Solución tecnológica para comercializadoras de vehículos

Rojas Serrano Holmer Mateo

Garcia Paez John Sebastian

Villamil Cardozo Geremi Santiago

Corporación Universitaria Minuto de Dios

Facultad de Ingeniería

Ingeniería de Sistemas

Programación Orientada a Objetos NRC 60 – 80841

Bogotá D.C, octubre de 2025

Tabla de Contenido

Fase 0:

Contextualización del enunciado del proyecto:

1. ¿Qué debemos hacer?

Desarrollar una solución POO para una red de comercializadoras y vitrinas que centralice la preventa y la venta de vehículos: gestión de prospectos, test-drive, cotizaciones, órdenes y un inventario unificado con reportes confiables.

2. ¿Qué se necesita?

Un sistema con CRM de prospectos e interacciones, sincronización de inventario por vitrina/concesionario, flujo de cotización→orden→entrega, campañas trazables y reportes automáticos.

3. Elementos (objetos) que se identifican

Prospecto, Cliente, Interacción, Asesor, Campaña, Vehículo, Marca, Modelo, Inventario, Vitrina, Concesionario, Cotización, Orden de Venta, Pago, Plan de Financiación, Test-Drive, Reporte, Usuario, Rol.

4. ¿Qué objetos se incluye?

Para el MVP: Prospecto, Interacción, Campaña, Asesor, Vehículo, Inventario, Vitrina, Concesionario, Cotización, Orden de Venta, Pago/Plan de Financiación, Reporte, Usuario y Rol.

5. ¿Cómo lo genera y qué objetos se requiere?

Se genera registrando leads en vitrinas/canales digitales, realizando seguimiento por asesores, sincronizando inventario por vitrina y ejecutando el flujo cotización→orden→reserva→entrega.

Fase I:

Listado de requerimientos del contexto planteado. Cada requerimiento se documenta con la tabla: Nombre, Descripción, Entradas y Salidas.

9 Grupos oficiales (funcionales y no funcionales integrados)

Nombre	R-01 Interfaz externa y GUI (Grupo 1)
Descripción	Pantallas y navegación estandarizadas; distribución, resoluciones y mensajes de error coherentes.
Entradas	• Guía de estilo y prototipos • Mapa de navegación • Resoluciones objetivo
Salidas	• Vistas consistentes • Mensajes de error uniformes • Accesos directos/menús definidos

Nombre	R-02 Interfaces de hardware (Grupo 2)
Descripción	Compatibilidad con periféricos en vitrinas (impresoras, lectores QR/código), definiendo protocolos.
Entradas	• Inventario de periféricos soportados • Drivers/configuración • Protocolos de comunicación
Salidas	• Operaciones de impresión/lectura exitosas • Lista certificada de dispositivos

Nombre	R-03 Interfaces de software (Grupo 3)
Descripción	Conexión con software externo: APIs REST/JSON, bases de datos y sistemas heredados.
Entradas	• Especificación OpenAPI • Conectores/credenciales • Esquemas de datos compartidos
Salidas	• Endpoints integrados • Importación/exportación de datos • Trazabilidad de integraciones

Nombre	R-04 Desempeño y concurrencia (Grupo 4)
Descripción	Tiempos de respuesta definidos, usuarios y operaciones concurrentes medidos en escenarios típicos.
Entradas	• Escenarios de carga • Volúmenes estimados • Herramientas de prueba
Salidas	• Informe de tiempos ≤ objetivo • Capacidad concurrente validada

Nombre	R-05 Tolerancia a fallos (safety) (Grupo 5)
Descripción	Prevención de acciones peligrosas, política de respaldo y recuperación ante pérdida/daño de información.
Entradas	• Matriz de riesgos y acciones críticas • Plan de respaldo y restauración
Salidas	• Confirmaciones y auditoría • Respaldos y restauración verificada

Nombre	R-06 Seguridad y calidad (usuario) (Grupo 6)
Descripción	Autenticación/autorización, protección de la información, disponibilidad y eficiencia de recursos.
Entradas	• Matriz de roles y permisos • Políticas de seguridad • Monitoreo de disponibilidad
Salidas	• Control de acceso aplicado • Métricas de disponibilidad • Registros de auditoría

