# INTEGRACIÓN DE SISTEMAS EMPRESARIALES AVANZADO

## LABORATORIO N° 09

# Módulo Básico y Herencia


| | | | |
|---|---|---|---|
| ***Alumno(s):*** | | ***Nota*** | |
| ***Grupo:*** | | ***Ciclo: VI*** | |

| *Criterio de Evaluación* | *Excelente (4pts)* | *Bueno (3pts)* | *Requiere mejora (2pts)* | *No acept. (0pts)* | *Puntaje Logrado* |
|---|---|---|---|---|---|
| Identifica los principales componentes de la creación de un módulo | | | | | |
| Prueba la API de Odoo | | | | | |
| Configura un modelo desde el inicio | | | | | |
| Documenta los acuerdos y detalles del trabajo a realizar | | | | | |
| Es puntual y redacta el informe adecuadamente | | | | | |

# Laboratorio 09: Módulo Básico y Herencia

## Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:
- Identificar los principales requerimientos del trabajo a realizar
- Definir las responsabilidades de los integrantes del grupo
- Documentar los alcances del trabajo a realizar

## Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

## Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - Conexión a la red del laboratorio
  - Software de virtualización (Opcional)
  - Software ERP ODOO instalado

## Procedimiento:

1. **Configuración inicial.**
   1.1. Se puede utilizar la base de datos utilizada en el laboratorio 7. Caso contrario, se necesitará una base de datos sin datos de ejemplo.

2. **Módulos a instalar**
   2.1. Inventarios
   2.2. Ventas
   2.3. Compras
   2.4. Punto de Venta

3. **Creación de módulo Facturación**
   3.1. Crearemos nuestra carpeta **facturacion** dentro de nuestra carpeta de addons. Recuerde que esta carpeta debe ser una carpeta externa a la instalación de Odoo, tal como se configuró en laboratorios anteriores para instalar módulos de terceros. Dentro de dicha carpeta crearemos la carpeta **static**. Dentro de la misma, la carpeta **description** y finalmente dentro, pegaremos el archivo **icon.png**.
   3.2. Dentro de dicha carpeta, crearemos el archivo más esencial de nuestro módulo: el manifiesto. Creamos el archivo **__manifest__.py** (doble subguión) con el siguiente contenido:

```
# -*- coding: utf-8 -*-
{
    'name' : 'Facturacion',
    'version' : '1.0',
    'summary': 'Modulo basico para facturacion en Peru',
    'description': """
Core mechanisms for the accounting modules. To display the menuitems, install the module account_invoicing.
    """,
    'depends' : [],
    'data': [
      'views/series_view.xml',
    ]
}
```

   3.3. Procederemos a crear una carpeta llamadas models. Este nombre (a comparación de __manifest__.py) no es obligatorio, pero es una convención siempre llamar así a nuestro folder donde estarán nuestros modelos a crear o modificar. En la raíz de nuestra carpeta facturacion (es decir, a la

altura del __manifest__.py), crearemos el archivo __init__.py (doble subguión) con el siguiente contenido (sirve para importar la carpeta models, o como sea que llamásemos a nuestra carpeta)

```
# -*- coding: utf-8 -*
from . import models
```

3.4. Ahora creamos el archivo __init__.py (doble subguión) dentro de la carpeta models. En este archivo, invocamos los archivos a ser incluidos en nuestra compilación.

```
# -*- coding: utf-8 -*
from . import series
```

3.5. Creamos el archivo series.py con el siguiente contenido.

```
# -*- coding: utf-8 -*-
from odoo import models, fields, api

You, 40 minutes ago | 2 authors (You and others)
class Series(models.Model):
    _name = 'facturacion.series'
    _description = 'Series de documentos'

    name = fields.Char(string='Nombre de la serie')
    prefix = fields.Char(string='Prefijo de la serie')
    document_type = fields.Selection([('01','Factura'),
        ('03','Boleta'),
        ('07','Nota de Crédito'),
        ('08','Nota de Débito')
    ])
    active = fields.Boolean(string='Estado de la serie', default=True)
```

Este archivo es la declaración de nuestro modelo llamado **facturacion.series**. Esto se reflejará en la base de datos como una tabla llamada facturacion_series. Adjunte una imagen de la misma después de haber ejecutado el paso 3.7 de este laboratorio.

