

Módulo Gestión de Inventario, Tickets y Clientes

2º DAM
Álvaro Pavón Martínez

Tabla de contenido

Descripción del Problema.....	3
Programación de la aplicación	3
Lista de Ficheros de Código Fuente y Breve Descripción	4
Documentación Técnica Detallada de cada Archivo	5
__manifest__.py	5
__init__.py	6
Ir.model.access.csv	7
Modelo Hardware (hardware.py)	7
Modelo Ticket (ticket.py)	8
Modelo ticket line (ticket_line.py)	10
Modelo Mantenimiento (mantenimiento.py)	11
Extensión del módulo Cliente (cliente.py)	11
Vistas y Menús (Archivos XML)	12
Manual de Usuario	20
Instalación y configuración inicial	20
Requisitos previos	20
Pasos para la instalación.....	20
Navegación en la aplicación.....	21
Inicio de sesión y acceso	21
Menú principal y submenús.....	22
Gestión del inventario (hardware)	22
Creación de un registro de hardware	22
Edición y consulta	23
Gestión de tickets	24
Registro de incidencias y comprobantes.....	24
Asociaciones en el ticket.....	25
Gestión de mantenimientos	25
Registro de mantenimientos	25
Consulta y búsqueda	26
Gestión de clientes	26
Registro y consulta del historial de tickets.....	26
Uso de reportes	27
Generación del reporte PDF	27
Webgrafía.....	28
Conclusión	29
Link de descarga	29

Descripción del Problema

El proyecto consiste en desarrollar un módulo para Odoo titulado “Gestión de Inventario, Tickets, Mantenimientos y Clientes”. La aplicación esta diseñada para solucionar varias problemáticas que enfrentan las empresas:

- **Control del Inventario:**
Permite registrar y mantener actualizado el stock de productos o equipos (hardware), incluyendo información como nombre, descripción, modelo, precio, marca, imagen y cantidad disponible, y asignar un responsable a cada producto si fuese necesario.
- **Gestion de Tickets:**
Los tickets se generan para entregar un comprobante a los clientes una vez realizado el servicio o la venta. Además, se mantiene un registro en el sistema que diferencia los tickets pagados de los pendientes de pago, facilitando el control de la facturación.
- **Asignacion de mantenimientos:**
Permite registrar y asignar mantenimientos o servicios a los productos del inventario, lo que facilita el seguimiento de intervenciones y reparaciones.
- **Registro de clientes:**
Los clientes se registran cuando se genera un ticket, es decir, se crean en el sistema al producirse una venta o servicio. Este registro se utiliza para llevar un historial de clientes que han comprado para dar un seguimiento a futuras incidencias.

La solución integrada mejora la eficiencia operativa, asegura un control riguroso del stock, facilita la entrega de comprobantes a los clientes y permite mantener un historial detallado de compras y servicios.

Al final de esta documentación encontraras el Link de descarga del modulo.

Programación de la aplicación

La aplicación se organiza en varios componentes clave:

- **Archivos de Configuración:**
 - `__manifest__.py`: Define la identidad del módulo, dependencias, archivos a cargar y metadatos esenciales (nombre, versión, autor, etc.).
 - `__init__.py`: Importa los submódulos que contienen la lógica y definición de modelos.
- **Modelos y lógica de negocio (Archivos .py):**
 - `Hardware.py`: Modelo hardware para registrar y gestionar los productos del inventario.
 - `Ticket.py`: Modelo ticket para entregar comprobantes a los clientes y para llevar el control de los tickets pagados y pendientes.

- ***Ticket_line.py***: Gestiona las líneas de producto en cada ticket, controlando y ajustando el stock
- ***Mantenimiento.py***: Modelo Mantenimiento (custom) para registrar servicios realizados en el inventario
- ***Cliente.py***: Extiende el modelo res.partner para registrar a los clientes que han generado un ticket (clientes que han comprado) y llevar su historial
- **Vistas y Menús (Archivos .xml)**:
 - Se definen vistas (formulario, lista y búsqueda) para cada modelo
 - Se configuran menús de navegación para acceder a las áreas de Inventario, Tickets, Mantenimientos y Clientes
 - Se establecen plantillas y layouts para generación de reportes PDF, que se entregan a los clientes como comprobantes
- **Seguridad**
 - Archivo CSV (ir.model.access.csv) definen los permisos de acceso a cada modelo, protegiendo la integridad de los datos

Lista de Ficheros de Código Fuente y Breve Descripción

- **__manifest__.py**:
Funcionalidad: Configuración e identificación del módulo, definición de dependencias y datos a cargar.
Cometido: Permitir que Odoo reconozca, instale y configure el módulo correctamente
- **__init__.py**:
Funcionalidad: Importa los módulos Python que contienen los modelos y la lógica de negocio.
Cometido: Garantizar la carga inicial de todos los componentes del módulo
- **Hardware.py**:
Funcionalidad: Define el modelo Hardware para registrar productos del inventario
Cometido: Gestionar datos esenciales (nombre, descripción, modelo, precio, marca, imagen, stock y responsable si fuese necesario) y calcular el número de tickets asociados
- **Ticket.py**:
Funcionalidad: Define el modelo Ticket para registrar ventas.
Cometido: Documentar todas las ventas y entregar comprobantes a los clientes y mantener un registro de los tickets pagados y pendientes
- **Ticket_line.py**:
Funcionalidad: Gestiona las líneas de producto en un ticket.
Cometido: Permitir la selección de productos, definir cantidades, calcular totales y ajustar el stock automáticamente.
- **Mantenimiento.py**:
Funcionalidad: Define el modelo Mantenimiento (custom) para registrar servicios realizados.

Cometido: Permitir la asignación de mantenimientos a uno o varios productos del inventario

- **Cliente.py:**
Funcionalidad: Extiende el modelo res.partner para incorporar una lista de tickets por cliente
Cometido: Registrar a los clientes una vez que generan un ticket, llevando un historial de clientes que han comprado
- **Archivos XML:**
Funcionalidad: Configuran vistas, menús y reportes PDF.
Cometido: Proveer una interfaz de usuario organizada e intuitiva
- **Archivo de seguridad (ir.model.access.csv)**
Funcionalidad: Define los permisos de acceso a cada modelo.
Cometido: Garantizar la integridad y seguridad de los datos

