IES RIBERA DE CASTILLA

PROYECTO SGE

CFGS Desarrollo de Aplicaciones Multiplataforma

Informática y Comunicaciones

Practica 11

Año: *2026*

Fecha de presentación: *14 de enero de 2026*

Nombre y Apellidos: David López Tomuta

Email: david.loptom@educa.jcyl.es

Contenido

1 Introducción 4
2 Estado del arte 4
2.1 Definición ERP 4
2.2 Explicar por qué se utiliza Odoo 4
2.3 Funcionamiento básico de Odoo 4
2.3.1 Sistema modular 4
2.3.2 Base de datos única 4
2.3.3 Automatización 4
3 Descripción general del proyecto 5
3.1 Objetivos 5
3.2 Entorno de trabajo (tecnologías de desarrollo y herramientas) 5
3.2.1 Hyper-V 5
3.2.2 Visual Studio Code (VS Code) 5
3.2.3 Python 3 6
3.2.4 PostgreSQL 6
3.2.5 Bootstrap 6
3.2.6 Odoo 7
4 Documentación técnica: análisis, diseño, implementación, pruebas y despliegue 8
4.1 Análisis del sistema 8
Pruebas 9
4.2 Diseño de la base de datos 9
4.3 Implementación 10
4.3.1 Estructura del proyecto 10
4.3.2 cliente.py 11
4.3.3 coche.py 12
4.3.4 marca.py 14

4.3.5 vendedor.py 15
4.3.6 venta.py 15
4.3.7 Ir.model.access.csv 17
4.3.8 models/__init__.py 17
4.3.9 controllers/controllers.py 17
4.3.10 Views/coche_detalle_template.xml 19
4.3.11 Views/ templates.xml 22
4.3.12 Views/ views_cliente.xml 24
4.3.13 Views/ views_coche.xml 25
4.3.14 Views/ views_marca.xml 28
4.3.15 Views/ views_vendedor.xml 29
4.3.16 Views/views_venta.xml 30
4.3.17 __manifest__.py 32
4.4 Pruebas y funcionamiento 33
4.5 Despliegue de la aplicación 37
5 Manual 37
5.1 Manual de usuario 37
5.2 Manual de instalación 37
6 Conclusiones y posibles ampliaciones 38
7 Bibliografía 38

1 Introducción

En este proyecto hemos creado un pequeño modulo para la gestión de coches y clientes en un concesionario, y la generación de una pagina web con los vehículos del catalogo del concesionario.

2 Estado del arte

2.1 Definición ERP

Un ERP (Enterprise Resource Planning, o Planificación de Recursos Empresariales) es un sistema de software que ayuda a una empresa a gestionar e integrar sus procesos principales en una sola plataforma. Sirve para centralizar la información y automatizar procesos.

2.2 Explicar por qué se utiliza Odoo

El uso de ODOO se debe a que es open source y gratuito, lo que nos permite crear módulos para expandir la funcionalidad y cubrir nuestras necesidades como usuario, un punto ventajoso comparado con otros ERP. Ademas Odoo usa Python el cual es un lenguaje con una curva de aprendizaje pequeña, un lenguaje que puede usar cualquier sin muchos conocimientos de informática o programación.

2.3 Funcionamiento básico de Odoo

2.3.1 Sistema modular

Odoo está formado por **módulos**. Cada módulo cubre un área del negocio:

- Ventas
- Compras
- Inventario
- Contabilidad
- CRM
- Recursos Humanos
- Producción
- Sitio web / e-commerce

Puedes **instalar o desinstalar módulos** sin afectar el resto del sistema.

2.3.2 Base de datos única

Todos los módulos trabajan sobre **una sola base de datos**.

2.3.3 Automatización

Odoo permite automatizar:

- Facturación recurrente
- Reglas de inventario (reabastecimiento)
- Recordatorios y tareas
- Flujos de aprobación

Esto reduce errores y trabajo manual.

3 Descripción general del proyecto

3.1 Objetivos

En este proyecto hemos tratado de crear un modulo que permita añadir vehículos, clientes, ventas y otros datos a la base de datos y relacionarlos, ademas de incorporar una pequeña API para poder obtener todos los vehículos del concesionarios y generar una maquina con los mismos.

3.2 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Hemos usado HiperV, VS code, Python 3, Postgress, bootstrap y Odoo.

3.2.1 Hyper-V

¿Qué es?
Es una plataforma de **virtualización de Microsoft**.

¿Para qué se usa?
Permite crear **máquinas virtuales** dentro de un mismo equipo físico.

¿Por qué se usa en proyectos con Odoo?

- Ejecutar Odoo en un entorno aislado
- Simular un servidor real
- Probar configuraciones sin afectar el sistema principal

3.2.2 Visual Studio Code (VS Code)

¿Qué es?
Un **editor de código fuente** ligero y muy potente.

¿Para qué se usa?

- Escribir y editar código
- Depurar aplicaciones
- Gestionar proyectos

Uso con Odoo

- Desarrollo de módulos Odoo
- Edición de código Python, XML y JavaScript
- Integración con Git y terminal

3.2.3 Python 3

¿Qué es?
Un **lenguaje de programación** de alto nivel.

¿Para qué se usa?

- Lógica del negocio
- Automatización
- Desarrollo backend

Uso en Odoo

- Odoo está desarrollado en **Python**
- Define modelos, reglas de negocio y procesos

3.2.4 PostgreSQL

¿Qué es?
Un **sistema gestor de bases de datos relacional**.

¿Para qué se usa?

- Almacenar datos de forma segura y estructurada

Uso en Odoo

- Guarda toda la información:
  - Clientes
  - Ventas
  - Facturas
  - Inventarios

3.2.5 Bootstrap

¿Qué es?
Un **framework de diseño web** (HTML, CSS y JavaScript).

¿Para qué se usa?

- Crear interfaces web responsivas
- Asegurar compatibilidad con móviles

Uso en Odoo

- Diseño de vistas web
- Formularios y páginas del sitio web
- Mejorar apariencia de módulos personalizados

3.2.6 Odoo

¿Qué es?
Un **ERP de código abierto**.

¿Para qué se usa?

- Gestionar procesos empresariales:
  - Ventas
  - Compras
  - Inventario
  - Contabilidad
  - CRM

¿Cómo integra todo?

- Backend en **Python**
- Base de datos **PostgreSQL**
- Frontend web con **Bootstrap**
- Desarrollo con **VS Code**
- Ejecutado en un entorno virtualizado con **Hyper-V**

4 Documentación técnica: análisis, diseño, implementación, pruebas y despliegue

4.1 Análisis del sistema

La aplicación desarrollada es un **sistema de gestión de un concesionario de coches**, implementado como un módulo personalizado en **Odoo 17**. Permite administrar marcas, coches, clientes, vendedores y ventas.

Gestión de marcas

- Crear, modificar y eliminar marcas de coches.
- Visualizar los coches asociados a cada marca mediante una relación **One2many**.

Gestión de coches

- Alta y edición de coches con los siguientes datos:
  - Nombre
  - Marca
  - Precio
  - Fecha de fabricación
  - Imagen
  - Estado (campo computado)
- Asociación con:
  - Marca (**Many2one**)
  - Clientes (**Many2many**)
- Visualización en vistas **Tree**, **Form** y **Kanban**.

Gestión de clientes

- Alta y gestión de clientes.
- Asociación de coches favoritos mediante una relación **Many2many**.
- Visualización de clientes desde el menú principal.

Gestión de vendedores

- Alta y edición de vendedores.
- Visualización de las ventas realizadas por cada vendedor (**One2many**).

Gestión de ventas

- Registro de ventas con:
  - Cliente
  - Coche
  - Vendedor
  - Precio final (campo related)
- El identificador de la venta se genera automáticamente con la fecha actual.

Manejo de errores

- Campos obligatorios (required=True) para evitar registros inválidos.
- Validación automática del ORM de Odoo.
- Errores de integridad relacional gestionados por PostgreSQL.
- Errores de carga XML detectados en tiempo de instalación del módulo.

Pruebas

- Pruebas manuales mediante la interfaz web de Odoo.

4.2 Diseño de la base de datos

La base de datos está gestionada automáticamente por **PostgreSQL**, siguiendo el diseño definido en los modelos Python.

