxtFields_KeyPress(object sender, KeyPressEventArgs e)**: Permite que el usuario inicie sesión al presionar la tecla Enter mientras escribe en los campos de texto.
- **btnExit_Click(object sender, EventArgs e)**: Cierra la aplicación cuando se hace clic en el botón de salir.

Funcionalidades

- **Inicio de Sesión**: Permite a los pilotos autenticarse usando su correo y contraseña. En caso de éxito, redirige al menú principal.
- **Registro de Actividad**: Guarda los intentos de inicio de sesión y cualquier error de conexión en la base de datos, para monitoreo y auditoría.
- **Gestión de Errores**: Muestra mensajes de error apropiados cuando ocurren excepciones durante el proceso de inicio de sesión o la conexión a la base de datos.

Este formulario es esencial para el flujo de autenticación de la aplicación y garantiza que solo los usuarios autorizados puedan acceder a las funcionalidades del sistema.

frmMenu - Anika

El formulario frmMenu sirve como la pantalla principal de la aplicación PilotosRepartoApp una vez que el piloto ha iniciado sesión correctamente. Desde este menú, los pilotos pueden acceder a diferentes funcionalidades como ver pedidos, registrar entregas, ver rutas y administrar su perfil.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.

Estructura de la Clase

Campos

- **idPersona**: Un entero que representa el ID de la persona (piloto) que ha iniciado sesión.
- **correoPiloto**: Una cadena que almacena el correo electrónico del piloto.
- **timerFechaHora**: Un objeto Timer que se utiliza para actualizar la etiqueta de fecha y hora en el formulario.

Constructor

- **frmMenu(int idPersona, string correo)**: Constructor que inicializa el formulario, configura el Timer para actualizar la fecha y hora cada segundo y muestra el correo electrónico del piloto en la interfaz.

Métodos Principales

- **TimerFechaHora_Tick(object sender, EventArgs e)**: Método manejador de eventos que se activa cada segundo para actualizar la etiqueta lblFyH con la fecha y hora actual en el formato "HH

dd/MM/yyyy".

Eventos

- **btnLogOut_Click(object sender, EventArgs e)**: Oculta el formulario actual y muestra el formulario de inicio de sesión (frmLogin), permitiendo al piloto cerrar sesión y volver a la pantalla de inicio.
- **btnVerPedidos_Click(object sender, EventArgs e)**: Abre el formulario frmVerPedidos, que permite al piloto ver los pedidos asignados a su correo electrónico.
- **btnRegistroEntregas_Click(object sender, EventArgs e)**: Abre el formulario frmActualizarPedido, donde el piloto puede registrar las entregas realizadas.
- **btnVerRutas_Click(object sender, EventArgs e)**: Abre el formulario frmVerRutas, pasando valores predeterminados para la dirección y el número de factura. Estos valores se actualizarán cuando se cargue el formulario.

- **btnVerPerfil_Click(object sender, EventArgs e)**: Intenta abrir el formulario frmPerfil, que permite al piloto ver y actualizar su perfil personal. Si ocurre algún error al abrir el formulario, se muestra un mensaje de error.

Funcionalidades

- **Actualización Automática de Fecha y Hora**: Utiliza un Timer para actualizar continuamente la visualización de la fecha y hora en el formulario.
- **Navegación a Funcionalidades Clave**: Proporciona botones que permiten a los pilotos acceder rápidamente a las funcionalidades esenciales, como ver pedidos, registrar entregas, y ver su perfil.
- **Gestión de Sesión**: Facilita el cierre de sesión del usuario, redirigiéndolo de nuevo al formulario de inicio de sesión (frmLogin).

Este formulario actúa como el hub central para los pilotos en la aplicación, facilitando el acceso a varias funciones que son esenciales para su trabajo diario en la gestión de entregas y rutas.

frmPerfil - Pablo

El formulario frmPerfil permite a los pilotos ver y actualizar su información personal en el sistema. Los pilotos pueden modificar sus datos personales, incluyendo su dirección, correo electrónico, contraseña y más.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL.

