Manual técnico

App delivery

Contenido

1. Descripción General de la Solución 3
1.1. Introducción 3
Diccionario de clases 4
Paquete Lógica 4
Aministrador 4
Datos Globales 4
Main 5
Motocicleta 5
Pedido 5
Producto 6
Usuario 6
Vehículo 7
Paquete TableModels 7
PedidoTableModel 7
ProductosTableModel 8
RegistrosTableModel 8
Paquete Windows 9
VtnAgregarProductos 9
VtnEstadoEntregas 9
VtnIngreso 10
VtnOpcionesAdmin 10
VtnRealizarPedidos 11
VtnRegistroEntregas 12
Paquete eventos 12
RecorridoGrafico 12

1. Descripción General de la Solución

1.1. Introducción

El Manual Técnico de App delivery es una guía detallada destinada a proporcionar a los desarrolladores, ingenieros y administradores del sistema una visión profunda de la solución que se ha desarrollado. Este documento se crea con el propósito de ayudar a entender, mantener y optimizar la aplicación App delivery

Este manual está estructurado para brindar información fundamental sobre la solución, su arquitectura, los componentes principales y los procesos clave involucrados en la ejecución de la aplicación. Además, se presenta un diccionario de clases que describe en detalle las clases y objetos utilizados en el desarrollo, lo que facilita la comprensión y el mantenimiento del código.

En la sección de Diagramas de Flujo, se presentan visualmente los procesos clave y la interacción entre los componentes. Estos diagramas son valiosos para comprender el flujo de trabajo de la aplicación y para identificar áreas de mejora.

Este manual se mantiene actualizado para reflejar los cambios en la aplicación y se espera que sea una herramienta esencial para todo el equipo de desarrollo y para aquellos que participen en la implementación y el mantenimiento de App delivery

A medida que se adentre en este manual, descubrirá una descripción exhaustiva de la solución, los requisitos del sistema, las instrucciones de instalación y configuración, un diccionario de clases y diagramas de flujo que guiarán a través de la aplicación App delivery

Diccionario de clases

Paquete Lógica

Aministrador

Dentro de esta clase se encuentra la lógica que sigue el rol de administrador, este puede agregar productos y realizar todo el proceso de los pedidos.

Atributos

- Nombre
- Contraseña

Métodos

- Agregar productos: este método ejecuta la lógica al registrar nuevos productos al sistema, este pide datos como el nombre, precio, cantidad y los procede a agregar dentro del sistema .
- Agregar productos al pedido: este método recibe un id del pedido y el producto que fue agregado, se hace una condicional para ver si el pedido existe en primera instancia y si existe se agrega el producto a ese pedido.

Datos Globales

Esta es una clase dnominada singleton, la cual nos permite tener una sola instancia de esta clase, en esta clase se manejan los datos que son de utilidad para todo el programa y que se necesitan en diversas clases.

Atributos

- MOTOCICLETAS
- PRODUCTOS
- PEDIDOS
- Administrador

Métodos

- Actualizar con datos: este método recibe un objeto tipo Datos globales el cual se utiliza para mantener la serialización de nuestro programa y así no perder datos
- Guardar datos: Este método es responsable de guardar los datos del objeto actual (instancia de la clase en la que se encuentra) en un archivo de datos serializados. La serialización es un proceso que convierte el objeto en una secuencia de bytes que puede almacenarse en un archivo o enviarse a través de la red.
  El método utiliza una función llamada serializarDatos para realizar la serialización y guardar los datos en un archivo específico llamado "datos.dat". Los datos guardados pueden incluir información relevante para la instancia actual, como listas de pedidos, motocicletas y otros atributos específicos de la aplicación.
- Serializar Datos: Este método se encarga de llevar a cabo la serialización de los datos de la clase DatosGlobales y almacenarlos en un archivo en el sistema de archivos. La serialización es el proceso de convertir objetos de Java en una secuencia de bytes para que puedan ser guardados en disco o transmitidos a través de la red. En este caso, se está

serializando la instancia de la clase DatosGlobales y guardándola en un archivo cuya ruta se especifica como argumento.
- DeSerializarDatos: Este método se encarga de realizar la deserialización de los datos guardados previamente en un archivo y restaurarlos en una instancia de la clase DatosGlobales. La deserialización es el proceso inverso de la serialización, donde los objetos almacenados en un archivo son recuperados y reconstruidos en objetos de Java.