Modelo	Tabla	Descripción
concesionario.marca	concesionario_marca	Marcas de coches
concesionario.coche	concesionario_coche	Coches del concesionario
concesionario.cliente	concesionario_cliente	Clientes
concesionario.vendedor	concesionario_vendedor	Vendedores
concesionario.venta	concesionario_venta	Ventas

El diagrama se puede visualizar de la siguiente manera:

Marca (1) ———< (N) Coche

|

Cliente (N) >———< (N) Coche

|

Cliente (1) ———< (N) Venta >——— (1) Vendedor

4.3 *I*mplementación

4.3.1 Estructura del proyecto

```
C:\Program Files\Odoo 17.0.20251123\custom_addons\practica11>tree /F
Listado de rutas de carpetas
El número de serie del volumen es EC40-DD6D
C:.
    .gitignore
    README.md
    run.ps1
    __init__.py
    __manifest__.py

├───controllers
        controllers.py
        __init__.py

├───models
        cliente.py
        coche.py
        marca.py
        vendedor.py
        venta.py
        __init__.py

├───security
        ir.model.access.csv

└───views
        coche_detalle_template.xml
        templates.xml
        views_cliente.xml
        views_coche.xml
        views_marca.xml
        views_vendedor.xml
        views_venta.xml


C:\Program Files\Odoo 17.0.20251123\custom_addons\practica11>
```

Cada modelo hereda de models.Model y define:

- _name
- _description
- Campos (fields.Char, Many2one, One2many, Many2many)
- Campos computados y relacionados

Ejemplo (Venta):

```
class ConcesionarioVenta(models.Model):
    _name = "concesionario.venta"
    _description = "Venta"

    name = fields.Char(default=lambda self: "VENTA-" +
fields.Date.today().strftime("%Y%m%d"))
    cliente_id = fields.Many2one("concesionario.cliente", required=True)
    coche_id = fields.Many2one("concesionario.coche", required=True)
    vendedor_id = fields.Many2one("concesionario.vendedor")
    precio_final = fields.Float(related="coche_id.precio", store=True)
```

Vistas (XML)

- **Tree**: listado de registros
- **Form**: edición detallada
- **Kanban**: visualización gráfica (coches)
- **Search**: filtros y búsquedas
- **Menus** y **Actions** para navegación

4.3.2 cliente.py

```
from odoo import models, fields

class ConcesionarioCliente(models.Model):
    _name = "concesionario.cliente"
    _description = "Clientes del concesionario"
    """
    Modelo ConcesionarioCliente: Representa a los clientes del concesionario.

    Requisitos cumplidos en este modelo:

    * 5 tablas o modelos como mínimo:
        - Este es uno de los modelos (Cliente), junto con Coche, Marca, etc.

    * Relaciones: Many2many
        - coche_favorito_ids: relación Many2many con 'concesionario.coche' usando tabla intermedia.

    * Campos con valores por defecto:
        - No tiene, pero se podrían añadir (ej. booleano activo=True con lambda).

    * ORM:
        - Se pueden usar search(), create(), browse()… para acceder y modificar registros de
clientes.

    * Herencia en algún modelo:
        - Este modelo no hereda de otros modelos, pero otros modelos como ConcesionarioCoche pueden
heredar de mail.thread.

    * Manejo de errores:
```

```
        - Se puede controlar que los clientes tengan nombre obligatorio (required=True) evitando
registros inválidos.

    * Campos relacionales computados y computados avanzados:
        - Se podrían añadir campos computados como el número de coches favoritos
(len(coche_favorito_ids)).
    """

    name = fields.Char(required=True)
    """Nombre del cliente (obligatorio, cumple manejo básico de errores de validación)"""

    email = fields.Char()
    """Email del cliente (campo opcional)"""

    coche_favorito_ids = fields.Many2many(
        "concesionario.coche",
        "cliente_coche_favorito_rel",  # Nombre de la tabla intermedia
        "cliente_id",                  # Columna que apunta a cliente
        "coche_id",                    # Columna que apunta a coche
        string="Coches favoritos"
    )
    """Relación Many2many con el modelo Coche: permite almacenar los coches favoritos del cliente"""
```

4.3.3 coche.py

```
from odoo import models, fields, api
from odoo.exceptions import ValidationError

class ConcesionarioCoche(models.Model):
    _name = "concesionario.coche"
    _description = "Coches del concesionario"
    """
    Modelo ConcesionarioCoche: Representa los coches del concesionario.

    Requisitos cumplidos en este modelo:

    * 5 tablas o modelos como mínimo
        - Este es uno de los modelos (junto con Cliente, Marca, etc.)

    * Relaciones:
        - Many2one: marca_id apunta a 'concesionario.marca'
        - Many2one: propietario_id apunta a 'concesionario.cliente'
        - Many2many: cliente_ids apunta a 'concesionario.cliente' para clientes interesados

    * Campos computados avanzados y almacenados en la BBDD:
        - antiguedad: se calcula a partir de fecha_fabricacion y se almacena (store=True)
        - estado: depende de antiguedad y se almacena (store=True)

    * Decoradores:
        - @api.depends: en _compute_antiguedad y _compute_estado
        - @api.constrains: en _check_precio para validar que el precio sea mayor a 0

    * Campos relacionales computados:
        - estado depende indirectamente de fecha_fabricacion y antiguedad

    * Campos con valores por defecto:
        - vendido: Booleano con default=False

    * ORM:
        - Se puede usar search(), create(), write(), browse() sobre 'concesionario.coche'

    * Herencia en algún modelo:
        - _inherit = ["mail.thread"] para agregar funcionalidades de seguimiento (tracking)
```

```
    * Manejo de errores:
        - _check_precio valida que precio sea mayor que 0 y lanza ValidationError
    """

    name = fields.Char(required=True, tracking=True)
    """Nombre del coche (obligatorio y rastreable)"""

    precio = fields.Float(required=True)
    """Precio del coche (obligatorio, validado con _check_precio)"""

    fecha_fabricacion = fields.Date()
    """Fecha de fabricación del coche"""

    vendido = fields.Boolean(default=False)
    """Indica si el coche está vendido, valor por defecto False"""

    marca_id = fields.Many2one(
        "concesionario.marca",
        string="Marca",
        required=True
    )
    """Relación Many2one con el modelo Marca"""

    cliente_ids = fields.Many2many(
        "concesionario.cliente",
        "cliente_coche_favorito_rel",
        "coche_id",
        "cliente_id",
        string="Clientes interesados"
    )
    """Relación Many2many con clientes interesados"""

    propietario_id = fields.Many2one(
        "concesionario.cliente",
        string="Propietario",
        ondelete="set null"
    )
    """Relación Many2one con el cliente propietario"""

    image = fields.Image(
        string="Imagen del Coche",
        max_width=800,
        max_height=600
    )
    """Imagen del coche"""

    antiguedad = fields.Integer(
        compute="_compute_antiguedad",
        store=True
    )
    """Campo computado: antigüedad del coche en años, almacenado en la base de datos"""

    estado = fields.Selection(
        [
            ("nuevo", "Nuevo"),
            ("usado", "Usado"),
        ],
        compute="_compute_estado",
        store=True
    )
    """Campo computado: estado del coche ('nuevo' o 'usado'), almacenado"""

    @api.depends("fecha_fabricacion")
```

```
    def _compute_antiguedad(self):
        """Calcula la antigüedad a partir de fecha_fabricacion"""
        for record in self:
            if record.fecha_fabricacion:
                record.antiguedad = fields.Date.today().year - record.fecha_fabricacion.year
            else:
                record.antiguedad = 0

    @api.depends("antiguedad")
    def _compute_estado(self):
        """Determina el estado del coche según la antigüedad"""
        for record in self:
            record.estado = "usado" if record.antiguedad > 1 else "nuevo"

    @api.constrains("precio")
    def _check_precio(self):
        """Valida que el precio sea mayor que 0"""
        for record in self:
            if record.precio <= 0:
                raise ValidationError("El precio debe ser mayor que 0")
```

4.3.4 marca.py

```
from odoo import models, fields

class ConcesionarioMarca(models.Model):
    _name = "concesionario.marca"
    _description = "Marcas de coches del concesionario"
    """
    Modelo ConcesionarioMarca: Representa las marcas de coches del concesionario.

    Requisitos cumplidos en este modelo:

    * 5 tablas o modelos como mínimo
        - Este es uno de los modelos (junto con Coche, Cliente, etc.)

    * Relaciones:
        - One2many: coche_ids conecta esta marca con los coches asociados
          (relación inversa de marca_id en ConcesionarioCoche)

    * ORM:
        - Se pueden usar search(), create(), browse(), write() para gestionar marcas

    * Manejo de errores:
        - El campo name es obligatorio (required=True), evitando registros inválidos

    * Campos relacionales computados / avanzados:
        - Aunque no tiene campos computados por ahora, coche_ids permite fácilmente agregarlos
          para contar coches por marca o filtrar coches por estado.
    """

    name = fields.Char(required=False) # False para pruebas, sino dara errores
    """Nombre de la marca (obligatorio)"""

    coche_ids = fields.One2many(
        "concesionario.coche",
        "marca_id",
        string="Coches"
    )
    """Relación One2many con los coches de esta marca"""
```