Nombre	R-07 Interoperabilidad, confiabilidad, robustez, usabilidad, instalación (Grupo 7)
Descripción	El sistema debe interoperar con su entorno, ser confiable y usable; procesos de instalación claros.
Entradas	• Checklist de interoperabilidad y robustez • Guías de instalación/uso • Pruebas con usuarios
Salidas	• Instalación validada • Encuestas/usabilidad aprobadas • Estabilidad operativa

Nombre	R-08 Mantenibilidad y documentación (desarrollador) (Grupo 8)
Descripción	Estructura de directorios y metodología de diseño, estándares de documentación y código.
Entradas	• Guía de arquitectura • Convenciones de proyecto • Plantillas de documentación
Salidas	• Documentación técnica disponible • Estructura modular y legible

Nombre	R-09 Portabilidad, reusabilidad y pruebas (desarrollador) (Grupo 9)
Descripción	Ejecución en plataformas objetivo, componentes reutilizables y facilidades de prueba.
Entradas	• Matriz de plataformas • Strategy de pruebas y datos semilla
Salidas	• Builds multiplataforma • Librerías reutilizables • Suites de prueba automatizadas

Fase II

1. Lista de sustantivos del enunciado y clases candidatas

A partir del caso de estudio se identifican los siguientes sustantivos relevantes:

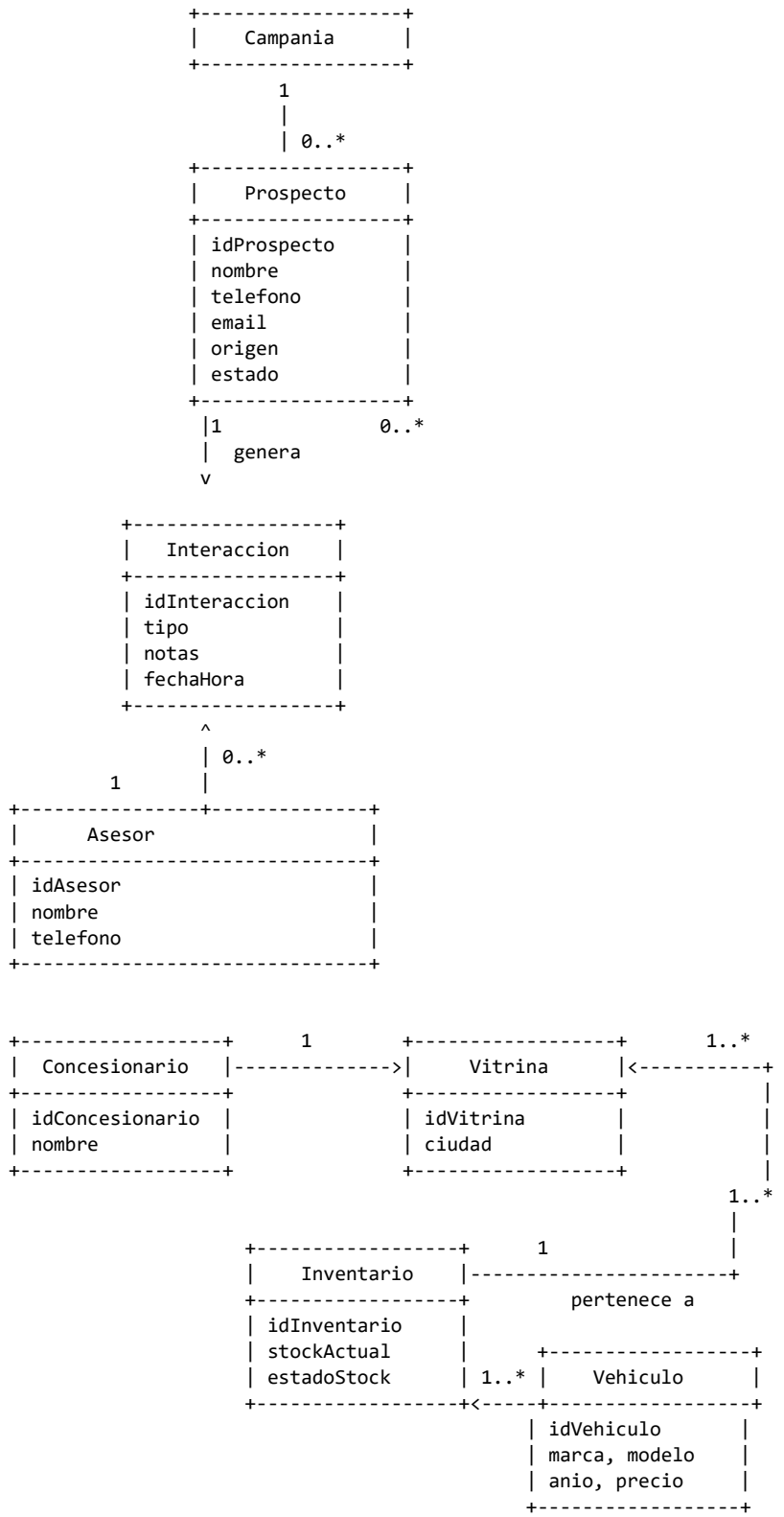
Empresa, concesionario, vitrina, red de distribución, vehículo, marca, tipo de vehículo, país de ensamble, campaña, publicidad, mercadeo, incentivo, cliente potencial (prospecto), cliente, proceso de compra, servicio, tiempo de decisión, asesor, atención, seguimiento, reporte, indicador, ventas, preventa, sistema de información.