Main

La clase Main es la puerta de entrada de la aplicación y se encarga de inicializar componentes esenciales, cargar datos previos si existen y gestionar el cierre adecuado de la aplicación con la serialización de datos. También, establece el aspecto visual de la aplicación. En resumen, es el control central de la aplicación en su inicio y cierre.

Motocicleta

La clase Motocicleta representa un tipo de vehículo que se utiliza en la aplicación. Está diseñada para almacenar información sobre las motocicletas, como su nombre, marca y velocidad. Además, gestiona si la motocicleta está ocupada o no y puede almacenar un pedido asociado.

Atributos

- nombre (heredado de la clase Vehiculo): Representa el nombre de la motocicleta.
- marca (heredado de la clase Vehiculo): Indica la marca de la motocicleta.
- velocidad (heredado de la clase Vehiculo): Almacena la velocidad de la motocicleta.
- pedido: Un objeto de tipo Pedido que se asocia con esta motocicleta.
- motoOcupada: Un booleano que indica si la motocicleta está ocupada (true) o desocupada (false).

Métodos

- desocuparMoto(): Este método marca la motocicleta como desocupada, estableciendo el valor de motoOcupada en false.
- ocuparMoto(): Este método marca la motocicleta como ocupada, estableciendo el valor de motoOcupada en true.
- isMotoOcupada(): Retorna un valor booleano que indica si la motocicleta está ocupada (true) o no (false).

Pedido

Atributos

- id: Un identificador único del pedido, generado utilizando la clase UUID.
- productos: Una lista de productos asociados con el pedido.
- vehiculoName: El nombre del vehículo (en este caso, motocicleta) asignado al pedido.
- horaPartida: La hora de partida del pedido, que indica cuándo se inició la entrega.
- horaLlegada: La hora de llegada del pedido, que indica cuándo se completó la entrega.
- distancia: La distancia que se debe recorrer para entregar el pedido.
- total: El monto total del pedido.
- motoAsignada: Un booleano que indica si una motocicleta ha sido asignada al pedido.
- pedidoFinalizado: Un booleano que indica si el pedido ha sido finalizado.

Métodos

- actualizarTotal(double monto): Actualiza el monto total del pedido sumando un valor específico al total actual. Retorna el nuevo total.
- agregarProducto(Producto producto): Agrega un producto a la lista de productos asociados con el pedido.
- configurarHoraSalida(Date horaSalida): Establece la hora de partida del pedido.
- configurarHoraLlegada(Date horaLlegada): Establece la hora de llegada del pedido.
- FinalizarPedido(): Marca el pedido como finalizado, estableciendo el valor de pedidoFinalizado en true.
- isPedidoFinalizado(): Retorna un valor booleano que indica si el pedido ha sido finalizado (true) o no (false).
- asignarMoto(): Marca el pedido como asignado a una motocicleta, estableciendo el valor de motoAsignada en true.
- isMotoAsignada(): Retorna un valor booleano que indica si el pedido ha sido asignado a una motocicleta (true) o no (false).

Producto

La clase Producto representa un producto que se puede agregar a un pedido en la aplicación. Cada instancia de esta clase almacena información sobre el producto, como su nombre, precio y cantidad.

Atributos

- nombre: El nombre del producto.
- precio: El precio del producto.
- cantidad: La cantidad del producto que existe.

Métodos

- Esta clase no contiene métodos adicionales.

Usuario

La clase Usuario representa a un usuario de la aplicación y almacena información básica sobre el usuario, incluyendo su nombre y contraseña. Esta clase se utiliza para gestionar y autenticar a los usuarios dentro de la aplicación.

Atributos

- nombre: El nombre del usuario.
- contrasena: La contraseña del usuario.

Métodos

- Esta clase no contiene métodos adicionales.

Vehículo

La clase Vehiculo es la clase base para representar cualquier tipo de vehículo en la aplicación. Almacena información básica sobre un vehículo, como su nombre, marca y velocidad.

Atributos

- nombre: El nombre del vehículo.
- marca: La marca del vehículo.
- velocidad: La velocidad máxima del vehículo.

Métodos

- Esta clase no contiene métodos adicionales

Paquete TableModels

PedidoTableModel

La clase PedidoTableModel es un modelo de tabla personalizado que se utiliza para representar los productos de un pedido en una tabla. Esta clase extiende AbstractTableModel para proporcionar un modelo de datos adecuado para su visualización en una interfaz de usuario.