Documentación Técnica Detallada de cada Archivo

__manifest__.py

- **Propósito:**
Establece la identidad y configuración del modulo
- **Elementos clave:**
 - Name, summary, description: Detallan la gestión del inventario, tickets (para comprobantes y gestión de pagos), mantenimientos y clientes.
 - Depends: Especifica la dependencia de los módulos base y producto
 - Data: Lista archivos XML, CSV (incluyendo ir.model.access.csv) a cargar

```

C:\Users > alvar > Odoo > addons > gestion_inventario > Modulo-Odoo > gestion_hardware > _manifest.py
1
2 {
3     'name': "gestion_hardware",
4     'summary': "Gestión de Hardware, Tickets y Mantenimientos",
5     'description': """
6         Proyecto creado como solución integral para la gestión de inventario de hardware, clientes, reparaciones y tickets.
7         ## Funcionalidades
8         Este módulo ofrece una solución completa para:
9         - **Inventario de Hardware:**
10            Registrar y mantener un inventario detallado con información como nombre, descripción, modelo, precio, stock, marca e imagen.
11         - **Gestión de Clientes:**
12            Administrar la información de clientes y sus interacciones.
13         - **Reparaciones y Mantenimientos:**
14            Registrar reparaciones realizadas en el hardware, actualizando automáticamente el stock y asociándolas a los tickets correspondientes.
15         - **Tickets de Soporte:**
16            Crear y administrar tickets que integran líneas de productos y reparaciones, con vistas personalizadas y la posibilidad de generar **PDF** para imprimirlos.
17         ## Beneficios
18         Ideal para empresas que necesitan:
19         - Centralizar la gestión de sus activos tecnológicos.
20         - Optimizar los procesos de atención al cliente y mantenimiento.
21         - Tener un control detallado y visual de su inventario y reparaciones.
22         ¡Una solución completa para mantener tu negocio en perfecto funcionamiento!
23     """,
24     'author': "Alvaro",
25     'website': "https://www.alvaropavon.com",
26     'category': 'Inventario',
27     'version': '1.0',
28     'license': 'LGPL-3',
29     'depends': ['base', 'product'],
30     'data': [
31         'security/ir.model.access.csv',      # Permisos de acceso
32         'views/hardware_views.xml',          # Vistas para Hardware
33         'views/ticket_views.xml',            # Vistas para Ticket (incluye referencia a la vista de líneas)
34         'views/mantenimiento_views.xml',     # Vistas para Mantenimiento (modelo custom)
35         'views/cliente_views.xml',           # Vistas extendidas para Clientes (res.partner)
36         'views/menu_views.xml',             # Menú de navegación
37         'views/report_layout.xml',           # Layout para reportes
38         'reports/report_ticket.xml',         # Reporte PDF para Ticket
39     ],
40     'images': ['static/description/icon.png'], # Icono de la aplicación
41     'installable': True,
42     'application': True,
43 }

```

__init__.py

- **Propósito:**
Importar todos los modelos y la lógica
- **Elementos clave:**
 - Importa: hardware, cliente, ticket_line, ticket y mantenimiento

```

__init__.py X
__init__.py
1  # -*- coding: utf-8 -*-
2
3  from . import controllers
4  from . import models
5

```

Ir.model.access.csv

- **Propósito:**
Define los permisos de acceso a cada modelo
- **Elementos clave:**
 - Permite controlar quien puede leer, crear, actualizar o eliminar registros, protegiendo la integridad de los datos

```
ir.model.access.csv X
security > ir.model.access.csv > data
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 access_gestion_hardware,Gestión de Hardware,gestion_hardware.model_gestion_hardware,base.group_user,1,1,1,1
3 access_gestion_ticket,Gestión de Ticket,gestion_hardware.model_gestion_ticket,base.group_user,1,1,1,1
4 access_gestion_ticket_line,Gestión de Línea de Ticket,gestion_hardware.model_gestion_ticket_line,base.group_user,1,1,1,1
5 access_gestion_mantenimiento,Gestión de Mantenimiento,gestion_hardware.model_x_gestion_mantenimiento,base.group_user,1,1,1,1
6
```

Modelo Hardware (hardware.py)

- **Propósito:**
Gestionar el inventario de productos
- **Campos básicos:**
Nombre, descripción, modelo, precio, marca, imagen, stock, responsable
- **Campo computado:**
Ticket_count: Calcula incidencias asociadas
- **Validaciones y Acciones:**
Valida precio y ofrece action_view_tickets para acceder a los tickets relacionados

```

models > hardware.py > ...
1  from odoo import models, fields, api
2
3  class Hardware(models.Model):
4      _name = 'gestion.hardware'
5      _description = 'Gestión de Hardware'
6
7      name = fields.Char(string="Nombre del Hardware", required=True)
8      description = fields.Text(string="Descripción")
9      modelo = fields.Char(string="Modelo")
10     precio = fields.Integer(string="Precio", required=True)
11     marca = fields.Char(string="Marca")
12     imagen = fields.Binary(string="Imagen")
13     stock = fields.Integer(string="Stock", default=0, help="Cantidad disponible")
14     user_id = fields.Many2one('res.users', string="Responsable", help="Usuario responsable del hardware")
15
16     # Campo computado: cuenta los tickets asociados a través de ticket_line
17     ticket_count = fields.Integer(string="Cantidad de Tickets", compute="_compute_ticket_count", store=False)
18
19     def _compute_ticket_count(self):
20         for hw in self:
21             ticket_lines = self.env['gestion.ticket.line'].search([('hardware_id', '=', hw.id)])
22             hw.ticket_count = len(ticket_lines.mapped('ticket_id'))
23
24     @api.constrains('precio')
25     def _check_precio_positive(self):
26         for record in self:
27             if record.precio < 0:
28                 raise ValueError("El precio debe ser un valor positivo")
29
30     def action_view_tickets(self):
31         return {
32             'type': 'ir.actions.act_window',
33             'name': 'Tickets asociados a este Hardware',
34             'view_mode': 'list,form,search',
35             'res_model': 'gestion.ticket',
36             'domain': [('ticket_line_ids.hardware_id', '=', self.id)],
37         }
38

```

Modelo Ticket (ticket.py)

- **Propósito:**
Registrar tickets que se usan para entregar comprobantes y controlar tickets pagados y pendientes
- **Campos clave:**
Titulo, descripción, fechas, estado; relaciones con clientes, contactos, líneas de producto y mantenimientos
- **Campos computados:**
Total_price y pagado para mostrar el estado visualmente
- **Acciones:**
Toggle_estado para cambiar el estado y action_print_ticket para generar el PDF


```

ticket.py 1 X
models > ticket.py > ...
1  from odoo import models, fields, api
2  from datetime import datetime
3
4  class Ticket(models.Model):
5      _name = 'gestion.ticket'
6      _description = 'Ticket de Soporte'
7
8      # Campos básicos
9      name = fields.Char(string="Título", required=True)
10     description = fields.Text(string="Descripción")
11     fecha_creacion = fields.Datetime(string="Fecha de Creación", default=fields.Datetime.now, required=True)
12     fecha_ticket = fields.Date(string="Fecha del Ticket", required=True)
13
14     estado = fields.Selection(
15         [
16             ('pendiente', 'Pendiente'),
17             ('pagado', 'Pagado')
18         ],
19         string="Estado",
20         default='pendiente'
21     )
22
23     # Relaciones con res.partner
24     cliente_id = fields.Many2one('res.partner', string="Cliente", required=True)
25     partner_ids = fields.Many2many('res.partner', string="Contactos Asociados",
26                                   help="Contactos adicionales relacionados con este ticket")
27
28     # Relación One2many: líneas de producto, para mas de un producto
29     ticket_line_ids = fields.One2many('gestion.ticket.line', 'ticket_id', string="Productos")
30
31     # Campo calculado: suma el total de las líneas + el costo de mantenimientos para obtener el precio total
32     total_price = fields.Float(string="Precio Total", compute="_compute_total_price", store=True)
33
34     # Mantenimientos realizados asociados al ticket
35     maintenance_ids = fields.Many2many(
36         'x_gestion.mantenimiento',
37         string="Mantenimientos realizados",
38         help="Mantenimientos realizados al hardware asociados a este ticket"
39     )
40
41     @api.depends('ticket_line_ids.line_total', 'maintenance_ids.cost')
42     def _compute_total_price(self):
43         for ticket in self:
44             total_lines = sum(line.line_total for line in ticket.ticket_line_ids)
45             total_maintenance = sum(m.cost for m in ticket.maintenance_ids)
46             ticket.total_price = total_lines + total_maintenance
47
48     pagado = fields.Html(string="Pagado", compute="_compute_estado_icon", store=False)
49
50     @api.depends('estado')
51     def _compute_estado_icon(self):
52         for record in self:
53             if record.estado == 'pagado':
54                 record.pagado = '<span style="color:green; font-size:16px;">&#x2713;</span>'
55             else:
56                 record.pagado = '<span style="color:red; font-size:16px;">&#x2717;</span>'
57
58     def toggle_estado(self):
59         for record in self:
60             record.estado = 'pagado' if record.estado == 'pendiente' else 'pendiente'
61
62     def action_print_ticket(self):
63         self.ensure_one()
64         return self.env.ref('gestion_hardware.action_report_ticket').report_action(self)
65