4.3.5 vendedor.py

```python
from odoo import models, fields

class ConcesionarioVendedor(models.Model):
    _name = "concesionario.vendedor"
    _description = "Vendedores del concesionario"
    """
    Modelo ConcesionarioVendedor: Representa a los vendedores del concesionario.

    Requisitos cumplidos en este modelo:

    * 5 tablas o modelos como mínimo
        - Este es uno de los modelos (junto con Cliente, Coche, Marca, Venta, etc.)

    * Relaciones:
        - One2many: venta_ids conecta al vendedor con las ventas realizadas
          (relación inversa de vendedor_id en ConcesionarioVenta)

    * ORM:
        - Se puede usar search(), create(), browse(), write() para gestionar vendedores

    * Manejo de errores:
        - El campo name es obligatorio (required=True), evitando registros inválidos

    * Campos relacionales computados:
        - Se podrían añadir fácilmente campos computados, por ejemplo
          para contar el número de ventas realizadas por el vendedor
    """

    name = fields.Char(required=True)
    """Nombre del vendedor (obligatorio)"""

    venta_ids = fields.One2many(
        "concesionario.venta",
        "vendedor_id"
    )
    """Relación One2many con las ventas realizadas por este vendedor"""
```

4.3.6 venta.py

```python
from odoo import models, fields, api

class ConcesionarioVenta(models.Model):
    _name = "concesionario.venta"
    _description = "Ventas realizadas en el concesionario"
    """
    Modelo ConcesionarioVenta: Representa las ventas realizadas en el concesionario.

    Requisitos cumplidos en este modelo:

    * 5 tablas o modelos como mínimo
        - Este es uno de los modelos (junto con Cliente, Coche, Marca, Vendedor, Venta)

    * Relaciones:
        - Many2one: cliente_id apunta al cliente de la venta
        - Many2one: coche_id apunta al coche vendido
        - Many2one: vendedor_id apunta al vendedor responsable
        - Relación inversa implícita con venta_ids en ConcesionarioVendedor (One2many)

    * Campos con valores por defecto:
        - name: valor por defecto generado dinámicamente usando lambda y la fecha actual
```

```
    * Campos relacionales computados / avanzados:
        - precio_final: campo relacionado (related) con coche_id.precio, almacenado en la BBDD
(store=True)

    * ORM:
        - Se puede usar search(), create(), browse(), write() para gestionar ventas

    * Manejo de errores:
        - Campos cliente_id y coche_id son obligatorios, evitando registros inválidos
    """

    name = fields.Char(
        default=lambda self: "VENTA-" + fields.Date.today().strftime("%Y%m%d")
    )
    """Identificador de la venta, generado automáticamente con la fecha actual"""

    cliente_id = fields.Many2one(
        "concesionario.cliente",
        required=True
    )
    """Relación Many2one con el cliente que realiza la compra (obligatorio)"""

    coche_id = fields.Many2one(
        "concesionario.coche",
        required=True
    )
    """Relación Many2one con el coche vendido (obligatorio)"""

    vendedor_id = fields.Many2one(
        "concesionario.vendedor"
    )
    """Relación Many2one con el vendedor responsable de la venta"""

    precio_final = fields.Float(
        related="coche_id.precio",
        store=True
    )
    """Precio final de la venta, campo relacionado con coche.precio, almacenado en la BBDD"""
```

4.3.7 Ir.model.access.csv

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_coche,coche,model_concesionario_coche,,1,1,1,1
access_marca,marca,model_concesionario_marca,,1,1,1,1
access_cliente,cliente,model_concesionario_cliente,,1,1,1,1
access_venta,venta,model_concesionario_venta,,1,1,1,1
access_vendedor,vendedor,model_concesionario_vendedor,,1,1,1,1
```

4.3.8 models/__init__.py

```
# -*- coding: utf-8 -*-

from . import marca
from . import coche
from . import cliente
from . import vendedor
from . import venta
```

4.3.9 controllers/controllers.py

```
from odoo import http
from odoo.http import request, Response
import json

class CocheAPI(http.Controller):
    """
    Clase CocheAPI: Implementa la API REST y vistas web para el módulo Concesionario.

    Requisitos cumplidos en este archivo:

    * ORM: env y métodos search(), create(), browse()…
      - En todas las funciones se usa request.env para acceder al modelo 'concesionario.coche'.
      - search(): se usa para listar coches en bootstrap_page() y get_coches().
      - browse(): se usa para acceder a un coche específico en coche_detalle().
      - create(): se usa para crear un coche nuevo en create_coche().

    * API-REST para CRUD
      - GET /api/coches en get_coches() para listar coches.
      - POST /api/coche en create_coche() para crear coches nuevos.

    * Bootstrap en alguna vista
      - La función bootstrap_page() renderiza 'practica11.bootstrap_example', que contiene Bootstrap.
      - La función coche_detalle() renderiza 'practica11.coche_detalle_template', que también usa
Bootstrap.

    * Manejo de errores
      - La función coche_detalle() verifica si el coche existe usando exists() y devuelve 404 si no
existe.
    """

    @http.route("/concesionario/boot", auth="public", website=True)
    def bootstrap_page(self):
        """
        Renderiza la página principal del concesionario usando Bootstrap.
```

```
        Cumplimiento de requisitos:
        - ORM: request.env["concesionario.coche"].sudo().search() para obtener todos los coches.
        - Bootstrap: renderiza 'practica11.bootstrap_example' con diseño Bootstrap.
        """
        coches = request.env["concesionario.coche"].sudo().search([])
        return request.render("practica11.bootstrap_example", {
            "coches": coches
        })

    @http.route("/concesionario/coche/<int:coche_id>", auth="public", website=True)
    def coche_detalle(self, coche_id):
        """
        Renderiza la vista de detalle de un coche específico.

        Cumplimiento de requisitos:
        - ORM: request.env["concesionario.coche"].sudo().browse(coche_id) para acceder al registro.
        - Manejo de errores: verifica si el coche existe con exists() y devuelve 404 si no existe.
        - Bootstrap: renderiza 'practica11.coche_detalle_template' para estética.
        """
        coche = request.env["concesionario.coche"].sudo().browse(coche_id)
        if not coche.exists():
            return request.not_found()
        return request.render("practica11.coche_detalle_template", {
            "coche": coche
        })

    @http.route("/api/coches", auth="public", methods=["GET"], csrf=False, type='http')
    def get_coches(self):
        """
        API REST GET para obtener todos los coches en formato JSON.

        Cumplimiento de requisitos:
        - ORM: request.env["concesionario.coche"].sudo().search() para obtener todos los coches.
        - API-REST: devuelve todos los coches en JSON.
        """
        coches = request.env["concesionario.coche"].sudo().search([])
        data = {
            "coches": [
                {"id": c.id, "name": c.name, "precio": c.precio}
                for c in coches
            ]
        }
        return Response(
            json.dumps(data),
            content_type='application/json',
            status=200
        )

    @http.route("/api/coche", auth="public", methods=["POST"], csrf=False, type='http')
    def create_coche(self, **data):
        """
        API REST POST para crear un nuevo coche.

        Cumplimiento de requisitos:
        - ORM: request.env["concesionario.coche"].sudo().create() para crear un registro nuevo.
        - API-REST: recibe JSON con name, precio y marca_id y devuelve el ID del coche creado.
        """
        coche = request.env["concesionario.coche"].sudo().create({
            "name": data.get("name"),
            "precio": float(data.get("precio", 0)),
            "marca_id": int(data.get("marca_id")),
        })
        return Response(
```

```
            json.dumps({"id": coche.id}),
            content_type='application/json',
            status=200
        )
```

4.3.10 Views/coche_detalle_template.xml

```
<odoo>
    <template id="coche_detalle_template" name="Detalle del Coche">
        <t t-call="web.layout">
            <!--
            Plantilla QWeb: coche_detalle_template

            Descripción general:
            - Muestra el detalle completo de un coche incluyendo imagen, precio, estado,
              antigüedad, marca y propietario.
            - Incluye diseño estético usando Bootstrap 5 y Bootstrap Icons.

            Conceptos Odoo/QWeb usados:

            1. t-call="web.layout"
               - Extiende la plantilla base de Odoo para mantener encabezado, footer y estilos
comunes.

            2. t-if / t-else
               - Condicionales de QWeb para mostrar la imagen del coche si existe,
                 o un placeholder si no hay imagen.

            3. t-att-src
               - Asigna dinámicamente el atributo 'src' del <img> con la imagen del coche
                 codificada en Base64.

            4. t-esc
               - Escapa el contenido para mostrar datos de manera segura:
                 - coche.name, coche.precio, coche.estado, coche.antiguedad, coche.marca_id.name,
                   coche.propietario_id.name

            5. t-att-class
               - Permite definir dinámicamente clases CSS según condición
                 - Por ejemplo, cambia el color del badge según si el coche es 'nuevo' o 'usado'.