- El nombre de la clase (en este caso, Series) puede ser cualquiera, pero se suele poner el nombre del mismo modelo a crear.
- Así mismo, podemos ver que al declarar los campos, les damos un tipo, como Char, Selection, Boolean, etc. En la documentación de Odoo (https://www.odoo.com/documentation/11.0/reference/orm.html) encontrará más detalle de los tipos de campo.

3.6. Esto sin embargo no ha creado nada aún en el sistema visible, ya que debemos declarar un menú donde esté nuestra acción. Crearemos la carpeta **views** (el nombre no es obligatorio, pero es una convención para los archivos de vista) y dentro de este, el archivo **series_view.xml** (una vez más, el nombre es opcional pero es una convención hacer referencia al modelo a modificar) con el siguiente contenido:

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <data>
        <record id="series_tree_view" model="ir.ui.view">
            <field name="name">series.tree.view</field>
            <field name="model">facturacion.series</field>
            <field name="arch" type="xml">
                <tree string="Series">
                    <field name="name"/>
                    <field name="prefix"/>
                    <field name="active"/>
                </tree>
            </field>
        </record>
```

```
        <record model="ir.actions.act_window" id="facturacion.action_series">
            <field name="name">Listado de Series</field>
            <field name="res_model">facturacion.series</field>
            <field name="view_mode">tree,form</field>
        </record>

        <menuitem name="Peru" id="menu_facturacion"
            sequence="40" web_icon="facturacion,static/description/icon.png"/>

        <menuitem name="Series" id="facturacion.menu_1_series"
            parent="facturacion.menu_facturacion" action="facturacion.action_series"/>
    </data>
</odoo>
```

- En el primer record del archivo, estamos declarando la vista tree o listado, que nos mostrará nuestras series creadas.
- El segundo record hace referencia a la acción de mostrar el tree (listado) y luego el formulario de edición.
- El tercer ítem, menuitem, hace referencia al menú que aparecerá en nuestra aplicación después de instalado, que tendrá el nombre Peru (puede tener cualquier nombre).
- El cuarto ítem, menuitem, hace referencia a la opción que aparecerá dentro del módulo Perú. Adjunte una imagen de este menuitem después del paso 3.7

3.7. Procederemos a reiniciar nuestro Odoo (si está en Windows, vaya a la **consola de Servicios**, en el caso de Linux, utilice **sudo service odoo restart**) para luego Actualizar la lista de aplicaciones (recuerde que esto debe ser hecho en modo desarrollador) y luego instalaremos nuestro módulo facturación. Lo reconocerá porque tendrá nuestro ícono declarado en el paso 3.1
Adjunte capturas del funcionamiento del módulo.

**4. Vista de formulario**

4.1. Dentro del archivo series_view.xml agregaremos un record para crear la vista de formulario.

```
    <record id="series_form_view" model="ir.ui.view">
        <field name="name">series.form.view</field>
        <field name="model">facturacion.series</field>
        <field name="arch" type="xml">
            <form string="Editar Serie">
                <sheet>
                    <group>
                        <field name="name"/>
                    </group>
                    <notebook>
                        <page string="Prefijo">
                            <field name="prefix"/>
                        </page>
                        <page string="Estado">
                            <field name="active"/>
                        </page>
                    </notebook>
                </sheet>
            </form>
        </field>
    </record>
```

**Nota:** Si bien ahora estamos usando etiquetas reservadas de Odoo como son form, sheet y notebook, podemos utilizar etiquetas html dentro de esta declaración.

4.2. Actualizaremos nuestro módulo (al solamente haber modificado xml, no se necesita recompilar Python, por lo que no se necesita reiniciar el servicio) y adjuntaremos un GIF del funcionamiento de nuestro listado con su nuevo formulario.

## 5. Vista de búsqueda

5.1. Ahora mismo, nuestro listado permite la búsqueda solamente por el campo name. Modificaremos nuevamente nuestro archivo series_view.xml para agregar el siguiente record de declaración.