Atributos

- datosGlobales: Una instancia de la clase DatosGlobales, que se utiliza para acceder a los datos globales de la aplicación.
- columnNames: Un array de cadenas que almacena los nombres de las columnas de la tabla (en este caso, "Nombre" y "Precio").
- pedidoRec: El pedido asociado a la tabla.
- productosPedido: Una lista de productos que componen el pedido.

Métodos

- gregarProductoAlPedido(Producto producto): Este método permite agregar un producto al pedido actual y luego notifica a la tabla que los datos han cambiado, lo que activa una actualización en la interfaz de usuario.
- getRowCount(): Devuelve el número de filas en la tabla, que es igual al número de productos en el pedido.
- getColumnCount(): Devuelve el número de columnas en la tabla, que es igual a la longitud del array columnNames.
- getValueAt(int rowIndex, int columnIndex): Devuelve el valor que se mostrará en una celda específica de la tabla, basándose en el índice de fila y columna proporcionado. En este caso, devuelve el nombre y el precio de un producto según la columna correspondiente.
- getColumnName(int column): Devuelve el nombre de la columna en el índice especificado. Esto se utiliza para mostrar los encabezados de las columnas en la tabla.

ProductosTableModel

La clase ProductosTableModel es un modelo de tabla personalizado que se utiliza para representar productos en una tabla. Esta clase extiende AbstractTableModel para proporcionar un modelo de datos adecuado para su visualización en una interfaz de usuario.

Atributos

- datosGlobales: Una instancia de la clase DatosGlobales, que se utiliza para acceder a los datos globales de la aplicación.
- productos: Una lista de productos obtenida de los datos globales.
- columnNames: Un array de cadenas que almacena los nombres de las columnas de la tabla (en este caso, "Nombre" y "Precio").

Métodos

- AgregarProducto(String nombre, String precio, String cantidad): Este método permite agregar un producto con un nombre, precio y cantidad específicos a la lista de productos en los datos globales. Luego notifica a la tabla que los datos han cambiado, lo que activa una actualización en la interfaz de usuario.
- getRowCount(): Devuelve el número de filas en la tabla, que es igual al número de productos en la lista de productos.
- getColumnCount(): Devuelve el número de columnas en la tabla, que es igual a la longitud del array columnNames.
- getValueAt(int rowIndex, int columnIndex): Devuelve el valor que se mostrará en una celda específica de la tabla, basándose en el índice de fila y columna proporcionado. En este caso, devuelve el nombre y el precio de un producto según la columna correspondiente.
- getColumnName(int column): Devuelve el nombre de la columna en el índice especificado. Esto se utiliza para mostrar los encabezados de las columnas en la tabla.

RegistrosTableModel

La clase RegistrosTableModel es un modelo de tabla personalizado utilizado para representar registros de pedidos verificados en una tabla. Esta clase extiende AbstractTableModel y se utiliza para proporcionar un modelo de datos adecuado para su visualización en una interfaz de usuario.

Atributos

- datosGlobales: Una instancia de la clase DatosGlobales, que se utiliza para acceder a los datos globales de la aplicación.
- pedidosVerificados: Una lista de pedidos verificados que se obtiene de los datos globales. Estos son pedidos que han sido finalizados.
- columnNames: Un array de cadenas que almacena los nombres de las columnas de la tabla, que incluyen "Vehículo", "Distancia (km)", "Monto", "Fecha y hora de creación" y "Fecha y hora de entrega".

Métodos

- RegistrosTableModel(): Constructor de la clase que inicializa el modelo de tabla. En este constructor, se filtran los pedidos verificados de los datos globales y se almacenan en la lista pedidosVerificados.
- getRowCount(): Devuelve el número de filas en la tabla, que es igual al número de pedidos verificados en la lista pedidosVerificados.
- getColumnCount(): Devuelve el número de columnas en la tabla, que es igual a la longitud del array columnNames.
- getValueAt(int rowIndex, int columnIndex): Devuelve el valor que se mostrará en una celda específica de la tabla, basándose en el índice de fila y columna proporcionado. En este caso, devuelve información relacionada con el vehículo, la distancia, el monto, la fecha y hora de creación, y la fecha y hora de entrega de un pedido verificado.
- getColumnName(int column): Devuelve el nombre de la columna en el índice especificado. Esto se utiliza para mostrar los encabezados de las columnas en la tabla.