            6. Bootstrap 5
               - Uso de contenedores (container), filas (row), columnas (col-*), tarjetas (card),
                 badges, botones (btn) y utilidades de espaciado y sombra (shadow, p-3, my-5, me-2).
               - Hace que la vista sea responsive y visualmente atractiva.

            7. Bootstrap Icons
               - Se incluyen para mostrar iconos en botones:
                 - bi-arrow-left-circle para "Volver al listado"
```

```
                - bi-cart-plus para "Comprar"

            8. Diseño responsive
               - Imagen y texto distribuidos en columnas col-lg-6 para escritorio
                 y flex-column para centrar verticalmente el contenido en pantallas grandes.

            9. Manejo de datos relacionales
               - coche.marca_id.name: Many2one hacia Marca
               - coche.propietario_id.name: Many2one hacia Cliente
               - Se usa condición para mostrar 'Sin propietario' si no tiene propietario.

            Requisitos cumplidos:

            * Bootstrap en la vista: Sí, diseño responsive y tarjetas con sombra
            * Manejo de relaciones Many2one: coche.marca_id y coche.propietario_id
            * Manejo de datos dinámicos con QWeb: t-esc, t-att-src, t-att-class
            * Condiciones dinámicas: t-if / t-else
            * Estética: badges de estado, espaciado, sombras, botones con iconos
            -->

            <!-- Incluir Bootstrap CSS y JS -->
            <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"/>
            <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

            <!-- Incluir Bootstrap Icons en el head -->
            <t t-set="head" t-value="head or ''">
                <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.1/font/bootstrap-icons.css"/>
            </t>

            <div class="container my-5">
                <div class="card shadow-lg rounded-4 p-3">
                    <div class="row g-3">
                        <!-- Imagen del coche -->
                        <div class="col-lg-6 text-center">
                            <t t-if="coche.image">
                                <img t-att-src="'%s' % (coche.image.decode() if
isinstance(coche.image, bytes) else coche.image)"
                                     class="img-fluid rounded-3 shadow-sm" style="max-height:400px;
object-fit:cover;" />
                            </t>
                            <t t-else="">
                                <img src="https://via.placeholder.com/800x400?text=Sin+imagen"
                                     class="img-fluid rounded-3 shadow-sm" style="max-height:400px;
object-fit:cover;" />
                            </t>
```

```
                    </div>

                    <!-- Información del coche -->
                    <div class="col-lg-6 d-flex flex-column justify-content-center">
                        <h2 class="card-title mb-3"><t t-esc="coche.name"/></h2>
                        <p class="card-text mb-2"><strong>Precio:</strong> <span class="text-
success fs-5"><t t-esc="coche.precio"/> €</span></p>
                        <p class="card-text mb-2"><strong>Estado:</strong>
                            <span t-att-class="'badge ' + ('bg-success' if coche.estado=='nuevo'
else 'bg-warning text-dark')">
                                <t t-esc="coche.estado.capitalize()"/>
                            </span>
                        </p>
                        <p class="card-text mb-2"><strong>Antigüedad:</strong>
                            <span class="badge bg-info text-dark"><t t-esc="coche.antiguedad"/>
años</span>
                        </p>
                        <p class="card-text mb-2"><strong>Marca:</strong> <t t-
esc="coche.marca_id.name"/></p>
                        <p class="card-text mb-2"><strong>Propietario:</strong>
                            <t t-esc="coche.propietario_id.name if coche.propietario_id else 'Sin
propietario'"/>
                        </p>

                        <!-- Botones -->
                        <div class="mt-4">
                            <a href="/concesionario/boot" class="btn btn-primary me-2">
                                <i class="bi bi-arrow-left-circle"></i> Volver al listado
                            </a>
                            <a href="#" class="btn btn-success">
                                <i class="bi bi-cart-plus"></i> Comprar
                            </a>
                        </div>
                    </div>
                </div>
            </div>
        </t>
    </template>
</odoo>
```

4.3.11 Views/ templates.xml

```
<odoo>
    <template id="bootstrap_example" name="Página Concesionario">
        <t t-call="web.layout">

            <!-- ======================================== -->
            <!-- Incluir Bootstrap 5 CSS y JS -->
            <!-- ======================================== -->
            <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"/>
            <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

            <!-- ======================================== -->
            <!-- HEADER -->
            <!-- Diseño con Bootstrap: fondo, texto centrado, padding -->
            <!-- ======================================== -->
            <header class="bg-primary text-white p-4 mb-4">
                <div class="container text-center">
                    <h1 class="mb-1">Concesionario de Coches</h1>
                    <p class="mb-0 lead">Encuentra tu coche ideal con nosotros</p>
                </div>
            </header>

            <!-- ======================================== -->
            <!-- CONTENIDO PRINCIPAL -->
            <!-- ======================================== -->
            <main class="container">
                <h2 class="mb-4 text-center">Listado de Coches Disponibles</h2>

                <!-- t-foreach: recorre la lista de coches enviada desde el controlador -->
                <div class="row">
                    <t t-foreach="coches" t-as="coche">

                        <!-- Card individual para cada coche -->
                        <div class="col-lg-4 col-md-6 mb-4 d-flex align-items-stretch">
                            <div class="card shadow-sm w-100">

                                <!-- ======================================== -->
                                <!-- Imagen del coche -->
                                <!-- t-if / t-else para mostrar imagen o placeholder -->
                                <!-- t-att-src para asignar la imagen codificada en Base64 -->
                                <!-- ======================================== -->
                                <t t-if="coche.image">
                                    <img t-att-src="'%s' %
(coche.image.decode() if isinstance(coche.image, bytes) else coche.image)"
                                         class="card-img-top" style="height:200px; object-fit:cover;"
/>
```

```
                                </t>
                                <t t-else="">
                                    <img src="https://via.placeholder.com/400x200?text=Sin+imagen"
                                        class="card-img-top" style="height:200px; object-fit:cover;"
/>
                                </t>

                                <!-- ======================================== -->
                                <!-- Información del coche -->
                                <!-- Uso de t-esc para mostrar datos seguros del modelo -->
                                <!-- ======================================== -->
                                <div class="card-body d-flex flex-column">
                                    <h5 class="card-title">
                                        <t t-esc="coche.name" />
                                    </h5>
                                    <p class="card-text mb-1">Precio: <t t-esc="coche.precio" />
€</p>

                                    <p class="card-text mb-1">Estado: <t t-esc="coche.estado" /></p>
                                    <p class="card-text text-muted mb-2">Propietario:
                                        <t t-esc="coche.propietario_id.name if coche.propietario_id
else 'Sin propietario'" />
                                    </p>

                                    <!-- ======================================== -->
                                    <!-- Botón para ver detalle -->
                                    <!-- t-att-href para generar la URL dinámica según coche.id -->
                                    <!-- ======================================== -->
                                    <a t-att-href="'/concesionario/coche/%s' % coche.id" class="btn
btn-success mt-auto">
                                        Ver detalles
                                    </a>
                                </div>
                            </div>
                        </div>

                    </t>
                </div>
            </main>

            <!-- ======================================== -->
            <!-- FOOTER -->
            <!-- ======================================== -->
            <footer class="bg-dark text-white text-center py-3 mt-4">
                <div class="container">
                    <small>2026 Concesionario de Coches</small>
                </div>
            </footer>
```

```
        </t>
    </template>
</odoo>
```

4.3.12 Views/ views_cliente.xml

```
<odoo>
    <!-- ========================================== -->
    <!-- VISTA TREE DEL MODELO CLIENTE -->
    <!-- Requisito: Vistas tree y form para todos los modelos -->
    <!-- ========================================== -->
    <record id="view_cliente_tree" model="ir.ui.view">
        <field name="name">cliente.tree</field>
        <field name="model">concesionario.cliente</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Campos mostrados en la lista -->
                <field name="name" />                  <!-- Nombre del cliente -->
                <field name="email" />                 <!-- Email del cliente -->
                <field name="coche_favorito_ids" />    <!-- Relación Many2many con coches favoritos -
->
            </tree>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA FORM DEL MODELO CLIENTE -->
    <!-- Requisito: Vistas tree y form para todos los modelos -->
    <!-- ========================================== -->
    <record id="view_cliente_form" model="ir.ui.view">
        <field name="name">cliente.form</field>
        <field name="model">concesionario.cliente</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group>
                        <!-- Campos editables -->
                        <field name="name" />
                        <field name="email" />
                        <field name="coche_favorito_ids" widget="many2many_tags" /> <!-- Many2many
con widget -->
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- ========================================== -->
```