```

Modelo ticket line (ticket_line.py)

- **Propósito:**
Gestionar cada línea de producto en un ticket
- **Calculo automático:**
 $\text{Line_total} = \text{precio del producto} \times \text{cantidad}$
- **Control de Stock:**
Métodos create, write y unlink ajustan el inventario automáticamente
- **Validaciones:**
Se asegura que la cantidad sea mayor que 0

```
ticket_line.py 2 X
models > ticket_line.py > ...
1 from odoo import models, fields, api
2 from odoo.exceptions import ValidationError
3
4 class TicketLine(models.Model):
5     _name = 'gestion.ticket.line'
6     _description = 'Línea de Producto en Ticket'
7
8     # Relación Many2one con el ticket
9     ticket_id = fields.Many2one('gestion.ticket', string="Ticket", ondelete='cascade')
10    # Relación Many2one con producto
11    hardware_id = fields.Many2one('gestion.hardware', string="Producto", required=True)
12    # Cantidad de producto, debe ser mayor que 0
13    quantity = fields.Integer(string="Cantidad", default=1, required=True)
14    # Campo calculado: total de la línea = precio del hardware * cantidad
15    line_total = fields.Float(string="Total Línea", compute="_compute_line_total", store=True)
16
17    @api.depends('hardware_id', 'quantity')
18    def _compute_line_total(self):
19        for line in self:
20            if line.hardware_id:
21                line.line_total = line.hardware_id.precio * line.quantity
22            else:
23                line.line_total = 0
24
25    @api.constrains('quantity')
26    def _check_quantity_positive(self):
27        for line in self:
28            if line.quantity <= 0:
29                raise ValidationError("La cantidad debe ser mayor a 0")
30
31    @api.model
32    def create(self, vals):
33        hardware_id = vals.get('hardware_id')
34        quantity = vals.get('quantity', 0)
35        hardware = self.env['gestion.hardware'].browse(hardware_id)
36        if hardware.stock < quantity:
37            raise ValidationError("No hay suficiente stock para este producto (stock actual: %s)" % hardware.stock)
38        record = super(TicketLine, self).create(vals)
39        hardware.stock -= record.quantity
40        return record
```

```

41
42     def write(self, vals):
43         for line in self:
44             new_qty = vals.get('quantity', line.quantity)
45             if new_qty != line.quantity:
46                 diff = new_qty - line.quantity
47                 # Si se incrementa la cantidad, se verifica el stock adicional disponible
48                 if diff > 0:
49                     if line.hardware_id.stock < diff:
50                         raise ValidationError("No hay suficiente stock para aumentar la cantidad (stock actual: %s)" % line.hardware_id.stock)
51                     line.hardware_id.stock += diff
52                 # Si se reduce la cantidad, se devuelve la diferencia al stock
53                 else:
54                     line.hardware_id.stock += -diff
55         return super(Ticketline, self).write(vals)
56
57     def unlink(self):
58         for line in self:
59             if line.hardware_id:
60                 line.hardware_id.stock += line.quantity
61         return super(Ticketline, self).unlink()
62

```

Modelo Mantenimiento (mantenimiento.py)

- **Propósito:**
Registrar servicios realizados sobre los productos
- **Campos:**
Nombre, descripción, costo
- **Relación:**
Hardware_ids: Asocia mantenimientos a productos

```

models > mantenimiento.py > ...
1  from odoo import models, fields
2
3  class Mantenimiento(models.Model):
4      # Se renombra el modelo para que se considere custom (nombre que inicia con "x_")
5      _name = 'x_gestion.mantenimiento'
6      _description = 'Mantenimiento de Hardware'
7
8      # Campos básicos
9      name = fields.Char(string="Nombre del Mantenimiento", required=True)
10     description = fields.Text(string="Descripción")
11     cost = fields.Float(string="Costo", required=True)
12
13     # Relación Many2many con Hardware
14     hardware_ids = fields.Many2many('gestion.hardware', string="Hardware Asociado",
15                                     help="Hardware al que se aplica el mantenimiento")
16

```

Extensión del módulo Cliente (cliente.py)

- **Propósito:**
Registrar a los clientes que generan un ticket y llevar su historial
- **Elementos clave:**
Campo tickets_ids y método action_view_tickets en res.partner

```
cliente.py 1 x
models > cliente.py > ...
1  from odoo import models, fields
2
3  class Cliente(models.Model):
4      _inherit = 'res.partner'
5
6      # One2many para asociar tickets a cada cliente
7      ticket_ids = fields.One2many('gestion.ticket', 'cliente_id', string="Tickets")
8
9  def action_view_tickets(self):
10     """
11     Abre una ventana con la lista de tickets asociados a este cliente.
12     """
13     self.ensure_one()
14     return {
15         'type': 'ir.actions.act_window',
16         'name': 'Tickets del Cliente',
17         'view_mode': 'list,form',
18         'res_model': 'gestion.ticket',
19         'domain': [('cliente_id', '=', self.id)],
20     }
21
```

Vistas y Menús (Archivos XML)

- **Vistas:**
Formularios, listas y búsquedas para cada modelo
- **Menú:**
Menú principal “Gestión de Inventario, Tickets, Mantenimientos y Clientes” y submenús
- **Reportes:**
Report_layout.xml y report_ticket.xml definen el formato de los comprobantes en PDF

- *Cliente_views.xml*

```
gestion_hardware > views > cliente_views.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <odoo>
3      <!-- Vista de formulario para Cliente -->
4      <record id="view_gestion_cliente_form" model="ir.ui.view">
5          <field name="name">gestion.cliente.form.custom</field>
6          <field name="model">res.partner</field>
7          <field name="inherit_id" ref="base.view_partner_form"/>
8          <field name="arch" type="xml">
9              <xpath expr="//sheet/notebook" position="inside">
10                  <page string="Tickets">
11                      <field name="ticket_ids" options="{
12                          'list_view_ref': 'gestion_hardware.view_ticket_list',
13                          'form_view_ref': 'gestion_hardware.view_ticket_form'
14                      }"/>
15                  </page>
16              </xpath>
17          </field>
18      </record>
19
20      <!-- Vista de lista para Cliente -->
21      <record id="view_gestion_cliente_list" model="ir.ui.view">
22          <field name="name">gestion.cliente.list.custom</field>
23          <field name="model">res.partner</field>
24          <field name="arch" type="xml">
25              <list string="Clientes">
26                  <field name="name"/>
27                  <field name="phone"/>
28                  <field name="email"/>
29              </list>
30          </field>
31      </record>
32
33      <!-- Filtra solo aquellos clientes que tienen tickets asociados -->
34      <record id="action_gestion_cliente" model="ir.actions.act_window">
35          <field name="name">Clientes con Ticket</field>
36          <field name="res_model">res.partner</field>
37          <field name="view_mode">list,form</field>
38          <field name="domain">[('ticket_ids','!=',False)]</field>
39          <field name="view_id" ref="gestion_hardware.view_gestion_cliente_list"/>
40      </record>
41  </odoo>
```

