# Documentación Técnica: Concesionario Hakuna MaMoto

**Proyecto:** Gestión de Concesionario de Motos

**Tecnología:** .NET Framework (WPF) + Entity Framework + Crystal Reports

## Indice:

# 1.- Arquitectura del Proyecto

Para el desarrollo de la aplicación "Hakuna Mamoto", se ha implementado una **Arquitectura en Capas** estricta. Este diseño desacopla la interfaz de usuario de la lógica de negocio y el acceso a datos, garantizando un código mantenible y escalable.

El flujo de información sigue el esquema:

**Vista (WPF) ↔ Controlador (API) ↔ Modelo (Repositorio) ↔ Base de Datos**.

### Componentes de la Arquitectura:

- **Capa de Presentación (View - WPF):**
    - Desarrollada en **XAML** para el diseño visual y **C#** para la interacción.
    - Las ventanas (MainWindow, ClientesWindow, etc.) actúan como meros recolectores de datos, delegando toda la inteligencia a la capa API.
- **Capa de Lógica de Negocio (Controller/API):**
    - Es el núcleo del sistema. Aquí residen las clases ...Api.cs.
    - Su responsabilidad es **validar** los datos (formatos, fechas, reglas de negocio) antes de permitir cualquier operación en la base de datos.
- **Capa de Acceso a Datos (Model/Repo):**
    - Implementa el **Patrón Repositorio**.
    - Encapsula las operaciones de **Entity Framework**, aislando las consultas LINQ del resto de la aplicación.

# 2.- Estructura y Organización del Código

El código fuente se ha organizado en espacios de nombres (namespaces) que reflejan la separación de responsabilidades:

1. **Concesionario.ViewWPF**: Contiene las ventanas principales de gestión (CatalogoWindow, MecanicosWindow, etc.).
2. **Concesionario.ViewWPF.Views**: Subcarpeta específica para las ventanas visualizadoras de informes (InformeTallerWindow, InformeClientesWindow).
3. **Concesionario.Controller.API**: Contiene los controladores lógicos (ClienteApi, MecanicosApi, ModeloApi, ReservasApi, TallerApi).
4. **Concesionario.Model.Repo**: Contiene los repositorios de acceso a datos (ClientesRepo, MecanicosRepo, etc.).

5. **Concesionario.Model.Informes**: Aloja los definiciones de Crystal Reports (.rpt) y el DataSet tipado (ConcesionarioDS).

# 2.1.- Estructura de carpetas

```
│
├── [ Controller] (Lógica de Negocio & Validaciones)
│   └── [ API]
│       ├── ClienteApi.cs     # [!] Valida DNI, Email, Teléfono
│       ├── MecanicosApi.cs   # [!] Gestiona altas/bajas de personal
│       ├── ModeloApi.cs      # [!] Controla Stock, Matrículas y Años
│       ├── ReservasApi.cs    # [!] Lógica de señales y bloqueo de stock
│       └── TallerApi.cs      # [!] Control de fechas y costes de reparación
│
├── [ Model] (Acceso a Datos & Entidades)
│   ├── ConcesionarioModel.edmx  # Contexto de Entity Framework
│   ├── [ Repo] (Patrón Repositorio)
│   │   ├── ClientesRepo.cs   # CRUD directo a Tabla Clientes
│   │   ├── MecanicosRepo.cs  # CRUD directo a Tabla Mecánicos
│   │   ├── ModeloRepo.cs     # CRUD directo a Tabla Motos
│   │   ├── ReservasRepo.cs   # Transactional: Reservas + Updates Estado Moto
│   │   └── TallerRepo.cs     # Transactional: Mantenimientos + Updates Estado Moto
│   │
│   └── [ Informes] (Reporting)
│       ├── ConcesionarioDS.xsd  # DataSet Tipado (Esquema desconectado)
│       ├── crClientes.rpt       # Diseño: Listado General
│       ├── crReservas.rpt       # Diseño: Ventas y Señales
│       └── crTaller.rpt         # Diseño: Reparaciones
│
└── [ ViewWPF] (Interfaz de Usuario)
    ├── MainWindow.xaml          # Dashboard / Menú Principal
    ├── ClientesWindow.xaml      # Grid y Formulario Clientes
    ├── CatalogoWindow.xaml      # Grid y Formulario Stock
    ├── TallerWindow.xaml        # Gestión de Mantenimientos
    ├── ReservasWindow.xaml      # Gestión de Ventas
    ├── InformeClientesWindow.xaml # Viewer para crClientes
    └── InformeTallerWindow.xaml   # Viewer para crTaller
```