```
    <!-- VISTA SEARCH DEL MODELO CLIENTE -->
    <!-- Requisito: Búsqueda y filtros básicos -->
    <!-- ========================================== -->
    <record id="view_cliente_search" model="ir.ui.view">
        <field name="name">cliente.search</field>
        <field name="model">concesionario.cliente</field>
        <field name="arch" type="xml">
            <search>
                <field name="name" />  <!-- Buscar por nombre -->
                <field name="email" /> <!-- Buscar por email -->
            </search>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- ACTION PARA ABRIR LA VISTA DE CLIENTES -->
    <!-- Requisito: Acciones de ventana -->
    <!-- ========================================== -->
    <record id="action_cliente" model="ir.actions.act_window">
        <field name="name">Clientes</field>
        <field name="res_model">concesionario.cliente</field>
        <field name="view_mode">tree,form</field> <!-- Permite abrir en Tree y Form -->
    </record>

    <!-- ========================================== -->
    <!-- MENU PARA ACCEDER A CLIENTES -->
    <!-- Requisito: Navegación en el menú principal -->
    <!-- ========================================== -->
    <menuitem id="menu_cliente" name="Clientes" parent="menu_concesionario" action="action_cliente"
/>

</odoo>
```

4.3.13 Views/ views_coche.xml

```
<odoo>

    <!-- ========================================== -->
    <!-- VISTA TREE DEL MODELO COCHE -->
    <!-- Requisito: Vista tree -->
    <!-- ========================================== -->
    <record id="view_coche_tree" model="ir.ui.view">
        <field name="name">coche.tree</field>
        <field name="model">concesionario.coche</field>
        <field name="arch" type="xml">
            <tree>
                <!-- Campos -->
                <field name="image" widget="image" class="oe_avatar"/> <!-- Imagen (campo Image) -->
```

```
                <field name="name"/>                                    <!-- Nombre del coche -->
                <field name="marca_id"/>                                <!-- Relación Many2one con
Marca -->
                <field name="precio"/>                                  <!-- Precio -->
                <field name="propietario_id"/>                          <!-- Relación Many2one con
Cliente -->
                <field name="estado"/>                                   <!-- Campo computado estado
-->
            </tree>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA FORM DEL MODELO COCHE -->
    <!-- Requisito: Vista form, campos relacionales -->
    <!-- ========================================== -->
    <record id="view_coche_form" model="ir.ui.view">
        <field name="name">coche.form</field>
        <field name="model">concesionario.coche</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group>
                        <field name="name"/>
                        <field name="marca_id"/>                        <!-- Many2one -->
                        <field name="precio"/>
                        <field name="fecha_fabricacion"/>
                        <field name="propietario_id"/>                  <!-- Many2one -->
                        <field name="cliente_ids" widget="many2many_tags"/>    <!-- Many2many -->
                        <field name="image" widget="image" class="oe_avatar"/> <!-- Imagen -->
                        <field name="estado" readonly="1"/>             <!-- Campo computado -
->
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA KANBAN DEL MODELO COCHE -->
    <!-- Requisito: Vista Kanban mínima para un modelo -->
    <!-- ========================================== -->
    <record id="view_coche_kanban" model="ir.ui.view">
        <field name="name">coche.kanban</field>
        <field name="model">concesionario.coche</field>
        <field name="arch" type="xml">
            <kanban>
                <templates>
```

```
                    <t t-name="kanban-box">
                        <div class="oe_kanban_card">
                            <t t-if="record.image">
                                <img t-att-src="'%s' % record.image"
class="oe_kanban_image"/>
                            </t>
                            <t t-else="">
                                <img src="https://via.placeholder.com/150x100?text=Sin+imagen"
class="oe_kanban_image"/>
                            </t>
                            <strong><field name="name"/></strong>
                            <div><field name="marca_id"/></div>
                            <div><field name="precio"/> €</div>
                        </div>
                    </t>
                </templates>
            </kanban>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA SEARCH DEL MODELO COCHE -->
    <!-- Requisito: búsqueda y filtros -->
    <!-- ========================================== -->
    <record id="view_coche_search" model="ir.ui.view">
        <field name="name">coche.search</field>
        <field name="model">concesionario.coche</field>
        <field name="arch" type="xml">
            <search>
                <field name="name"/>
                <field name="marca_id"/>
                <filter string="Disponible" name="available" domain="[('estado','=','nuevo')]"/>
            </search>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- ACTION PARA ABRIR VISTA DE COCHES -->
    <!-- Requisito: abrir vistas tree, form y kanban -->
    <!-- ========================================== -->
    <record id="action_coche" model="ir.actions.act_window">
        <field name="name">Coches</field>
        <field name="res_model">concesionario.coche</field>
        <field name="view_mode">tree,form,kanban</field>
    </record>

    <!-- ========================================== -->
    <!-- MENÚS -->
```

```
    <!-- Requisito: navegación en menú principal -->
    <!-- ========================================== -->

    <menuitem id="menu_coche" name="Coches" parent="menu_concesionario" action="action_coche"/>

</odoo>
```

4.3.14 Views/ views_marca.xml

```
<odoo>
    <!-- ========================================== -->
    <!-- VISTA TREE DEL MODELO MARCA -->
    <!-- Requisito: Vista tree -->
    <!-- ========================================== -->
    <record id="view_marca_tree" model="ir.ui.view">
        <field name="name">marca.tree</field>
        <field name="model">concesionario.marca</field>
        <field name="arch" type="xml">
            <tree>
                <field name="name"/>      <!-- Nombre de la marca -->
            </tree>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA FORM DEL MODELO MARCA -->
    <!-- Requisito: Vista form, relación One2many con coches -->
    <!-- ========================================== -->
    <record id="view_marca_form" model="ir.ui.view">
        <field name="name">marca.form</field>
        <field name="model">concesionario.marca</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group>
                        <field name="name"/>        <!-- Nombre de la marca -->
                        <field name="coche_ids"/>   <!-- One2many: todos los coches de esta marca -->
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA SEARCH DEL MODELO MARCA -->
    <!-- Requisito: Búsqueda y filtros -->
    <!-- ========================================== -->
    <record id="view_marca_search" model="ir.ui.view">
```

```
        <field name="name">marca.search</field>
        <field name="model">concesionario.marca</field>
        <field name="arch" type="xml">
            <search>
                <field name="name"/>  <!-- Permite buscar por nombre de marca -->
            </search>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- ACTION PARA ABRIR VISTA DE MARCAS -->
    <!-- Requisito: Abrir vistas tree y form -->
    <!-- ========================================== -->
    <record id="action_marca" model="ir.actions.act_window">
        <field name="name">Marcas</field>
        <field name="res_model">concesionario.marca</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- ========================================== -->
    <!-- MENÚ PARA EL MODELO MARCA -->
    <!-- Requisito: Navegación en el menú principal -->
    <!-- ========================================== -->
     <menuitem id="menu_concesionario" name="Concesionario"/>
    <menuitem id="menu_marca" name="Marcas" parent="menu_concesionario" action="action_marca"/>
</odoo>
```

4.3.15 Views/ views_vendedor.xml

```
<odoo>
    <!-- ========================================== -->
    <!-- VISTA TREE DEL MODELO VENDEDOR -->
    <!-- Requisito: Vista tree -->
    <!-- ========================================== -->
    <record id="view_vendedor_tree" model="ir.ui.view">
        <field name="name">vendedor.tree</field>
        <field name="model">concesionario.vendedor</field>
        <field name="arch" type="xml">
            <tree>
                <field name="name"/> <!-- Nombre del vendedor -->
            </tree>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA FORM DEL MODELO VENDEDOR -->
    <!-- Requisito: Vista form, relación One2many con ventas -->
    <!-- ========================================== -->
    <record id="view_vendedor_form" model="ir.ui.view">
        <field name="name">vendedor.form</field>
        <field name="model">concesionario.vendedor</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
```

```
                    <group>
                        <field name="name"/>        <!-- Nombre del vendedor -->
                        <field name="venta_ids"/>   <!-- One2many: todas las ventas realizadas por
este vendedor -->
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA SEARCH DEL MODELO VENDEDOR -->
    <!-- Requisito: Búsqueda de registros -->
    <!-- ========================================== -->
    <record id="view_vendedor_search" model="ir.ui.view">
        <field name="name">vendedor.search</field>
        <field name="model">concesionario.vendedor</field>
        <field name="arch" type="xml">
            <search>
                <field name="name"/>  <!-- Permite buscar por nombre del vendedor -->
            </search>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- ACTION PARA ABRIR VISTA DE VENDEDORES -->
    <!-- Requisito: Abrir vistas tree y form -->
    <!-- ========================================== -->
    <record id="action_vendedor" model="ir.actions.act_window">
        <field name="name">Vendedores</field>
        <field name="res_model">concesionario.vendedor</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- ========================================== -->
    <!-- MENÚ PARA EL MODELO VENDEDOR -->
    <!-- Requisito: Navegación en el menú principal -->
    <!-- ========================================== -->
    <menuitem id="menu_vendedor" name="Vendedores" parent="menu_concesionario"
action="action_vendedor"/>
</odoo>
```

4.3.16 Views/views_venta.xml