- *Hardware_views.xml*

```

<?xml version="1.0" encoding="UTF-8"?>
<odoo>
  <!-- Vista de formulario para Hardware -->
  <record id="view_hardware_form" model="ir.ui.view">
    <field name="name">gestion.hardware.form</field>
    <field name="model">gestion.hardware</field>
    <field name="arch" type="xml">
      <form string="Hardware">
        <sheet>
          <group>
            <field name="name"/>
            <field name="description"/>
            <field name="modelo"/>
            <field name="precio"/>
            <field name="stock"/>
            <field name="marca"/>
          </group>
          <group>
            <field name="imagen" widget="image" style="width: 265px; height:265px;"/>
            <field name="user_id"/>
          </group>
          <group>
            <field name="ticket_count" readonly="1"/>
            <button name="action_view_tickets" type="object" string="Ver Tickets"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>

  <!-- Vista de lista para Hardware -->
  <record id="view_hardware_list" model="ir.ui.view">
    <field name="name">gestion.hardware.list</field>
    <field name="model">gestion.hardware</field>
    <field name="arch" type="xml">
      <list string="Hardware">
        <field name="name"/>
        <field name="precio"/>
        <field name="stock"/>
        <field name="ticket_count"/>
      </list>
    </field>
  </record>

  <!-- Vista de busqueda para Hardware -->
  <record id="view_hardware_search" model="ir.ui.view">
    <field name="name">gestion.hardware.search</field>
    <field name="model">gestion.hardware</field>
    <field name="arch" type="xml">
      <search string="Buscar Hardware">
        <field name="name"/>
        <!-- Filtros personalizados -->
        <filter name="filter_precio_alto" string="Precio Alto" domain="[('precio','>=',1000)]"/>
        <filter name="filter_sin_stock" string="Sin Stock" domain="[('stock','=',0)]"/>
        <filter name="filter_con_stock" string="Con Stock" domain="[('stock','>',0)]"/>
        <filter string="Marca" name="filter_marca" context="{ 'group_by': 'marca' }"/>
      </search>
    </field>
  </record>

  <!-- Accion para Hardware -->
  <record id="action_hardware" model="ir.actions.act_window">
    <field name="name">Hardware</field>
    <field name="res_model">gestion.hardware</field>
    <field name="view_mode">list,form,search</field>
    <field name="search_view_id" ref="view_hardware_search"/>
  </record>
</odoo>

```

- *Mantenimiento_views.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<odoo>
  <!-- Vista de formulario para Mantenimiento -->
  <record id="view_mantenimiento_form" model="ir.ui.view">
    <field name="name">gestion.mantenimiento.form</field>
    <field name="model">x_gestion.mantenimiento</field>
    <field name="arch" type="xml">
      <form string="Mantenimiento">
        <sheet>
          <group>
            <field name="name"/>
            <field name="cost"/>
          </group>
          <group>
            <field name="description"/>
          </group>
          <group>
            <field name="hardware_ids" widget="many2many_tags"/>
          </group>
        </sheet>
      </form>
    </field>
  </record>

  <!-- Vista de lista para Mantenimiento -->
  <record id="view_mantenimiento_list" model="ir.ui.view">
    <field name="name">gestion.mantenimiento.list</field>
    <field name="model">x_gestion.mantenimiento</field>
    <field name="arch" type="xml">
      <list string="Mantenimientos">
        <field name="name"/>
        <field name="cost"/>
      </list>
    </field>
  </record>
```



```

<!-- Vista de búsqueda para Mantenimiento -->
<record id="view_mantenimiento_search" model="ir.ui.view">
    <field name="name">gestion.mantenimiento.search</field>
    <field name="model">x_gestion.mantenimiento</field>
    <field name="arch" type="xml">
        <search string="Buscar Mantenimientos">
            <field name="name"/>
            <field name="cost"/>
        </search>
    </field>
</record>

<!-- Accion para Mantenimiento -->
<record id="action_mantenimiento" model="ir.actions.act_window">
    <field name="name">Mantenimientos</field>
    <field name="res_model">x_gestion.mantenimiento</field>
    <field name="view_mode">list,form,search</field>
</record>
</odoo>

```

- *Menú_views.xml*

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <odoo>
3     <!-- Menu principal: Gestion de Hardware -->
4     <record id="menu_gestion_hardware_app" model="ir.ui.menu">
5         <field name="name">Gestion de Hardware</field>
6         <field name="sequence">1</field>
7         <field name="web_icon" eval="'base/static/img/icons/gest_hardware.png'"/>
8     </record>
9
10    <!-- Submenu: Hardware -->
11    <record id="submenu_hardware_list" model="ir.ui.menu">
12        <field name="name">Hardware</field>
13        <field name="parent_id" ref="menu_gestion_hardware_app"/>
14        <field name="action" ref="action_hardware"/>
15        <field name="sequence">1</field>
16    </record>
17
18    <!-- Submenu: Tickets -->
19    <record id="submenu_ticket_list" model="ir.ui.menu">
20        <field name="name">Tickets</field>
21        <field name="parent_id" ref="menu_gestion_hardware_app"/>
22        <field name="action" ref="action_ticket"/>
23        <field name="sequence">2</field>
24    </record>
25
26    <!-- Submenu: Mantenimientos -->
27    <record id="submenu_mantenimiento_list" model="ir.ui.menu">
28        <field name="name">Mantenimientos</field>
29        <field name="parent_id" ref="menu_gestion_hardware_app"/>
30        <field name="action" ref="action_mantenimiento"/>
31        <field name="sequence">3</field>
32    </record>
33
34    <!-- Submenu: clientes -->
35    <record id="submenu_clientes_ticket" model="ir.ui.menu">
36        <field name="name">Clientes</field>
37        <field name="parent_id" ref="menu_gestion_hardware_app"/>
38        <field name="action" ref="action_gestion_cliente"/>
39        <field name="sequence">4</field>
40    </record>
41 </odoo>
42

```


- *Report_layout.xml*

```

<?xml version="1.0" encoding="UTF-8"?>
<odoo>
  <template id="custom_external_layout">
    <t t-call="web.html_container">
      <div class="page" style="font-family: Arial, sans-serif; font-size: 12px; color: #333;">
        <!-- Cabecera -->
        <div class="header" style="text-align: center; padding: 10px 0; border-bottom: 2px solid #ddd;">
          
          <p style="margin: 0;">Dirección: Calle Ejemplo 123, Granada, España</p>
          <p style="margin: 0;">Teléfono: +34 662 443 794 | Email: alvaro@gestion.com</p>
        </div>
        <!-- Contenido -->
        <div class="content" style="padding: 20px;">
          <t t-raw="0"/>
        </div>
        <!-- Pie de pagina -->
        <div class="footer" style="text-align: center; padding: 10px 0; border-top: 2px solid #ddd; font-size: 10px;">
          <p>No se realizan cambios ni devoluciones pasados 15 días de la compra del producto.</p>
        </div>
      </div>
    </t>
  </template>
</odoo>