Estructura de la Clase

Campos

- **idPersona**: Un entero que representa el ID de la persona (piloto) cuyo perfil se está visualizando o editando.
- **connectionString**: Una cadena de conexión utilizada para conectar a la base de datos MySQL.
- **isEditing**: Un booleano que indica si el formulario está en modo de edición o visualización.

Constructor

- **frmPerfil(int idPersona)**: Constructor que inicializa el formulario, carga los datos del piloto desde la base de datos, deshabilita la edición de campos y configura los campos de contraseña para ocultar los caracteres.

Métodos Principales

- **CargarDatosPiloto()**: Método que se conecta a la base de datos, ejecuta una consulta para obtener los datos personales del piloto y los muestra en los campos de texto correspondientes.
- **HabilitarCampos(bool habilitar)**: Método que habilita o deshabilita la edición de los campos de texto del formulario según el parámetro habilitar.

Eventos

- **btnActualizarDatos_Click(object sender, EventArgs e)**: Controla el evento de clic del botón de actualización. Alterna entre el modo de edición y confirmación de cambios. Si se confirma, actualiza los datos del piloto en la base de datos.

Funcionalidades

- **Ver Datos del Piloto**: Al cargar el formulario, se muestran los datos actuales del piloto, obtenidos desde la base de datos.

- **Edición de Datos**: Permite a los pilotos editar sus datos personales. Los campos se habilitan cuando el usuario hace clic en "Actualizar Datos".
- **Confirmación de Cambios**: Antes de aplicar los cambios, se solicita confirmación al usuario para asegurarse de que los datos se guarden correctamente.
- **Seguridad de Contraseña**: Las contraseñas se ocultan usando caracteres especiales mientras se visualizan y solo se muestran durante la edición.

Este formulario es fundamental para que los pilotos mantengan actualizada su información personal y asegura que cualquier cambio en sus datos se refleje correctamente en el sistema.

frmVerRutas - Kevin

El formulario frmVerRutas permite a los pilotos ver y gestionar las rutas de entrega asignadas. Los pilotos pueden seleccionar una factura para ver la dirección de entrega correspondiente y utilizar Google Maps para buscar la ubicación.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Diagnostics**: Proporciona clases para interactuar con procesos del sistema operativo, como abrir un navegador.
- **System.Data**: Proporciona clases para trabajar con datos en memoria.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL.

Estructura de la Clase

Campos

- **correoPiloto**: Una cadena que almacena el correo del piloto, utilizada para identificar al piloto en la base de datos.

Constructor

- **frmVerRutas(string correo, string direccion, string noFactura)**: Constructor que inicializa el formulario, carga las facturas asociadas al piloto, selecciona una factura específica y verifica la dirección.

Métodos Principales

- **CargarFacturas()**: Método que se conecta a la base de datos y carga todas las facturas asociadas al piloto que no han sido entregadas. Llena el combo box cmbFiltrar con los números de factura.
- **SeleccionarFactura(string noFactura)**: Método que selecciona una factura específica en el combo box y carga la dirección correspondiente.
- **CargarDireccion(string noFactura)**: Método que carga la dirección de entrega asociada a un número de factura específico y la muestra en el campo de texto txtDireccion.
- **LimpiarCampos()**: Método que restablece el formulario limpiando el combo box de facturas y el campo de texto de la dirección.
- **VerificarDireccion()**: Método que habilita o deshabilita el botón de búsqueda en función del contenido del campo de texto txtDireccion.

Eventos

- **cmbFiltrar_SelectedIndexChanged(object sender, EventArgs e)**: Evento que se desencadena cuando se selecciona una factura diferente en el combo box, cargando la dirección correspondiente.
- **btnLimpiarFiltro_Click(object sender, EventArgs e)**: Evento que se desencadena al hacer clic en el botón para limpiar el filtro, llamando al método LimpiarCampos.
- **TxtDireccion_TextChanged(object sender, EventArgs e)**: Evento que se desencadena cuando el texto de la dirección cambia, verificando si el botón de búsqueda debe estar habilitado.
- **btnBuscar_Click(object sender, EventArgs e)**: Evento que se desencadena al hacer clic en el botón de búsqueda. Codifica la dirección y abre Google Maps en el navegador con la ubicación.