```
<odoo>
    <!-- ========================================== -->
    <!-- VISTA TREE DEL MODELO VENTA -->
    <!-- Requisito: Vista tree -->
    <!-- ========================================== -->
    <record id="view_venta_tree" model="ir.ui.view">
        <field name="name">venta.tree</field>
        <field name="model">concesionario.venta</field>
        <field name="arch" type="xml">
            <tree>
                <field name="name"/>          <!-- Nombre de la venta -->
                <field name="cliente_id"/>    <!-- Many2one a Cliente -->
                <field name="coche_id"/>      <!-- Many2one a Coche -->
                <field name="vendedor_id"/>   <!-- Many2one a Vendedor -->
                <field name="precio_final"/>  <!-- Campo computado relacionado al coche -->
            </tree>
        </field>
    </record>
```

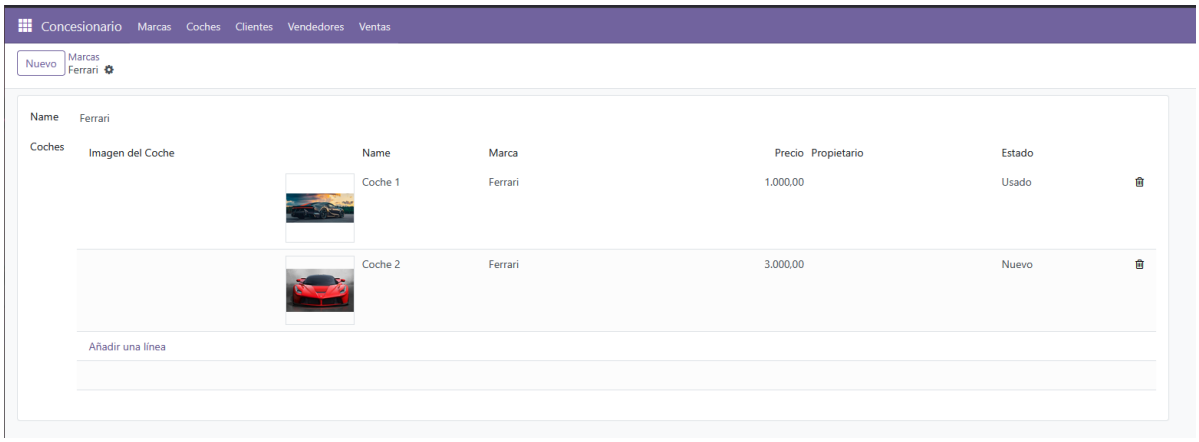
```
    <!-- ========================================== -->
    <!-- VISTA FORM DEL MODELO VENTA -->
    <!-- Requisito: Vista form, relaciones Many2one, campo computado -->
    <!-- ========================================== -->
    <record id="view_venta_form" model="ir.ui.view">
        <field name="name">venta.form</field>
        <field name="model">concesionario.venta</field>
        <field name="arch" type="xml">
            <form>
                <sheet>
                    <group>
                        <field name="name" readonly="1"/>          <!-- Nombre generado
automáticamente -->
                        <field name="cliente_id"/>                <!-- Selección de cliente -->
                        <field name="coche_id"/>                  <!-- Selección de coche -->
                        <field name="vendedor_id"/>               <!-- Selección de vendedor -->
                        <field name="precio_final" readonly="1"/> <!-- Campo relacionado al coche
(precio) -->
                    </group>
                </sheet>
            </form>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- VISTA SEARCH DEL MODELO VENTA -->
    <!-- Requisito: Búsqueda de registros -->
    <!-- ========================================== -->
    <record id="view_venta_search" model="ir.ui.view">
        <field name="name">venta.search</field>
        <field name="model">concesionario.venta</field>
        <field name="arch" type="xml">
            <search>
                <field name="name"/>
                <field name="cliente_id"/>
                <field name="coche_id"/>
                <field name="vendedor_id"/>
            </search>
        </field>
    </record>