De estos sustantivos se seleccionan como clases principales del sistema:

Prospecto, Interaccion, Asesor, Campania, Vehiculo, Vitrina, Concesionario, Inventario, Cotizacion, OrdenDeVenta, Cliente, TestDrive, Usuario, Rol, Reporte

2. Representación gráfica simplificada de clases y relaciones

A continuación se muestra una representación de las relaciones principales entre las clases del sistema. Sirve como guía para el diagrama UML de clases de la plantilla.


Campania
1
0..*
Prospecto
idProspecto
nombre
telefono
email
origen
estado
1
0..*
genera
Interaccion
idInteraccion
tipo
notas
fechaHora
0..*
1
Asesor
idAsesor
nombre
telefono
Concesionario
1
Vitrina
1..*
idConcesionario
nombre
idVitrina
ciudad
1..*
Inventario
1
pertenece a
idInventario
stockActual
estadoStock
1..*
Vehiculo
idVehiculo
marca, modelo
anio, precio

```
Prospecto 1..* ---- 0..* Cotizacion ---- 1 Vehiculo
Cotizacion 1 ---- 0..1 OrdenDeVenta
OrdenDeVenta 1 ---- 1 Vehiculo (reservado y luego vendido)
```

3. Clases principales: atributos y métodos

En esta sección se describen las clases seleccionadas para el modelo, junto con sus atributos más importantes y las operaciones (métodos) asociadas.

3.1 Clase: Prospecto

Atributos:

- idProspecto: String – identificador único del prospecto.
- nombre: String – nombre completo.
- telefono: String – número de contacto.
- email: String – correo electrónico.
- origen: String – campaña, vitrina o referencia.
- estado: String – nuevo, en_seguimiento, ganado, perdido.
- fechaRegistro: DateTime – fecha y hora de creación del registro.

Métodos:

- registrarInteraccion(tipo, notas, fechaHora): agrega una interacción al historial.
- cambiarEstado(nuevoEstado): actualiza el estado del prospecto en el proceso de venta.
- asignarAsesor(asesor): vincula el prospecto con un asesor responsable.

3.2 Clase: Interaccion

Atributos:

- idInteraccion: String – identificador único.
- prospecto: Prospecto – referencia al prospecto relacionado.
- asesor: Asesor – asesor que realiza la interacción.
- tipo: String – llamada, visita, correo, mensaje, etc.
- notas: String – detalle de lo tratado.
- fechaHora: DateTime – momento en que ocurre la interacción.

Métodos:

- resumen(): devuelve una descripción corta de la interacción.

3.3 Clase: Asesor

Atributos:

- idAsesor: String – identificador del asesor.
- nombre: String – nombre completo.
- telefono: String – contacto principal.
- email: String – correo institucional.
- vitrina: Vitrina – vitrina donde trabaja el asesor.

Métodos:

- asignarProspecto(prospecto): asocia un prospecto al asesor.
- generarCotizacion(prospecto, vehiculo): crea una cotización para un prospecto.