Funcionalidades

- **Selección de Factura**: Los pilotos pueden seleccionar facturas que no han sido entregadas y ver la dirección de entrega.
- **Visualización de Direcciones en Google Maps**: Una vez seleccionada una factura, los pilotos pueden abrir la ubicación en Google Maps directamente desde la aplicación.
- **Interfaz de Usuario Dinámica**: La interfaz permite la interacción fluida con la lista de facturas y asegura que los pilotos puedan ver las rutas de entrega de manera eficiente.

Este formulario es clave para que los pilotos gestionen y visualicen sus rutas de entrega, optimizando así el proceso de distribución y entrega.

frmVerPedidos - Jorge

El formulario frmVerPedidos permite a los pilotos ver los pedidos que tienen asignados. Los pilotos pueden filtrar los pedidos por número de factura, ver detalles del pedido y acceder a la dirección de entrega. Este formulario facilita la gestión de las entregas al permitir a los pilotos consultar y organizar sus pedidos pendientes.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Data**: Proporciona clases para trabajar con datos en memoria.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL.

Estructura de la Clase

Campos

- **correoPiloto**: Una cadena que almacena el correo del piloto. Se utiliza para identificar al piloto en la base de datos y cargar sus pedidos correspondientes.

Constructor

- **frmVerPedidos(string correo)**: Constructor que inicializa el formulario, carga los pedidos asignados al piloto en un DataGridView y llena un ComboBox con los números de factura de los pedidos pendientes de entrega.

Métodos Principales

- **CargarPedidos()**:
    - Se conecta a la base de datos y ejecuta una consulta para obtener los pedidos asignados al piloto.
    - Llena el DataGridView (dgvPedidos) con la información de cada pedido, incluyendo el número de factura, tipo de pedido, fecha de solicitud, estado y dirección personal.
- **CargarFacturas()**:
    - Se conecta a la base de datos y obtiene todos los números de factura de los pedidos del piloto que no están entregados.
    - Llena el ComboBox (cmbFiltrar) con estos números de factura para permitir el filtrado de pedidos.
- **FiltrarPedidosPorFactura()**:
    - Filtra y muestra en el DataGridView solo el pedido que coincide con el número de factura seleccionado en el ComboBox.
    - Utiliza el correo del piloto para asegurar que solo se muestren los pedidos correspondientes al piloto actual.

Eventos

- **btnLimpiarFiltro_Click(object sender, EventArgs e)**:
    - Restablece el ComboBox a su estado original (sin selección) y vuelve a cargar todos los pedidos en el DataGridView.
- **cmbFiltrar_SelectedIndexChanged(object sender, EventArgs e)**:
    - Se activa cuando el usuario selecciona un número de factura en el ComboBox.
    - Llama al método FiltrarPedidosPorFactura() para mostrar solo el pedido correspondiente en el DataGridView.
- **btnVerRuta_Click(object sender, EventArgs e)**:
    - Permite al piloto ver la dirección de entrega de un pedido específico al abrir el formulario frmVerRutas.
    - Se utiliza el número de factura seleccionado para obtener y mostrar la dirección asociada.

Funcionalidades

- **Visualización de Pedidos**: Los pilotos pueden ver todos los pedidos asignados en un formato de tabla fácil de leer, que muestra información relevante sobre cada pedido.
- **Filtrado por Factura**: Permite a los pilotos filtrar los pedidos por número de factura, facilitando la búsqueda y gestión de pedidos específicos.
- **Acceso a Rutas de Entrega**: Los pilotos pueden acceder a una interfaz que muestra la dirección de entrega de un pedido seleccionado, con la opción de ver la ruta en detalle.