```

- *Ticket_views.xml*

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <odoo>
3    <!-- Vista de formulario para Ticket -->
4    <record id="view_ticket_form" model="ir.ui.view">
5      <field name="name">gestion.ticket.form</field>
6      <field name="model">gestion.ticket</field>
7      <field name="arch" type="xml">
8        <form string="Ticket de Soporte">
9          <header>
10             <field name="estado" widget="statusbar"
11               statusbar_visible="pendiente,pagado"
12               decoration-danger="estado=='pendiente'"
13               decoration-success="estado=='pagado'"/>
14             <button name="toggle_estado" type="object" string="Cambiar Estado" class="oe_highlight"/>
15             <button name="action_print_ticket" type="object" string="Imprimir Ticket" class="oe_highlight"/>
16           </header>
17           <sheet>
18             <group>
19               <field name="name"/>
20               <field name="fecha_creacion"/>
21               <field name="fecha_ticket"/>
22             </group>
23             <group>
24               <field name="cliente_id"/>
25             </group>
26             <group>
27               <field name="partner_ids" widget="many2many_tags"/>
28             </group>
29             <group>
30               <field name="ticket_line_ids" widget="one2many_list"
31                 options='{ "list_view_ref": "gestion_hardware.ticket_line_list_view" }'/>
32             </group>
33             <group>
34               <field name="total_price" readonly="1"/>
35             </group>

```

```

35         </group>
36         <group>
37             <field name="description"/>
38         </group>
39         <group>
40             <field name="maintenance_ids" widget="many2many_tags"/>
41         </group>
42     </sheet>
43 </form>
44 </field>
45 </record>
46
47 <!-- Vista de lista para Ticket -->
48 <record id="view_ticket_list" model="ir.ui.view">
49     <field name="name">gestion.ticket.list</field>
50     <field name="model">gestion.ticket</field>
51     <field name="arch" type="xml">
52         <list string="Tickets de Soporte">
53             <field name="name"/>
54             <field name="fecha_ticket"/>
55             <field name="pagado" widget="html"/>
56             <field name="total_price"/>
57         </list>
58     </field>
59 </record>
60
61 <!-- Vista de búsqueda para Ticket -->
62 <record id="view_ticket_search" model="ir.ui.view">
63     <field name="name">gestion.ticket.search</field>
64     <field name="model">gestion.ticket</field>
65     <field name="arch" type="xml">
66         <search string="Buscar Tickets">
67             <field name="name"/>
68             <filter name="filter_pagado" string="Pagado" domain="[('estado','=','pagado')]" />
69             <filter name="filter_pendiente" string="Pendiente" domain="[('estado','=','pendiente')]" />
70         </search>
71     </field>
72 </record>
73
74 <!-- Acción para abrir la vista de Tickets -->
75 <record id="action_ticket" model="ir.actions.act_window">
76     <field name="name">Tickets</field>
77     <field name="res_model">gestion.ticket</field>
78     <field name="view_mode">list,form,search</field>
79 </record>
80
81 <!-- Vista externa para las líneas de ticket (Ticket line) -->
82 <record id="ticket_line_list_view" model="ir.ui.view">
83     <field name="name">gestion.ticket.line.list</field>
84     <field name="model">gestion.ticket.line</field>
85     <field name="type">list</field>
86     <field name="arch" type="xml">
87         <list>
88             <field name="hardware_id"/>
89             <field name="quantity"/>
90             <field name="line_total" readonly="1"/>
91         </list>
92     </field>
93 </record>
94 </odoo>

```

- *Report_ticket.xml*

```

reports > report_ticket.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <odoo>
3      <!-- Template principal del reporte -->
4      <template id="report_ticket">
5          <t t-call="web.html_container">
6              <t t-foreach="docs" t-as="doc">
7                  <t t-call="gestion_hardware.report_ticket_document" t-lang="doc.cliente_id.lang if doc.cliente_id else 'es_ES'"/>
8              </t>
9          </t>
10     </template>
11
12     <!-- Documento del Ticket de Soporte -->
13     <template id="report_ticket_document">
14         <t t-set="doc" t-value="doc.with_context(lang=(doc.cliente_id.lang if doc.cliente_id else 'es_ES'))"/>
15         <!-- layout personalizado -->
16         <t t-call="gestion_hardware.custom_external_layout">
17             <!-- Encabezado del ticket -->
18             <div style="margin-bottom: 20px;">
19                 <h1 style="text-align: center; color: #000080; margin-bottom: 5px;">Ticket</h1>
20                 <hr style="border: 1px solid #ccc;">
21                 <table style="width: 100%; margin-top: 20px;">
22                     <tr>
23                         <td style="width: 30%; font-weight: bold; padding: 5px;">Titulo</td>
24                         <td style="padding: 5px; text-align: right;">{{ doc.name }}</td>
25                     </tr>
26                     <tr>
27                         <td style="font-weight: bold; padding: 5px;">Descripcion</td>
28                         <td style="padding: 5px; text-align: right;">{{ doc.description }}</td>
29                     </tr>
30                     <tr>
31                         <td style="font-weight: bold; padding: 5px;">Fecha del Ticket</td>
32                         <td style="padding: 5px; text-align: right;">{{ doc.fecha_ticket }}</td>
33                     </tr>
34                     <tr>
35                         <td style="font-weight: bold; padding: 5px;">Estado</td>
36                         <td style="padding: 5px; text-align: right;">{{ doc.estado }}</td>
37                     </tr>
38                     <tr>
39                         <td style="font-weight: bold; padding: 5px;">Cliente</td>
40                         <td style="padding: 5px; text-align: right;">{{ doc.cliente_id.name if doc.cliente_id else 'Cliente no asignado' }}</td>
41                     </tr>
42                 </table>
43             </div>
44
45             <!-- Seccion de Productos -->
46             <div style="margin-top: 30px;">
47                 <h2 style="color: #000080; margin-bottom: 10px;">Productos</h2>
48                 <table style="width: 100%; border-collapse: collapse;">
49                     <thead>
50                         <tr style="background-color: #f2f2f2;">
51                             <th style="border: 1px solid #ddd; padding: 8px; text-align: left;">Producto</th>
52                             <th style="border: 1px solid #ddd; padding: 8px; text-align: right;">Cantidad</th>
53                             <th style="border: 1px solid #ddd; padding: 8px; text-align: right;">Total</th>
54                         </tr>
55                     </thead>
56                     <tbody>
57                         <t t-foreach="doc.ticket_line_ids" t-as="line">
58                             <tr>
59                                 <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">{{ line.hardware_id.name }}</td>
60                                 <td style="border: 1px solid #ddd; padding: 8px; text-align: right;">{{ line.quantity }}</td>
61                                 <td style="border: 1px solid #ddd; padding: 8px; text-align: right;">{{ line.line_total }}</td>
62                             </tr>
63                         </t>
64                     </tbody>
65                 </table>
66             </div>
67
68             <!-- Seccion de Mantenimientos solo si existen -->
69             <t t-if="doc.maintenance_ids">
70                 <div style="margin-top: 30px;">
71                     <h2 style="color: #000080; margin-bottom: 10px;">Mantenimientos Realizados</h2>
72                     <table style="width: 100%; border-collapse: collapse;">
73                         <thead>
74                             <tr style="background-color: #f2f2f2;">
75                                 <th style="border: 1px solid #ddd; padding: 8px; text-align: left;">Nombre</th>
76                                 <th style="border: 1px solid #ddd; padding: 8px; text-align: right;">Costo</th>
77                                 <th style="border: 1px solid #ddd; padding: 8px; text-align: left;">Descripcion</th>
78                             </tr>
79                         </thead>

```