3.4 Clase: Campania

Atributos:

- idCampania: String – identificador de la campaña.
- nombre: String – nombre comercial.
- tipo: String – radio, redes sociales, valla, etc.
- fechaInicio: Date – inicio de la campaña.
- fechaFin: Date – fin de la campaña.
- presupuesto: Decimal – monto asignado.

Métodos:

- calcularProspectosGenerados(): cuenta los prospectos asociados a esta campaña.
- calcularConversion(): calcula el porcentaje de prospectos que se convierten en venta.

3.5 Clase: Vehiculo

Atributos:

- idVehiculo: String – identificador interno.
- vin: String – número único de chasis.
- marca: String – marca del fabricante.

- modelo: String – modelo del vehículo.
- anio: Int – año modelo.
- precioLista: Decimal – precio base de venta.
- estado: String – disponible, reservado o vendido.

Métodos:

- esDisponible(): indica si el vehículo está disponible para ser vendido.
- cambiarEstado(nuevoEstado): actualiza el estado del vehículo.

3.6 Clase: Vitrina

Atributos:

- idVitrina: String – identificador de la vitrina.
- nombre: String – nombre comercial.
- ciudad: String – ciudad donde se encuentra.
- direccion: String – dirección física.
- concesionario: Concesionario – concesionario al que pertenece.

Métodos:

- obtenerStock(): consulta el inventario asociado a la vitrina.
- contarProspectosActivos(): devuelve el número de prospectos en seguimiento.

3.7 Clase: Inventario

Atributos:

- idInventario: String – identificador del registro de inventario.
- vitrina: Vitrina – vitrina responsable.
- vehiculo: Vehiculo – vehículo al que corresponde el registro.
- estadoStock: String – disponible, reservado o vendido.
- fechaActualizacion: DateTime – fecha de última actualización.

Métodos:

- reservar(ordenVenta): marca el vehículo como reservado para una orden de venta.
- liberar(): devuelve el vehículo a disponible si la venta no se concreta.

3.8 Clase: Cotizacion

Atributos:

- idCotizacion: String – identificador de la cotización.
- prospecto: Prospecto – destinatario de la cotización.
- vehiculo: Vehiculo – vehículo ofertado.
- asesor: Asesor – asesor que la genera.
- precioFinal: Decimal – precio después de descuentos.
- descuento: Decimal – valor o porcentaje de descuento.
- vigencia: Date – fecha límite de validez.
- estado: String – abierta, aceptada o vencida.

Métodos:

- calcularPrecioFinal(): calcula el precio final aplicando el descuento.
- esVigente(): indica si la cotización no ha vencido.
- marcarComoAceptada(): cambia el estado y permite generar la orden de venta.

3.9 Clase: OrdenDeVenta

Atributos:

- idOrden: String – identificador de la orden.
- cotizacion: Cotizacion – cotización origen.
- vehiculo: Vehiculo – vehículo asociado.
- prospecto: Prospecto – comprador (o cliente).
- estado: String – creada, pagada o entregada.
- fechaCreacion: DateTime – momento en que se genera la orden.
- fechaEntrega: DateTime – fecha programada de entrega.

Métodos:

- registrarPago(monto): registra un pago asociado a la orden.
- programarEntrega(fecha): define la fecha de entrega del vehículo.
- cerrarOrden(): marca la orden como entregada y el vehículo como vendido.