Este formulario es esencial para la gestión diaria de los pilotos, proporcionando una vista clara y organizada de los pedidos que deben manejar, lo que ayuda a mejorar la eficiencia en la entrega de productos.

frmActualizarPedido - Emanuel

El formulario frmActualizarPedido permite a los pilotos gestionar el estado de los pedidos que tienen asignados. Los pilotos pueden marcar los pedidos como entregados o retrasados y, en el caso de entregas, registrar una calificación para el cliente. Este formulario es crucial para mantener la información de entrega actualizada y para gestionar las evaluaciones de los clientes.

Librerías Utilizadas

- **System**: Proporciona clases y funciones básicas del sistema.
- **System.Data**: Proporciona clases para trabajar con datos en memoria.
- **System.Windows.Forms**: Proporciona clases para crear aplicaciones de Windows Forms.
- **MySql.Data.MySqlClient**: Proporciona clases para interactuar con bases de datos MySQL.
- **Microsoft.VisualBasic**: Se utiliza para funciones específicas de VB.NET, como el InputBox.

Estructura de la Clase

Campos

- **correoPiloto**: Una cadena que almacena el correo del piloto. Se utiliza para identificar al piloto en la base de datos y cargar los pedidos correspondientes.

Constructor

- **frmActualizarPedido(string correoPiloto)**: Inicializa el formulario, carga las facturas asociadas al piloto en un ComboBox, y deshabilita los botones de acción hasta que se seleccione un pedido.

Métodos Principales

- **CargarFacturas()**:
    - Se conecta a la base de datos y ejecuta una consulta para obtener los números de factura de los pedidos que no están entregados y que están asignados al piloto.
    - Llena el ComboBox (cmbFiltrar) con estos números de factura para permitir la selección de un pedido específico.
- **FiltrarPedido()**:
    - Filtra y muestra en el DataGridView solo el pedido que coincide con el número de factura seleccionado en el ComboBox.
    - Muestra información detallada del pedido, incluyendo tipo de pedido, fecha de solicitud, estado, calificación del cliente, y dirección personal.

Eventos

- **cmbFiltrar_SelectedIndexChanged(object sender, EventArgs e)**:
    - Se activa cuando el usuario selecciona un número de factura en el ComboBox.

- Llama al método FiltrarPedido() para mostrar el pedido correspondiente en el DataGridView.
- Habilita los botones para marcar como entregado, retrasado y limpiar el filtro.

- **btnLimpiarFiltro_Click(object sender, EventArgs e)**:
  - Restablece el ComboBox a su estado original (sin selección) y vacía el DataGridView.
  - Deshabilita los botones de acción después de limpiar el filtro.
- **btnMarcarEntregado_Click(object sender, EventArgs e)**:
  - Permite al piloto marcar un pedido como entregado. Solicita una calificación para el cliente usando un cuadro de entrada (InputBox).
  - Actualiza el estado del pedido en la base de datos y registra la calificación del cliente.
- **btnRetrasado_Click(object sender, EventArgs e)**:
  - Permite al piloto marcar un pedido como retrasado. Actualiza el estado del pedido en la base de datos.

Funcionalidades

- **Gestión de Estados de Pedidos**: Los pilotos pueden actualizar el estado de un pedido a entregado o retrasado, reflejando cambios en tiempo real en la base de datos.
- **Registro de Calificaciones**: Al marcar un pedido como entregado, se solicita al piloto que ingrese una calificación para el cliente, lo que ayuda a mantener un registro de la satisfacción del cliente.
- **Interfaz Intuitiva**: La interfaz está diseñada para ser sencilla y fácil de usar, permitiendo una rápida actualización de los pedidos mediante un sistema de botones claramente definido.

Este formulario es fundamental para la gestión de entregas en la aplicación, asegurando que los pilotos puedan mantener actualizada la información sobre el estado de sus pedidos y contribuir al proceso de retroalimentación del cliente.