```

80         <tbody>
81             <t t-foreach="doc.maintenance_ids" t-as="mnt">
82                 <tr>
83                     <td style="border: 1px solid #ddd; padding: 8px;" t-esc="mnt.name"/>
84                     <td style="border: 1px solid #ddd; padding: 8px; text-align: right;" t-esc="mnt.cost"/>
85                     <td style="border: 1px solid #ddd; padding: 8px;" t-esc="mnt.description"/>
86                 </tr>
87             </t>
88         </tbody>
89     </table>
90 </div>
91 </t>
92
93     <!-- Total del Ticket -->
94     <div style="margin-top: 20px; text-align: right; font-size: 16px; font-weight: bold;">
95         <h3>Total de venta: <span t-esc="doc.total_price"/> Euros;</h3>
96     </div>
97 </t>
98 </template>
99
100 <!-- Definición del reporte PDF -->
101 <record id="action_report_ticket" model="ir.actions.report">
102     <field name="name">Ticket de Soporte</field>
103     <field name="model">gestion.ticket</field>
104     <field name="report_type">qweb-pdf</field>
105     <field name="report_name">gestion_hardware.report_ticket</field>
106     <field name="binding_model_id" ref="model_gestion_ticket"/>
107     <field name="binding_type">report</field>
108     <field name="attachment_use">False</field>
109 </record>
110 </odoo>

```

Manual de Usuario

El manual de usuario explica de forma practica como instalar, configurar y utilizar la aplicación. Se incluyen instrucciones paso a paso

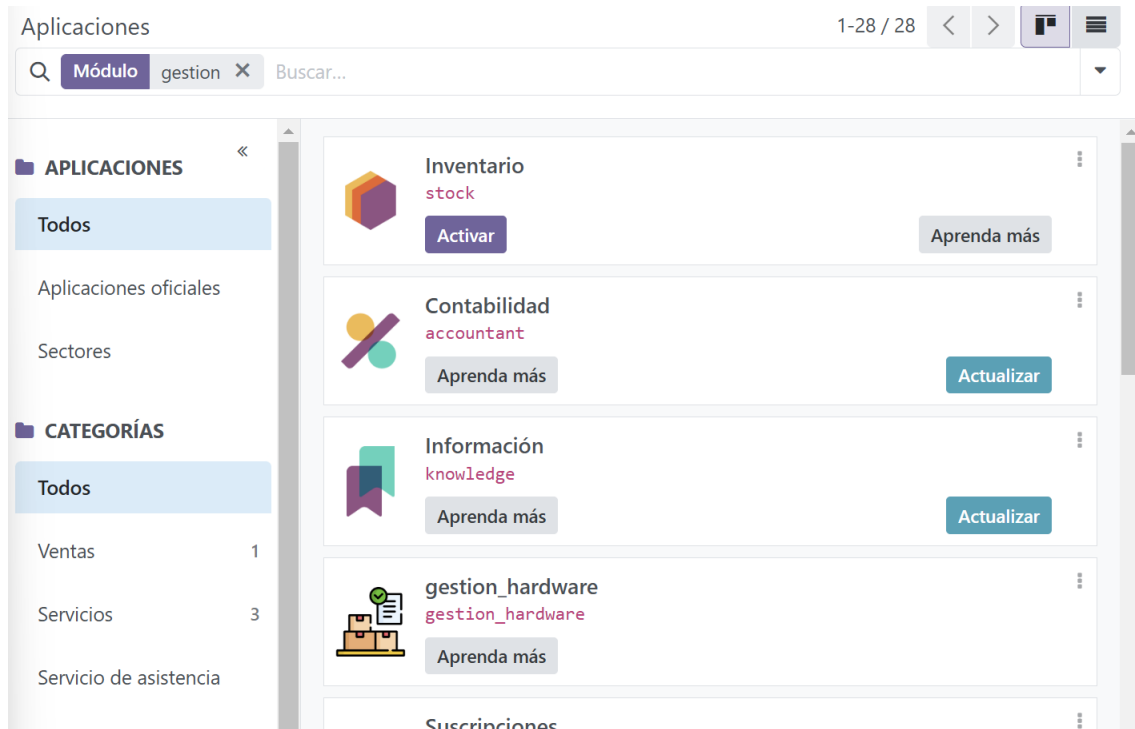
Instalación y configuración inicial

Requisitos previos

- Odoo en versión compatible (Odoo 18 o superior)
- Módulos base y producto activos
- Acceso al directorio addons

Pasos para la instalación

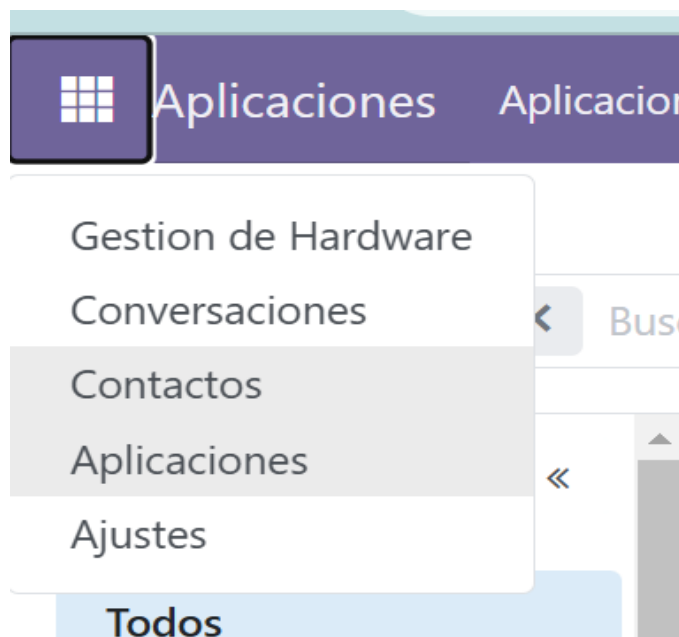
- 1. Copia del módulo:**
Descarga y coloca la carpeta del modulo en el directorio de addons
- 2. Actualizar la lista de módulos:**
Selecciona “Actualizar lista de módulos” desde el menú de aplicaciones
- 3. Instalación:**
Busca “gestion_hardware” y pulsa en instalar
- 4. Verificación inicial:**
Comprueba que el icono y el nombre aparezcan correctamente



Navegación en la aplicación

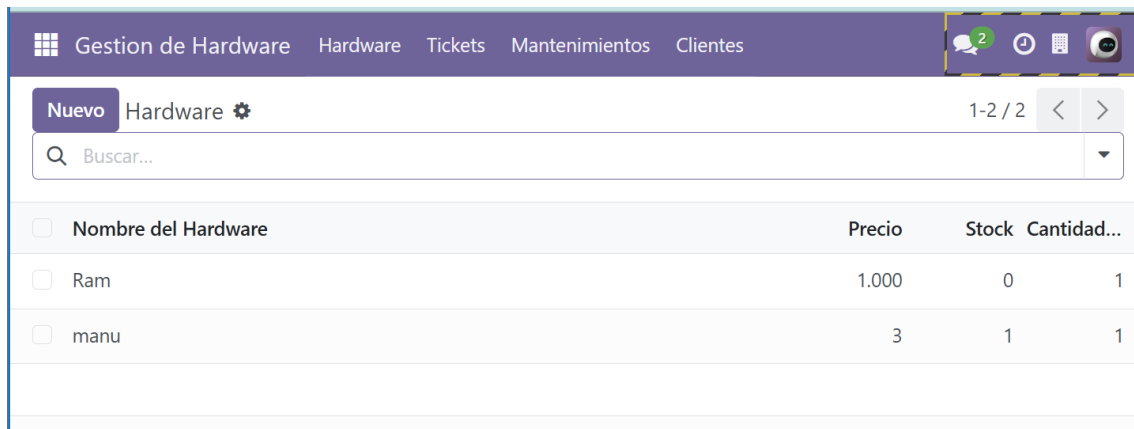
Inicio de sesión y acceso

- Inicia sesión con tus credenciales en Odoo
- La aplicación se mostrará en el panel principal



Menú principal y submenús

- Menú principal: “Gestión de Inventario, tickets, mantenimientos y clientes
- Submenús:
 - Inventario (hardware): gestión y control del stock
 - Tickets: registro de incidencias (comprobantes para clientes y control de pagos)
 - Mantenimientos: registro y asignación de servicios
 - Clientes: registro de clientes (generados al emitir un ticket) y su historial.
 -



The screenshot shows the 'Gestion de Hardware' section of the application. At the top, there is a navigation bar with tabs for 'Gestion de Hardware', 'Hardware', 'Tickets', 'Mantenimientos', and 'Clientes'. Below this, there is a 'Nuevo Hardware' button and a search bar labeled 'Buscar...'. The main area displays a table with the following data:

<input type="checkbox"/>	Nombre del Hardware	Precio	Stock	Cantidad...
<input type="checkbox"/>	Ram	1.000	0	1
<input type="checkbox"/>	manu	3	1	1




Gestión del inventario (hardware)

Creación de un registro de hardware

1. Acceso a la sección:
En “Inventario”, pulsa “Crear”.
2. Rellenar el formulario:
 - Nombre: Ejemplo: “Impresora Láser HP”