```xml
<record id="series_search_view" model="ir.ui.view">
    <field name="name">series.search.view</field>
    <field name="model">facturacion.series</field>
    <field name="arch" type="xml">
        <search>
            <field name="name"/>
            <field name="prefix"/>
        </search>
    </field>
</record>
```

5.2. Actualizaremos nuestro módulo (al solamente haber modificado xml, no se necesita recompilar Python, por lo que no se necesita reiniciar el servicio) y adjuntaremos un GIF del funcionamiento de nuestro listado con su nueva forma de búsqueda.

## 6. Data por defecto

6.1. En muchas ocasiones, necesitaremos importar data por defecto. Para lograr esto, modificaremos nuestro archivo __manifest__.py para agregar la línea **data/series.xml**

```
'data': [
 'data/series.xml',
 'views/series_view.xml',
```

6.2. Crearemos la carpeta **data** y dentro, el archivo **series.xml** con el siguiente contenido. Puede ver que los campos declarados son los mismos de nuestro modelo, y el contenido, su valor a guardar.

```xml
<odoo>
    <data>
        <record id="serie1" model="facturacion.series">
            <field name="name">Serie Boleta</field>
            <field name="prefix">B001</field>
            <field name="document_type">03</field>
            <field name="active">True</field>
        </record>
        <record id="serie2" model="facturacion.series">
            <field name="name">Serie Factura</field>
            <field name="prefix">F001</field>
            <field name="document_type">01</field>
            <field name="active">True</field>
        </record>
        <record id="serie3" model="facturacion.series">
            <field name="name">Serie Nota de Crédito</field>
            <field name="prefix">FC01</field>
            <field name="document_type">07</field>
            <field name="active">True</field>
        </record>
        <record id="serie4" model="facturacion.series">
            <field name="name">Serie Nota de Débito</field>
            <field name="prefix">FD01</field>
            <field name="document_type">08</field>
            <field name="active">True</field>
        </record>
    </data>
</odoo>
```

6.3. Actualizaremos nuestro módulo (al haber modificado archivos python, se necesita recompilar, por lo que se necesita reiniciar el servicio) y adjuntaremos una captura de la importación de nuestra data por defecto generada por el módulo.

## 7. Verificación de API automática

7.1. Ahora, haremos una pausa en el desarrollo y verificaremos que Odoo crea por nosotros una API de consulta del modelo ya creado sin necesidad de hacer alguna configuración extra. Crearemos una carpeta llamada node con el siguiente **package.json.** Luego haremos **npm install** para instalar las dependencias

```
{
  "name": "node",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "odoo-xmlrpc": "^1.0.8"
  }
}
```

7.2. Crearemos el archivo **index.js** con el siguiente contenido. En este ejemplo, yo he puesto las credenciales de mi computadora, pero usted deberá colocar las credenciales de su instancia. Como puede ver, se está haciendo una consulta de lectura de los registros dentro del modelo **facturacion.series**

```
var Odoo = require("odoo-xmlrpc");

var odoo = new Odoo({
  url: "localhost",
  port: 8070,
  db: "bd11_tecsup",
  username: "admin",
  password: "admin"
});

// Connect to Odoo
odoo.connect(function(err) {
  if (err) {
    return console.log(err);
  }
  console.log("Connected to Odoo server.");
  var inParams = [];
  inParams.push([]);
  var params = [];
  params.push(inParams);
  odoo.execute_kw("facturacion.series", "search_read", params, function(
    err,
    value
  ) {
    if (err) {
      return console.log(err);
    }
    console.log("Result: ", value);
  });
});
```

**7.3.** Una vez creado el archivo, ejecutaremos **node index.js** y adjunte un GIF de la ejecución de dicho comando. Deberíamos obtener los registros en dicho modelo, demostrando que Odoo ya crea una API automática para nuestros modelos con solamente declararlos.

## 8. Herencia de modelos

8.1. Hasta ahora hemos creado nuestro propio modelo, pero ahora modificaremos un modelo ya existente. Como hemos creado el modelo series (para facturación electrónica) procederemos a modificar el modelo de facturas de Odoo. Crearemos dentro de la carpeta **models** el archivo **account_invoice.py** con el siguiente contenido.