REQUERIMIENTOS TÉCNICOS

- **MySQL**: Versión 8.0 o superior.
- **.NET Framework**: Versión 4.7.2 o superior.
- **Visual Studio 2019 o superior**.
- **Sistema Operativo**: Windows 10 o superior.
- **Espacio en Disco**: Al menos 500 MB para la instalación de las aplicaciones y bases de datos.
- **Conexión a Internet**: Requerida para la integración con la API de Google Maps.
- **Bibliotecas necesarias**: MySQL.Data, Google.Maps, System.Data.

ESTANDARIZACIÓN DE CÓDIGO

CONVENCIÓN DE NOMBRES

TABLAS Y CAMPOS

Tipo de Elemento	Convención de Nombres	Ejemplo
Tabla	Singular, PascalCase	Cliente, Pedido
Campo	CamelCase	idCliente, nombreProducto
Llave Foránea	Referencia a tabla	idPersona, idPedido

CONTROLES DE UI

Tipo de Control	Prefijo	Ejemplo
TextBox	txt	txtCorreo, txtNombre
Button	btn	btnLogin, btnSubmit
GroupBox	gbx	gbxUserDetails
Label	lbl	lblCorreo, lblNombre
ComboBox	cmb	cmbProductos, cmbEstado
DataGridView	dgv	dgvPedidos, dgvDetalles
Form	frm	frmLogin, frmMenu
PictureBox	pb	pbImagenProducto

ESTILO DE CÓDIGO

- **Indentación**: Uso consistente de indentación de 4 espacios para mejorar la legibilidad.
- **Comentarios**: Comentarios claros y concisos para explicar bloques de código y su funcionalidad. Ejemplo:

```
// Verifica si el usuario es un cliente válido antes de iniciar sesión
```

Nombres de Variables:

- **Privadas**: Usar _camelCase para las variables privadas. Ejemplo: _connectionString.
- **Públicas**: Usar PascalCase para propiedades y métodos públicos. Ejemplo: IniciarSesion().

Nombres de Métodos: Usar verbos en PascalCase que describan claramente la acción que realiza el método. Ejemplo: ActualizarDatos(), CargarProductos().

Manejo de Errores: Implementar bloques try-catch para capturar y manejar excepciones, con mensajes de error claros para el usuario.

```
try
{
   // Código para abrir la conexión a la base de datos
}
catch (MySqlException ex)
{
   MessageBox.Show("Error al conectar con la base de datos: " + ex.Message);
}
```

CONVENCIONES ESPECÍFICAS

MANIPULACIÓN DE DATOS:

CONSULTAS SQL DEBEN ESTAR EN MAYÚSCULAS Y ALINEADAS PARA FACILITAR SU LECTURA

Ejemplo:

```
SELECT nombreProducto, precio
FROM productos
WHERE idProducto = @idProducto;
```

E**structura de Clases**:

- Las clases deben estar organizadas con la siguiente estructura:
    1. Propiedades públicas.
    2. Métodos públicos.
    3. Métodos privados.

4. Constructores al inicio de la clase.

Eventos de Controles: Los eventos deben seguir la convención Control_Evento. Ejemplo: btnLogin_Click, txtCorreo_KeyPress.

Ejemplos

1. **Métodos y Propiedades**:

```
public void IniciarSesion()
{
  // Código para iniciar sesión
}

private string ObtenerDireccionIp()
{
  // Código para obtener la IP del usuario
}
```

Eventos y Controles:

```
private void btnLogin_Click(object sender, EventArgs e)
{
  IniciarSesion();
}

private void txtCorreo_KeyPress(object sender, KeyPressEventArgs e)
{
  if (e.KeyChar == (char)Keys.Enter)
  {
    IniciarSesion();
  }
}
```