 - Descripción: “Impresora láser de alta velocidad, ideal para oficinas”
 - Modelo: “LaserJet Pro MFP M426”
 - Precio: Ej. 250 (valor positivo)
 - Marca: “HP”
 - Imagen: Sube una imagen
 - Stock: Ej. 10 unidades
 - Responsable: Selecciona el usuario encargado en caso de que se necesite
3. Guardar registro:
Se actualiza el stock y el campo de incidencias (inicialmente 0)

Nuevo

Hardware
Nuevo



Nombre del Hardware

Descripción

Modelo

Precio


0

Stock ?

0

Marca

Imagen



Responsable ?

Cantidad de Tickets 0

Ver Tickets

Edición y consulta



- Visualiza la lista de productos y edita o consulta detalles
- Pulsa “Ver Tickets” para ver incidencias asociadas

Nuevo

Hardware / Ram

Tickets asociados a este Hardware

1-1 / 1



Q

Buscar...

<input type="checkbox"/> Título	Fecha del ...	Pagado	Precio Total
<input type="checkbox"/> venta1	21/02/2025	✓	100.300,00

Gestión de tickets

Registro de incidencias y comprobantes

1. Crear un ticket:
En “Tickets”, pulsa “Crear”
2. Completar el formulario:
 - Título: Ejemplo: “Falla en impresora HP”
 - Descripción: “La impresora no imprime y muestra errores”
 - Fecha del ticket: selecciona la fecha
 - Estado: Inicialmente “pendiente”
 - Cliente: Selecciona el cliente (se registra automáticamente al generar el ticket)
 - Contactos: Añade contactos adicionales si procede
3. Agregar líneas de producto:
 - En “Productos”, pulsa “Agregar un elemento”
 - Selecciona el hardware y define la cantidad
 - El sistema calcula el total y verifica el stock

Nuevo Tickets
venta2

2 / 2 < >

Cambiar Estado

Imprimir Ticket

Pendiente

Pagado

Título

venta2

Fecha de Creación

21/02/2025 10:18:43

Fecha del Ticket

21/02/2025

Cliente

Deco Addict

Contactos Asociados ?

Productos

Producto	Cantidad	Total Línea
manu	9	27,00
Añadir una línea		

Precio Total

327,00

Descripción

Mantenimientos realizados ?

Reparacion

Asociaciones en el ticket

- Cambiar estado:
Pulsa “Cambiar Estado” para alternar entre “pendiente” y “pagado”, facilitando el control de tickets entregados y pendientes
- Generar reporte PDF:
Pulsa “Imprimir Ticket” para generar un comprobante en PDF que se entrega al cliente y se registra internamente

Cambiar Estado

Imprimir Ticket

Pendiente

Pagado

Título

venta2

Fecha de Creación

21/02/2025 10:18:43

Fecha del Ticket

21/02/2025

Cliente

Deco Addict

Contactos Asociados ?

Productos

Producto	Cantidad	Total Línea
manu	9	27,00
<div>Añadir una línea</div>		

Precio Total

327,00

Descripción

Mantenimientos realizados ?

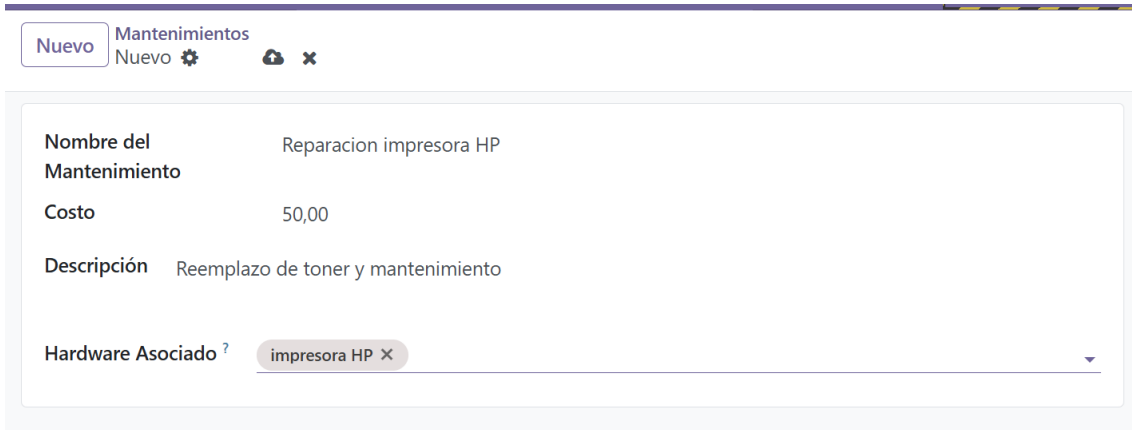
Reparacion X

Gestión de mantenimientos

Registro de mantenimientos

1. Acceder a la sección:
En “Mantenimientos”, pulsa “Crear”
2. Rellenar el formulario:
 - Nombre: Ejemplo: “Reparación de impresora HP”
 - Costo: Ej. 50
 - Descripción: “Reemplazo de tóner y mantenimiento general”
 - Hardware asociado: Selecciona uno o varios productos (ej. “Impresora Láser HP”)
3. Guardar el registro:

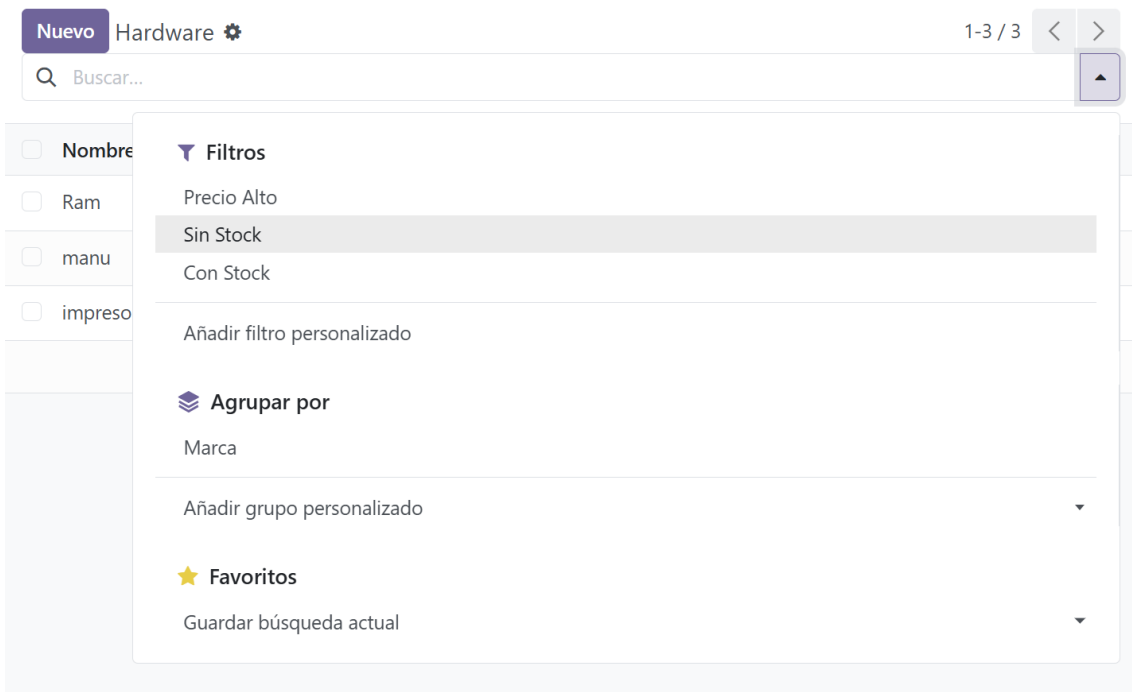
El mantenimiento se vincula a los productos seleccionados



Formulario de Mantenimientos. Encabezado: Nuevo Mantenimientos. Botones: Nuevo, configuración, compartir, cerrar. Campos: Nombre del Mantenimiento (Reparacion impresora HP), Costo (50,00), Descripción (Reemplazo de toner y mantenimiento), Hardware Asociado (impresora HP).

Consulta y búsqueda

- Utiliza la vista lista y la barra de búsqueda para filtrar mantenimientos