Paquete Windows

VtnAgregarProductos

La clase VtnAgregarProductos representa la ventana de la interfaz de usuario donde los usuarios pueden agregar productos a la aplicación. Esta ventana permite ingresar información sobre el nombre, precio y cantidad de un producto y luego agregarlo a la lista de productos disponibles.

Atributos

- productosTableModel: Una instancia de la clase ProductosTableModel que se utiliza para manejar la tabla de productos disponibles en la ventana.

Métodos

- VtnAgregarProductos(): Constructor de la clase que inicializa la ventana y sus componentes. También inicializa el modelo de tabla productosTableModel.
- btnAgregarProducto Un método de evento que se llama cuando se hace clic en el botón "Agregar". Este método invoca el método AgregarProducto del modelo de tabla productosTableModel para agregar un producto con la información ingresada a la lista de productos.
- jButton2ActionPerformed): Un método de evento que se llama cuando se hace clic en el botón "Regresar". Este método cierra la ventana actual.

VtnEstadoEntregas

La clase VtnEstadoEntregas representa una ventana de la interfaz de usuario que muestra un escenario con tres motocicletas y permite enviar cada motocicleta a realizar entregas. La ventana también incluye un botón para enviar todas las motocicletas y otro para regresar.

Atributos

- datosGlobales: Un objeto de la clase DatosGlobales que se utiliza para acceder a los datos globales de la aplicación.
- pedidos: Una lista de objetos Pedido que almacena los pedidos disponibles en la aplicación.

Métodos

- VtnEstadoEntregas(): Constructor de la clase que inicializa la ventana y sus componentes. También carga la imagen de una motocicleta y establece sus ubicaciones iniciales.
- enviarMoto: Método para enviar una motocicleta a realizar entregas. Busca un pedido no finalizado asociado a la motocicleta y crea un hilo (RecorridoGrafico) para mover la motocicleta en el escenario.
- btnEnviarMoto1ActionPerformed: Método de evento que se llama cuando se hace clic en el botón "Enviar" de la primera motocicleta.
- btnEnviarSegundaMotoActionPerformed: Método de evento que se llama cuando se hace clic en el botón "Enviar" de la segunda motocicleta.
- btnEnviarTerceraMotoActionPerformed: Método de evento que se llama cuando se hace clic en el botón "Enviar" de la tercera motocicleta.
- btnEnviarTodosActionPerformed:Método de evento que se llama cuando se hace clic en el botón "Enviar Todos" para enviar todas las motocicletas.
- jButton1ActionPerformed:Método de evento que se llama cuando se hace clic en el botón "Regresar" para cerrar la ventana.

VtnIngreso

La clase VtnIngreso representa una ventana de la interfaz de usuario que se utiliza para ingresar credenciales de administrador en la aplicación. Esta ventana permite al administrador iniciar sesión proporcionando su nombre de usuario y contraseña.

Atributos

- datosGlobales: Un objeto de la clase DatosGlobales que se utiliza para acceder a los datos globales de la aplicación.
- administrador: Un objeto de la clase Administrador que almacena los datos del administrador.

Métodos

- VtnIngreso(): Constructor de la clase que inicializa la ventana y sus componentes.
- btnIngresarActionPerformed(): Método de evento que se llama cuando se hace clic en el botón "Ingresar". Verifica si las credenciales ingresadas coinciden con las del administrador y, si es así, abre la ventana de opciones del administrador (VtnOpcionesAdmin). Si las credenciales no coinciden, muestra "no" en la consola. También imprime en la consola la lista de productos y pedidos disponibles en la aplicación.

VtnOpcionesAdmin

La clase VtnOpcionesAdmin representa una ventana de la interfaz de usuario que muestra las opciones disponibles para el administrador de la aplicación. Estas opciones permiten al administrador realizar diferentes tareas de administración, como realizar pedidos, ver el estado de los envíos, acceder a informes de entregas y agregar nuevos productos.

Atributos

- no tiene

Métodos

- VtnOpcionesAdmin(): Constructor de la clase que inicializa la ventana y sus componentes.

- btnAgregarNuevosProdActionPerformed(evt): Método de evento que se llama cuando se hace clic en el botón "Agregar Nuevos Productos". Abre la ventana para agregar nuevos productos (VtnAgregarProductos).
- btnRealizarPedidosActionPerformed(evt): Método de evento que se llama cuando se hace clic en el botón "Realizar Pedidos". Abre la ventana para realizar pedidos (VtnRealizarPedidos).
- btnVerEstadoEnviosActionPerformed(evt): Método de evento que se llama cuando se hace clic en el botón "Ver Estado de Envíos". Abre la ventana para ver el estado de los envíos (VtnEstadoEntregas).