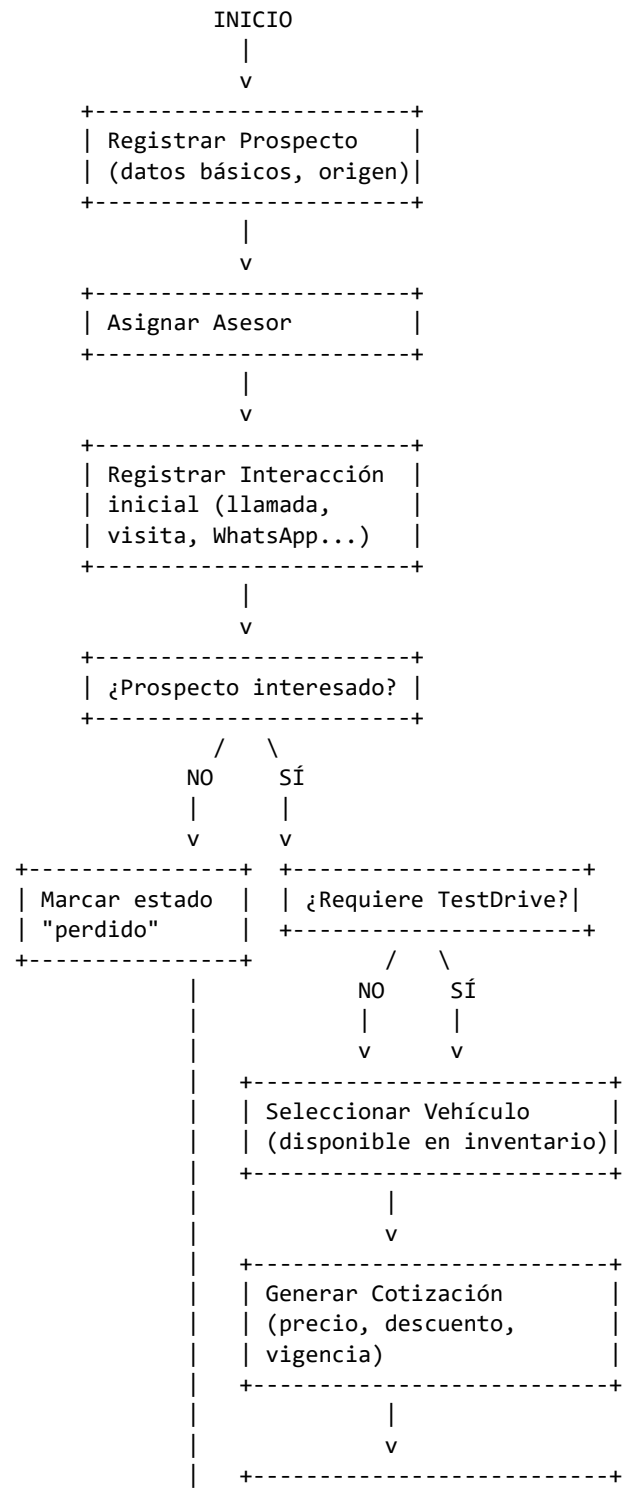
4. Documentación de atributos de la clase

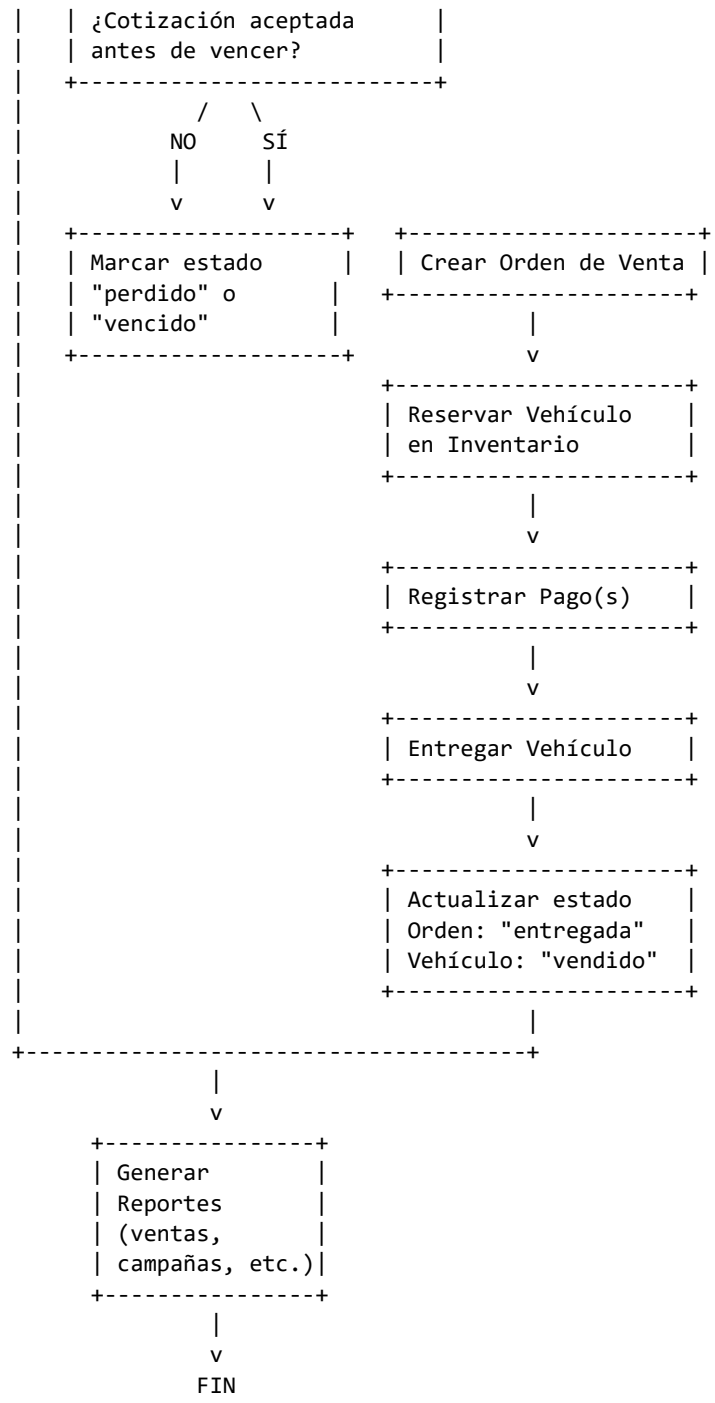
La siguiente tabla documenta algunos atributos clave de las clases, incluyendo su nombre de codificación y objetivo dentro del sistema.