    <!-- ========================================== -->
    <!-- ACTION PARA ABRIR VISTA DE VENTAS -->
    <!-- Requisito: Abrir vistas tree y form -->
    <!-- ========================================== -->
    <record id="action_venta" model="ir.actions.act_window">
        <field name="name">Ventas</field>
        <field name="res_model">concesionario.venta</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- ========================================== -->
    <!-- MENÚ PARA EL MODELO VENTA -->
    <!-- Requisito: Navegación en el menú principal -->
    <!-- ========================================== -->
    <menuitem id="menu_venta" name="Ventas" parent="menu_concesionario" action="action_venta"/>
</odoo>
```

4.3.17 __manifest__.py

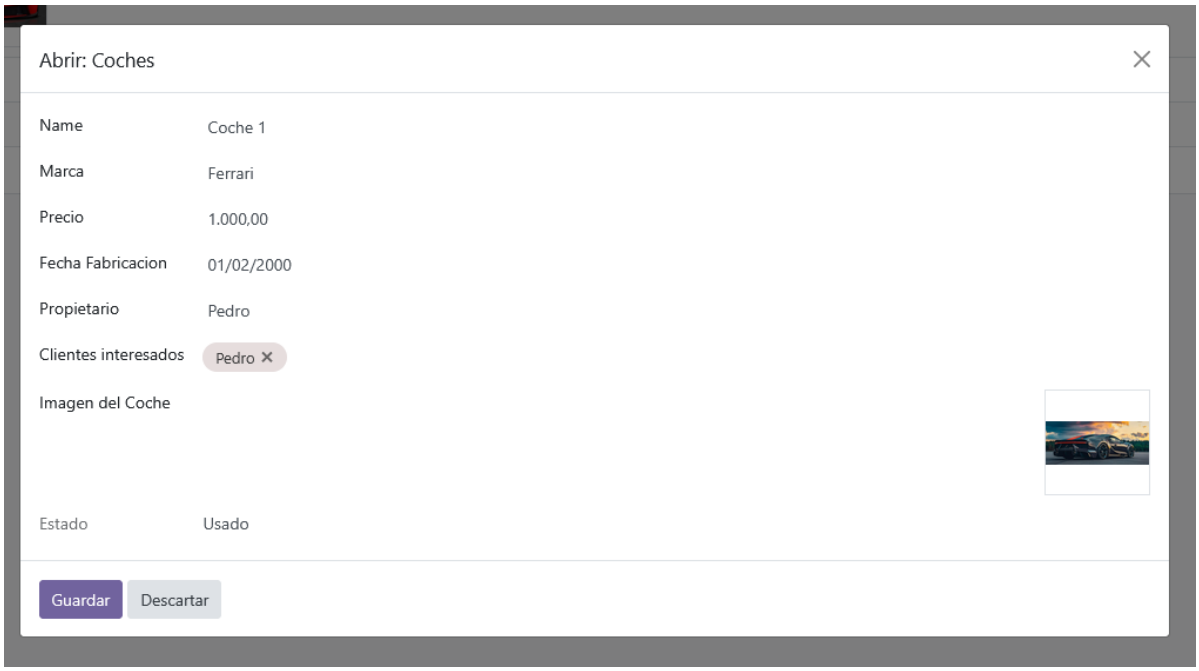
```
{
    "name": "Concesionario de Coches (Practica 11)",
    "version": "17.0.1.0.0",
    "category": "Tools",
    "summary": "Gestión de concesionario de coches",
    "author": "Alumno",
    "depends": ["base", "web", "mail"],
    "data": [
        "security/ir.model.access.csv",
        'views/views_marca.xml',
        'views/views_coche.xml',
        'views/views_cliente.xml',
        'views/views_vendedor.xml',
        'views/views_venta.xml',
        "views/templates.xml",
        "views/coche_detalle_template.xml"
    ],
    "application": True,
    'installable': True,
}
```

4.4 Pruebas y funcionamiento

Registramos algunos cohes, recomiendo añadir alguna imagen del coche para luego poder visualizar bien el coche en la API:

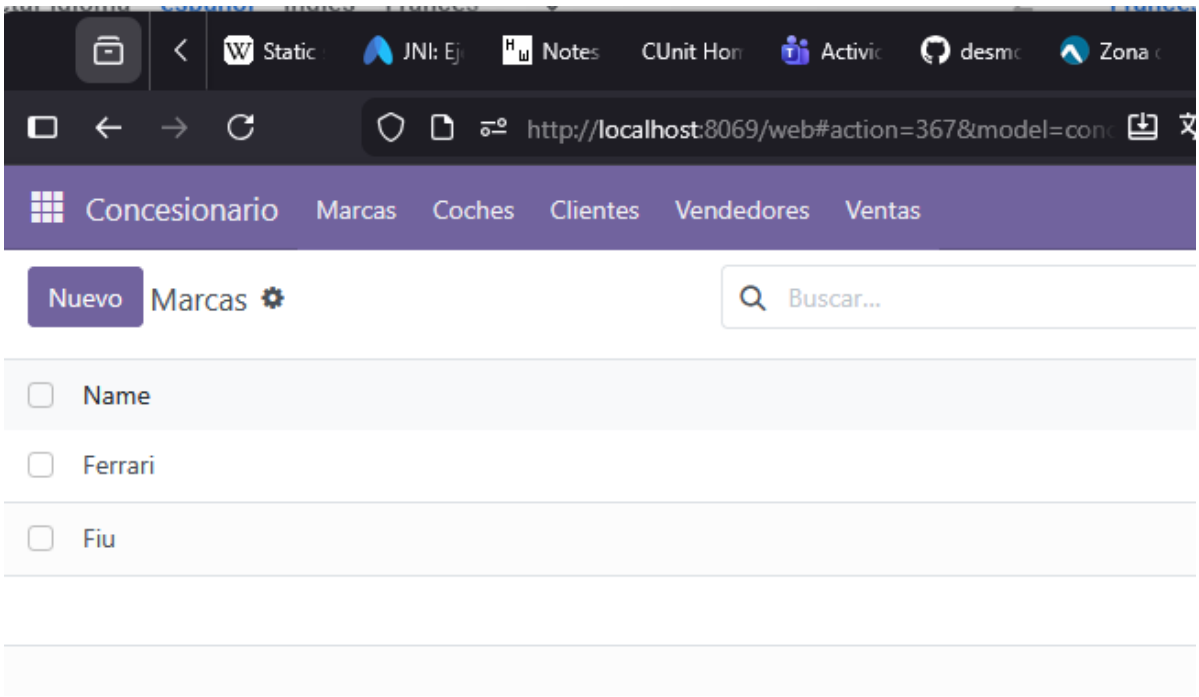


Estos coches tienen los siguientes datos:

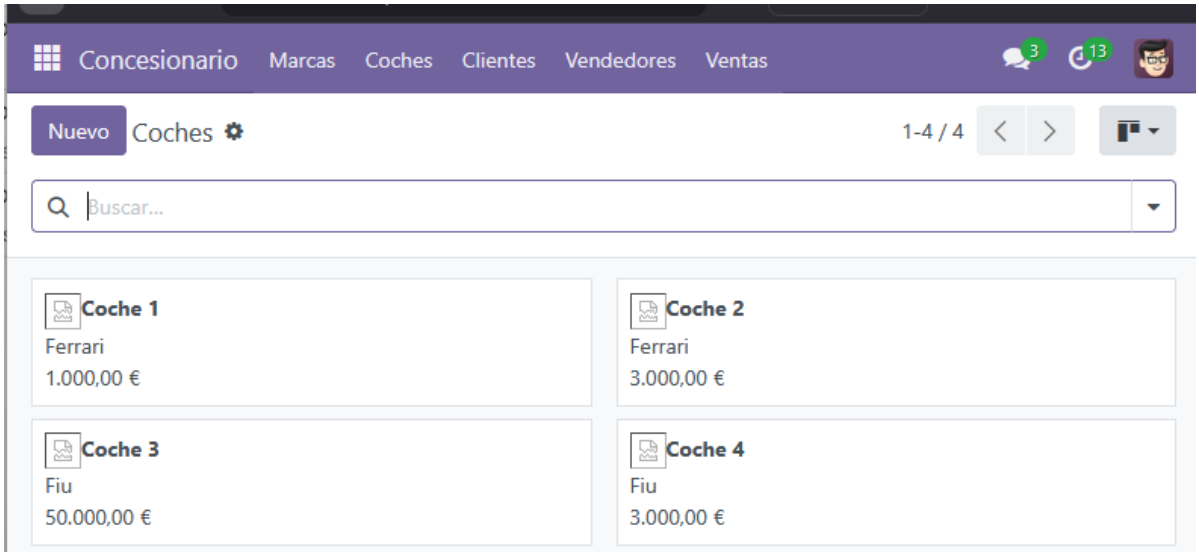


Algunos los podremos crear sobre la marcha al añadir el coche al concesionario. Otros campos como las Ventas solo tendrán sentido crearlos después del coche.

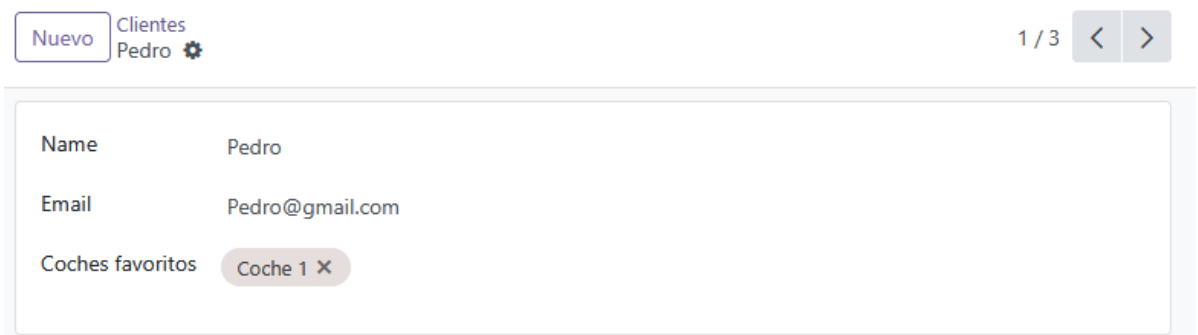
En el apartado de marcas podemos ver todas las marcas para nuestros autos:



Los coches también podremos visualizarlos en una vista Kanban:

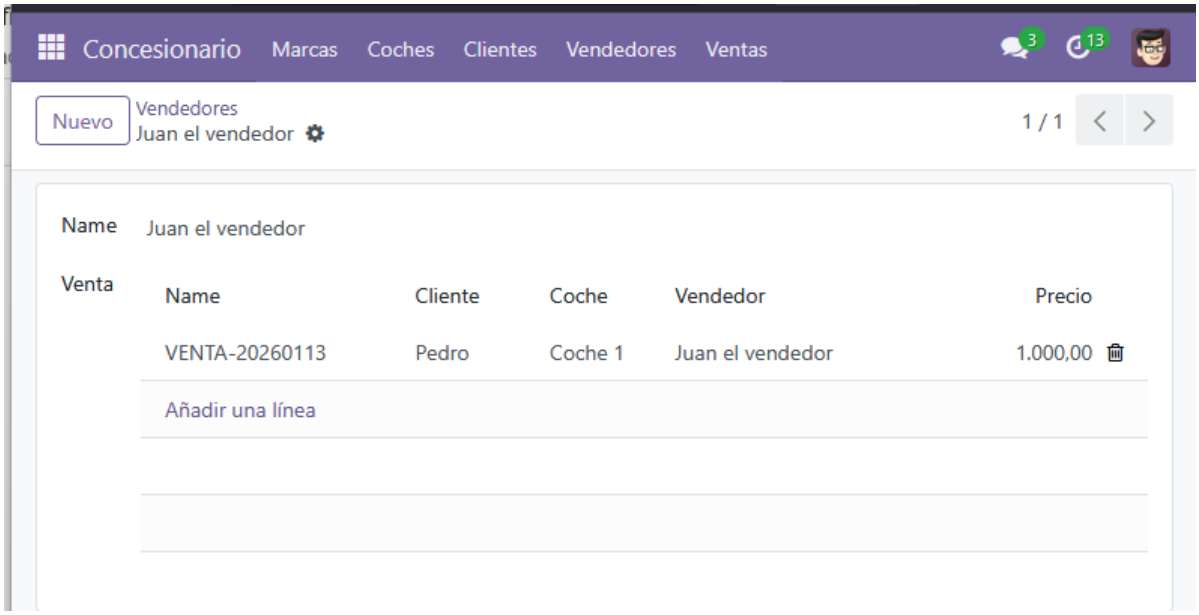


Ahora el tema de los clientes:

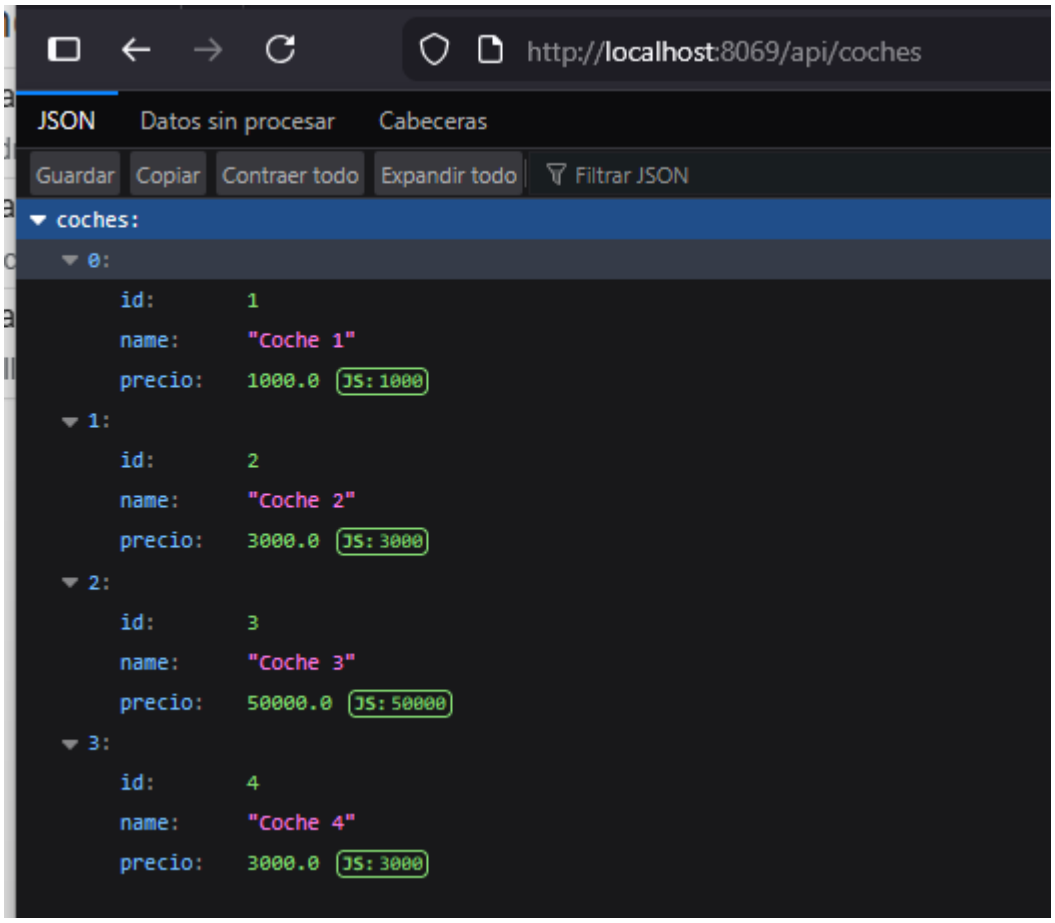


Un cliente tiene asociado sus coches favoritos, podemos hacer tantos clientes como necesitemos.

A la hora de hacer las ventas, necesitamos que esten definidor el vendedor, el cliente y el coche para poder hacer una venta, asi que haremos primero a nuestro vendedor y luego a nuestra venta:

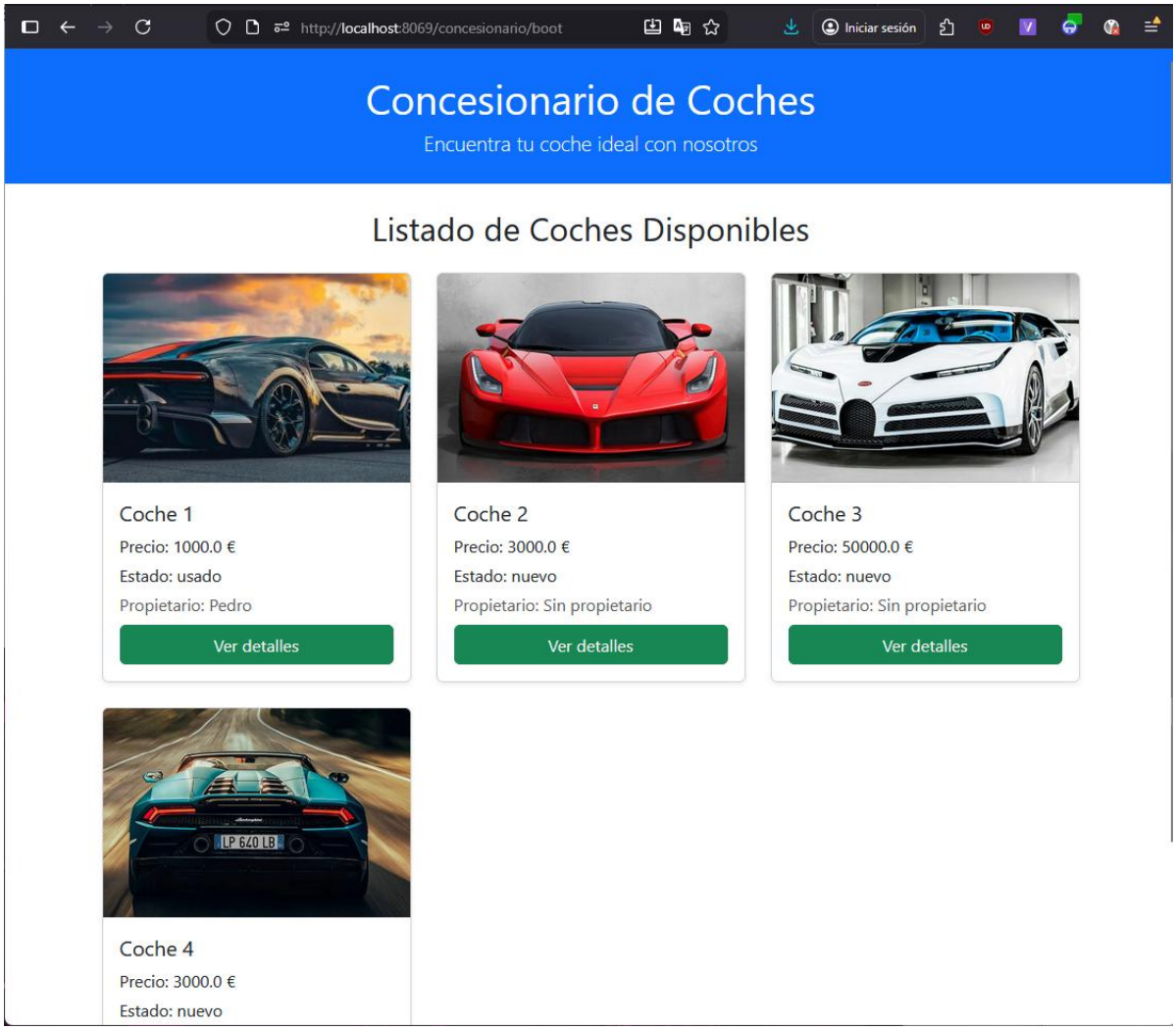