Manipulación de Datos:

```
using (MySqlConnection connection = new MySqlConnection(_connectionString))
{
  try
  {
    connection.Open();
    string query = @"
      SELECT No_Factura, Fecha_Solicitud, Estado
      FROM pedidos
      WHERE idCliente = @idCliente";
    MySqlCommand cmd = new MySqlCommand(query, connection);
    cmd.Parameters.AddWithValue("@idCliente", idCliente);
    MySqlDataReader reader = cmd.ExecuteReader();

    // Procesamiento de datos
  }
  catch (MySqlException ex)
  {
    MessageBox.Show("Error al cargar los datos: " + ex.Message);
  }
}
```

Este conjunto de convenciones y reglas está diseñado para asegurar que todo el código dentro del sistema de reparto sea consistente, fácil de mantener y comprensible para cualquier desarrollador que trabaje en el proyecto en el futuro.

CONCLUSIONES

El desarrollo del sistema de reparto compuesto por tres aplicaciones distintas (Administrativa, Pilotos y Clientes) ha permitido la creación de una solución integral que responde a las necesidades específicas de cada usuario involucrado en el proceso de entrega. A lo largo del proyecto, se han alcanzado varios objetivos importantes que fortalecen la funcionalidad y eficiencia del sistema:

1. **Integración Efectiva**: Se logró integrar tres aplicaciones de manera efectiva, permitiendo una comunicación fluida entre administradores, pilotos y clientes. Esto ha optimizado la gestión de pedidos, asignación de rutas, y seguimiento de entregas.

2. **Optimización del Flujo de Trabajo**: La aplicación administrativa ofrece herramientas poderosas para la gestión de usuarios, pedidos, y rutas, facilitando las operaciones CRUD y permitiendo a los administradores tomar decisiones informadas a través de la generación de reportes.

3. **Facilidad de Uso y Eficiencia**: La aplicación de pilotos se diseñó para maximizar la eficiencia en las entregas, integrando la API de Google Maps para la optimización de rutas. Los pilotos pueden verificar sus asignaciones y registrar las entregas de manera sencilla, lo que mejora su desempeño diario.

4. **Mejora en la Experiencia del Cliente**: La aplicación de clientes ha sido diseñada con un enfoque en la usabilidad, permitiendo a los usuarios realizar pedidos y seguir el estado de los mismos de manera intuitiva. La posibilidad de valorar las entregas también contribuye a mejorar la calidad del servicio y la satisfacción del cliente.

5. **Estándar de Codificación**: La estandarización del código fue fundamental para mantener la coherencia y la legibilidad del sistema, asegurando que el mantenimiento futuro del software sea manejable y eficiente.

6. **Cumplimiento de Requerimientos Técnicos**: Se utilizaron las tecnologías más adecuadas para el desarrollo, incluyendo MySQL, .NET Framework, y Visual Studio, garantizando que el sistema es robusto y cumple con los estándares actuales de desarrollo de software.

REFERENCIAS

Documentación de Google Maps Platform. (2024). Google for Developers. https://developers.google.com/maps/documentation?hl=es-419

Williams, D. (2024, March 13). *Usability*. Digital.gov. https://digital.gov/topics/usability/

MySQL :: MySQL Documentation. (2024). Mysql.com. https://dev.mysql.com/doc/

dotnet-bot. (2024). *.NET Framework documentation*. Microsoft.com. https://learn.microsoft.com/en-us/dotnet/framework/

BillWagner. (2024). *C# Guide - .NET managed language*. Microsoft.com. https://learn.microsoft.com/en-us/dotnet/csharp/

ghogen. (2022). *Visual Studio product family documentation*. Microsoft.com. https://learn.microsoft.com/en-us/visualstudio/?view=vs-2019

jcjiang. (2024). *Entity Framework documentation hub*. Microsoft.com. https://learn.microsoft.com/en-us/ef/

MySQL :: MySQL Connector/NET Developer Guide. (2024). Mysql.com. https://dev.mysql.com/doc/connector-net/en/

Git - Documentation. (2024). Git-Scm.com. https://git-scm.com/doc

What is REST?: REST API Tutorial. (2023, December 11). REST API Tutorial. https://restfulapi.net/