Nombre de la clase	Nombre del atributo	Nombre Codificación	Objetivo
Prospecto	fechaRegistro	fecha_registro	Controlar cuándo se crea el prospecto y apoyar procesos de auditoría.
Prospecto	estado	estado	Identificar en qué etapa del proceso comercial se encuentra el prospecto.
Interaccion	tipo	tipo	Clasificar la interacción (llamada, visita, mensaje, correo, etc.).
Interaccion	fechaHora	fecha_hora	Ordenar cronológicamente el historial de interacciones.
Vehiculo	vin	vin	Identificar de forma única el chasis del vehículo.
Vehiculo	precioLista	precio_lista	Servir como base para calcular cotizaciones y descuentos.
Inventario	estadoStock	estado_stock	Indicar si el vehículo está disponible, reservado o vendido.
Cotizacion	vigencia	vigencia	Definir hasta cuándo es válida la cotización.
Cotizacion	estado	estado	Saber si la cotización está abierta, aceptada o vencida.
OrdenDeVenta	estado	estado	Controlar el avance de la orden hasta su entrega final.

5. Diagrama de flujo

El siguiente diagrama de flujo representa el recorrido principal del sistema, desde el registro de un prospecto hasta la generación de una orden de venta y la entrega del vehículo.




¿Cotización aceptada antes de vencer?
NO
SÍ
Marcar estado "perdido" o "vencido"
Crear Orden de Venta
Reservar Vehículo en Inventario
Registrar Pago(s)
Entregar Vehículo
Actualizar estado Orden: "entregada" Vehículo: "vendido"
Generar Reportes (ventas, campañas, etc.)
FIN