Para acceder a la API tenemos 4 end point, la primera nos permite obtener todos los coches a través de un GET: http://localhost:8069/api/coches



También podemos usar la versión http://localhost:8069/api/coche para crear un coche nuevo, pero es necesario usar una petición POST.
Si queremos acceder a la información de un coche y conocemos su ID podemos ir a http://localhost:8069/concesionario/coche/1, donde cambiaremos el 1 por el id que queramos:

Si accedemos a http://localhost:8069/concesionario/boot podemos visualizar una pagina web autogenerada y dinámica con todos los coches, según añadamos o eliminemos aparecerán o desaparecerán de aquí. Además podemos ver datos como el propietario y otros.



4.5 Despliegue de la aplicación

En este caso el despliegue se realizó en un servidor local

5 Manual

5.1 Manual de usuario

Una vez realizada la instalación del módulo podremos encontrar en nuestro Menu la opción de concesionario. En la parte superior se puede visualizar un conjunto de opciones, Coche, ventas, clientes, etc. Deberemos acceder a cada uno y registrar los datos que nos interese.

5.2 Manual de instalación

Primero deberemos tener el módulo que hemos creado. Instalar Odoo 17 en la maquina deseada y cambiar el path de los addons, a posterior deberemos ingresar nuestro nuevo módulo en el path de addons personalizados. Iremos al apartado de configuración de Odoo y activaremos el modo

desarrollador. Posteriormente iremos al apartado de aplicaciones en el Menú y alli usaremos el buscados para buscar nuestro modulo por el nombre. *Debe reiniciar Odoo una vez haya metido el módulo en la carpeta de addons para que el módulo sea visible por Odoo*.

6 Conclusiones y posibles ampliaciones

No hubo muchas dificultades en el desarrollo, las mayorías de las dificultades encontradas fueron en la instalación y en el apartado de pruebas. En algún punto la maquina de Odoo dio un fallo critico por alguna razón y ya no consegui hacerla funcionar. Aun sacando el módulo y etc.

Como futuras ampliaciones se podría pulir el apartado de la API y meter algunas opciones más para obtener más datos de otros modulos, o en otros formatos. Tambien se podría añadir algún mecanismo que permita contar todas las ventas realizadas al dia, mes y año u otras similares.

7 Bibliografía

- Manuales de clase.
- Otros trabajos realizados.
- Chapter 1 - Theming — Odoo 19.0 documentation
- https://www.cybrosys.com/blog/bootstrap-basics
- Videos de youtube