# 3.- Lógica de Negocio y Validaciones (API)

Se han implementado reglas de validación robustas para asegurar la integridad de la información.

- **Gestión de Clientes (ClienteApi.cs):**
  - **DNI:** Se valida mediante Regex (^\d{8}[A-Za-z]$) y se comprueba en la base de datos que sea único para evitar duplicados.
  - **Teléfono y Email:** Se aplican patrones de formato estricto (9 dígitos para el teléfono).
- **Gestión de Taller (TallerApi.cs):**
  - **Coherencia Temporal:** El sistema impide registrar una fecha de entrega anterior a la fecha de recepción (fecha_entrega < fecha_recepcion).
  - **Costos:** Se valida que el costo de la reparación no sea negativo.
- **Gestión de Vehículos (ModeloApi.cs):**
  - **Matrícula:** Debe cumplir el formato oficial "1234ABC" (4 números y 3 letras mayúsculas).
  - **Año:** Se valida que el año de fabricación esté en un rango lógico (>= 1950).
- **Gestión de Reservas (ReservasApi.cs):**
  - Se valida que la fecha de reserva no sea anterior a la fecha actual y que el importe de la señal sea positivo.

# 4.- Capa de Acceso a Datos

La persistencia de datos se gestiona mediante **Entity Framework**, utilizando repositorios que abstraen la complejidad de las consultas SQL.

- **Repositorio de Reservas (ReservasRepo.cs):**
  - Gestiona automáticamente el **ciclo de vida del stock**:
    - Al crear una reserva, cambia el estado de la moto a **"Reservada"**.
    - Al eliminarla, libera la moto a estado **"Disponible"**.
    - Si se edita la reserva cambiando de moto, el sistema libera la anterior y reserva la nueva en una sola transacción.
  - Utiliza Include para cargar datos relacionados (Clientes y Motos) en una sola consulta (Eager Loading).

- **Repositorio de Taller (TallerRepo.cs):**
    - Similar al anterior, marca los vehículos como **"Taller"** cuando entran a mantenimiento, impidiendo que sean vendidos mientras están en reparación.

# 5.- Módulo de Informes

Para la generación de documentos, se utiliza **SAP Crystal Reports** sobre un modelo de datos desconectado.

- **DataSet Tipado (ConcesionarioDS):**
    - Se ha diseñado un esquema XML (.xsd) que define la estructura de las tablas TablaCliente, TablaTaller y TablaReservas.
    - Esto permite desacoplar el diseño del informe de la base de datos real.
- **Informes Implementados:**
    - **crClientes.rpt:** Listado administrativo de la cartera de clientes.
    - **crTaller.rpt:** Informe de gestión de mantenimientos y productividad del taller.
    - **crReservas.rpt:** Documento de control de ventas y señales recibidas.

# 6.- Diseño de Base de Datos

El sistema se apoya en una base de datos relacional SQL Server (ConcesionarioBD).

**Tablas Principales:**

- **Clientes:** Información personal y de contacto.
- **Motos:** Inventario de vehículos. Campo clave: estado (Disponible, Reservada, Taller).
- **Mecanicos:** Personal técnico con especialidad.
- **Mantenimientos:** Tabla intermedia (N:M) que vincula Motos y Mecánicos, registrando intervenciones y costos.
- **Reservas:** Tabla intermedia (N:M) que vincula Clientes y Motos, controlando el proceso de venta.