Interfaz de consulta y búsqueda. Encabezado: Nuevo Hardware. Barra de búsqueda: Buscar... Filtros: Precio Alto, Sin Stock (seleccionado), Con Stock. Agrupar por: Marca. Favoritos: Guardar búsqueda actual.

Gestión de clientes

Registro y consulta del historial de tickets

1. Registro automático:
Los clientes se registran automáticamente al generar un ticket (cuando se realiza una compra o servicio)

2. Acceso al módulo de tickets
Usa el submódulo “tickets”
3. Visualización de la ficha:
Selecciona un ticket para ver su ficha, que muestra la información y cliente

Cambiar EstadoImprimir Ticket

PendientePagado

Títuloventa1

Fecha de Creación21/02/2025 10:16:35

Fecha del Ticket21/02/2025

ClienteAzure Interior

Contactos Asociados?

Productos

Producto	Cantidad	Total Línea
Ram	100	100.000,00
Añadir una línea		

Precio Total100.300,00

Descripción

Mantenimientos realizados?

Reparacion X

Uso de reportes

Generación del reporte PDF

1. Desde el ticket:
Pulsa “Imprimir ticket” en el formulario del ticket
2. Contenido del reporte:
 - El PDF incluirá:
 - Encabezado: logotipo, dirección, teléfono y correo
 - Detalles: titulo, descripción, fechas, estado y cliente
 - Productos: tabla con cada línea (nombre, cantidad, total)
 - Mantenimientos: tabla (si existen) con detalles de servicios
 - Resumen: total final del ticket
3. Utilidad:
El comprobante PDF sirve para entregar al cliente y para mantener el registro interno, diferenciando tickets pagados y pendientes



Dirección: Calle Ejemplo 123, Granada, España
Teléfono: +34 662 443 794 | Email: alvaro@gestion.com

Ticket

Título: venta2
Descripción:
Fecha del Ticket: 2025-02-21
Estado: pagado
Cliente: Deco Addict

Productos

Producto	Cantidad	Total
manu	9	27.0

Mantenimientos Realizados

Nombre	Costo	Descripción
Reparacion	300.0	Reparacion

Total de venta: 327.0 Euros;

Webgrafía

- Documentación oficial de Odoo:
 - <https://www.odoo.com/documentation>
- Ejemplos y tutoriales de OpenAcademy:
 - <https://github.com/OCA/OpenAcademy>
- Foros de ayuda de Odoo:
 - <https://www.odoo.com/forum/help-1>
- Guía de buenas prácticas en desarrollo de módulos de Odoo:
 - <https://vauxoo.github.io/odoo/howtos/backend.html>
- Informes personalizados:
 - <https://www.odoo.com/documentation/18.0/es/applications/finance/accounting/reporting/customize.html>

- https://www.odoo.com/documentation/18.0/es/developer/tutorials/pdf_report_s.html
- Etapas y estados de tareas:
 - https://www.odoo.com/documentation/18.0/es/applications/services/project/tasks/task_stages_statuses.html
- Filtros y agrupaciones personalizadas:
 - <https://www.odoo.com/documentation/18.0/es/applications/essentials/search.html>
- Heredar de un módulo de Odoo:
 - <https://www.youtube.com/watch?v=1R4jtxj72Lc>

Conclusión

El módulo “Gestión de inventario, tickets, mantenimientos y clientes” integra de manera eficaz el control del stock, la generación y seguimiento de tickets (que se utilizan para entregar comprobantes a los clientes y llevar un registro de tickets pagados y pendientes), la asignación de mantenimientos y el registro de clientes que han realizado compras.

Gracias a la implementación de campos computados, validaciones automáticas y vistas personalizadas, el sistema optimiza la gestión operativa y facilita la documentación formal de los servicios.

Link de descarga

- Link modulo:
 - <https://github.com/AlvaroPavon/Modulo-Odoo>