7. Pseudocódigo

El pseudocódigo siguiente describe el comportamiento general del sistema, utilizando las clases definidas en el diseño orientado a objetos para la comercializadora de vehículos.

```
ALGORITMO FlujoComercializadoraVehiculos

  // 1. Registro inicial del prospecto
  RECIBIR datosProspecto
  prospecto ← CREAR_PROSPECTO(datosProspecto)
  prospecto.estado ← "nuevo"

  // 2. Asignar asesor
  asesor ← SELECCIONAR_ASESOR_DISPONIBLE(vitrina)
  ASIGNAR_ASESOR_A_PROSPECTO(asesor, prospecto)

  // 3. Registrar primera interacción
  REGISTRAR_INTERACCION(
      prospecto,
      asesor,
      tipo = "contacto_inicial",
      notas = "Registro de primer acercamiento",
      fechaHora = AHORA()
  )

  // 4. Evaluar interés del prospecto
  SI NOT ESTA_INTERESADO(prospecto) ENTONCES
      prospecto.estado ← "perdido"
      GUARDAR(prospecto)
      LLAMAR ActualizarReportes()
      TERMINAR_ALGORITMO
  FIN_SI

  // 5. (Opcional) Programar test drive
  SI REQUIERE_TEST_DRIVE(prospecto) ENTONCES
      vehiculoPrueba ← SELECCIONAR_VEHICULO_DISPONIBLE()
      PROGRAMAR_TEST_DRIVE(prospecto, vehiculoPrueba, asesor, fecha = DEFINIR_FECHA())
      REGISTRAR_INTERACCION(
          prospecto,
          asesor,
          tipo = "test_drive_programado",
          notas = "Se agenda prueba de manejo",
          fechaHora = AHORA()
      )
      // Aquí podría ir la lógica de realizar/cancelar el test drive
  FIN_SI

  // 6. Seleccionar vehículo para cotización
  vehiculoCotizado ← SELECCIONAR_VEHICULO_DISPONIBLE()
  SI vehiculoCotizado = NULO ENTONCES
      MOSTRAR_MENSAJE("No hay vehículos disponibles para cotizar.")
      TERMINAR_ALGORITMO
  FIN_SI

  // 7. Generar cotización
```

```
cotizacion ← GENERAR_COTIZACION(
    prospecto,
    vehiculoCotizado,
    asesor,
    descuento = CALCULAR_DESCUENTO(campania, prospecto),
    vigencia = FECHA_ACTUAL() + 7_dias
)

// 8. Esperar respuesta del prospecto (modelo simplificado)
MIENTRAS cotizacion.estaVigente() Y NO cotizacion.aceptada HACER
    RESPUESTA ← OBTENER_RESPUESTA_PROSPECTO(cotizacion)
    SI RESPUESTA = "aceptar" ENTONCES
        cotizacion.aceptada ← VERDADERO
    SINO_SI RESPUESTA = "rechazar" ENTONCES
        SALIR_DEL_BUCLE
    FIN_SI
FIN_MIENTRAS

// 9. Decisión sobre la cotización
SI cotizacion.aceptada = FALSO ENTONCES
    SI NO cotizacion.estaVigente() ENTONCES
        cotizacion.estado ← "vencida"
    SINO
        cotizacion.estado ← "rechazada"
    FIN_SI
    prospecto.estado ← "perdido"
    GUARDAR(cotizacion)
    GUARDAR(prospecto)
    LLAMAR ActualizarReportes()
    TERMINAR_ALGORITMO
FIN_SI

// 10. Crear Orden de Venta
orden ← CREAR_ORDEN_DE_VENTA(
    cotizacion,
    prospecto,
    vehiculoCotizado
)
orden.estado ← "creada"

// 11. Reservar vehículo en inventario
resultadoReserva ← RESERVAR_VEHICULO_EN_INVENTARIO(vehiculoCotizado, orden)
SI resultadoReserva = FALLO ENTONCES
    MOSTRAR_MENSAJE("No fue posible reservar el vehículo.")
    TERMINAR_ALGORITMO
FIN_SI

// 12. Registrar pagos
MIENTRAS orden.estado ≠ "pagada" HACER
    monto ← RECIBIR_PAGO()
    REGISTRAR_PAGO_EN_ORDEN(orden, monto)
    SI SALDO_PENDIENTE(orden) = 0 ENTONCES
        orden.estado ← "pagada"
    FIN_SI
FIN_MIENTRAS
```

```
  // 13. Entregar vehículo
  PROGRAMAR_ENTREGA(orden, fechaEntrega = DEFINIR_FECHA_ENTREGA())
  ENTREGAR_VEHICULO(orden, vehiculoCotizado)
  orden.estado ← "entregada"
  vehiculoCotizado.estado ← "vendido"
  prospecto.estado ← "ganado"

  // 14. Guardar cambios finales y actualizar reportes
  GUARDAR(orden)
  GUARDAR(vehiculoCotizado)
  GUARDAR(prospecto)

  LLAMAR ActualizarReportes()

FIN_ALGORITMO


PROCEDIMIENTO ActualizarReportes()
  // Actualiza reportes de gestión comercial:
  // - Ventas por concesionario/vitrina
  // - Resultados de campañas
  // - Estado del inventario
  GENERAR_REPORTE_VENTAS()
  GENERAR_REPORTE_CAMPANIAS()
  GENERAR_REPORTE_INVENTARIO()
FIN_PROCEDIMIENTO
```

Link de Github:

https://github.com/SebastianGarcia0815/poo-comercializadora-vehiculos-g10