VtnRealizarPedidos

La clase VtnRealizarPedidos es una ventana de interfaz de usuario (GUI) que permite a los usuarios realizar pedidos de productos. Los usuarios pueden seleccionar productos de una tabla, agregarlos al pedido, especificar la distancia a recorrer y asignar una motocicleta para la entrega.

Atributos

- datosGlobales: Un objeto de la clase DatosGlobales que proporciona acceso a datos globales compartidos en la aplicación.

- productosTableModel: Un objeto de la clase ProductosTableModel que se utiliza para mostrar productos en una tabla.

- pedidoTableModel: Un objeto de la clase PedidoTableModel que se utiliza para mostrar los productos en el pedido.

- pedido: Un objeto de la clase Pedido que representa el pedido actual.

Métodos

- VtnRealizarPedidos(): Constructor de la clase que inicializa la ventana y sus componentes. También crea un nuevo objeto de pedido y configura los modelos de tabla para mostrar productos y el pedido actual.

- actualizarCbb(): Método que actualiza un JComboBox (cbbMotocicletas) con motocicletas disponibles para asignar al pedido.

- btnAgregarAlPedidoActionPerformed(ActionEvent evt): Método de evento que se llama cuando se hace clic en el botón "Agregar Productos al pedido". Permite agregar un producto seleccionado al pedido y actualiza el total.

- btnConfirmarPedidoActionPerformed(ActionEvent evt): Método de evento que se llama cuando se hace clic en el botón "Confirmar pedido". Confirma y envía el pedido con la información seleccionada, incluyendo la distancia a recorrer y la motocicleta asignada.

- jButton1ActionPerformed(ActionEvent evt): Método de evento que se llama cuando se hace clic en el botón "Regresar". Cierra la ventana y permite al usuario regresar al menú anterior.

VtnRegistroEntregas

La clase VtnRegistroEntregas es una ventana de interfaz de usuario (GUI) que muestra los registros de entregas realizadas. Los registros incluyen información sobre los pedidos entregados, como la hora de entrega, el nombre del cliente, la distancia recorrida, entre otros.

Atributos

- registrosTableModel: Un objeto de la clase RegistrosTableModel que se utiliza para mostrar los registros de entregas en una tabla.

Métodos

- VtnRegistroEntregas(): Constructor de la clase que inicializa la ventana y sus componentes. También configura el modelo de tabla para mostrar los registros de entregas.

- jButton1ActionPerformed(ActionEvent evt): Método de evento que se llama cuando se hace clic en el botón "Regresar". Cierra la ventana y permite al usuario regresar al menú anterior.

Paquete eventos

RecorridoGrafico

La clase RecorridoGrafico se utiliza para simular el recorrido gráfico de una motocicleta que realiza una entrega de un pedido en una interfaz de usuario. La motocicleta se mueve en la pantalla desde un punto de inicio hasta un punto de destino, y luego regresa al punto de inicio.

Atributos

- pedido: Un objeto de la clase Pedido que representa el pedido que se está entregando.
- labelMoto: Un objeto de la clase JLabel que se utiliza para mostrar gráficamente la motocicleta en la interfaz de usuario.
- escenario: Un objeto de la clase VtnEstadoEntregas que representa el escenario en el que se muestra la entrega.
- vtnRealizarPedidos: Un objeto de la clase VtnRealizarPedidos que se utiliza para actualizar la interfaz de usuario cuando se completa el recorrido.
- registrosTableModel: Un objeto de la clase RegistrosTableModel que se utiliza para actualizar los registros de entregas.

Métodos

- RecorridoGrafico(JLabel moto, VtnEstadoEntregas escenario, Pedido pedido): Constructor de la clase que recibe como parámetros un objeto JLabel que representa la motocicleta, un objeto VtnEstadoEntregas que representa el escenario, y un objeto Pedido que representa el pedido a entregar.

- obtenerMoto(): Método privado que busca y devuelve una motocicleta en función del nombre del vehículo especificado en el pedido.

- run(): Método principal que se ejecuta cuando se inicia el hilo (Thread). Controla el movimiento de la motocicleta en la interfaz gráfica, simulando el recorrido de entrega. Una vez que se completa la entrega y la motocicleta regresa al punto de inicio, se actualiza la interfaz de usuario y se registra la hora de llegada.