```
# -*- coding: utf-8 -*-
from odoo import models, fields, api

Unsaved changes (cannot determine recent change or authors)
class AccountInvoice(models.Model):
    _inherit = 'account.invoice'

    serie_id = fields.Many2one(comodel_name='facturacion.series',
        string='Serie Electrónica')
```

Lo más importante a recalcar de este código, es que en vez de declarar un modelo con el atributo **_name**, utilizamos **_inherit** para "heredar" un modelo ya existente. A ese modelo le añadimos el campo serie_id que es de tipo Many2one, es decir, hace referencia a una relación muchos a uno (muchas facturas pueden tener una serie). Esto es muy común para hacer referencia a otros registros.

8.2. Para que este archivo sea considerado, debemos agregarlo a nuestros imports. Modificaremos el archivo **__init__.py** dentro de la carpeta **models** (no el de la raíz del proyecto) para que incluya nuestro nuevo archivo.

```
# -*- coding: utf-8 -*-
from . import series
from . import account_invoice
```

8.3. Crearemos un nuevo archivo dentro de la carpeta **views**, llamado **account_invoice_view.xml**. Nuevamente, no es obligatorio tener que separarlo en otro archivo (podríamos tener todas nuestras vistas en un solo archivo) pero es por un tema de orden, además el nombre tampoco es obligatorio, pero al declarar el modelo que modifica es más fácil identificarlo a futuro.

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <data>
        <record id="invoice_form_series" model="ir.ui.view">
            <field name="name">invoice.series.form.view</field>
            <field name="model">account.invoice</field>
            <field name="inherit_id" ref="account.invoice_form"/>
            <field name="arch" type="xml">
                <field name="date_invoice" position="after">
                    <field name="serie_id"/>
                </field>
            </field>
        </record>
    </data>
</odoo>
```

Cuando heredamos una vista, agregaremos la etiqueta inherit_id y haremos referencia primero al módulo y luego a la vista modificada, en este caso, el módulo es account y la vista invoice_form. Si tienes dudas de la vista a modificar, puede ir en modo desarrollador al Bicho de debug y darle a editar Formulario, allí aparecerá el nombre de la vista que deseas editar.

La otra parte importante, es que ahora ubicamos un campo del modelo, en este caso, **date_invoice**, y le decimos a la vista, después de este campo (el atributo **position** y su valor **after**) coloca este nuevo campo (**serie_id**)

8.4. Finalmente, modificaremos el archivo **__manifest__.py** para agregar dos cosas importantes: en el **depends** debe ir un arreglo de los módulos de los que dependemos (ahora dependemos del módulo account ya que modificamos uno de sus modelos) y agregaremos la vista de **account_invoice_view.xml**

```
    """,
    'depends' : ['account'],
    'data': [
      'data/series.xml',
      'views/series_view.xml',
      'views/account_invoice_view.xml',
    ]
}
```

8.5. Actualizaremos nuestro módulo (al haber modificado archivos python, se necesita recompilar, por lo que se necesita reiniciar el servicio) y adjuntaremos un GIF de la vista de creación de facturas (Dentro del módulo Facturación, Ventas, Facturas de Cliente). Deberíamos ser capaces de relacionar nuestras series a las facturas ahora creadas.

**9. Tarea**

9.1. Deberá crear el modelo **facturacion.documentos** basado en el **catálogo 6** dentro del siguiente PDF:

- http://cpe.sunat.gob.pe/sites/default/files/inline-files/anexoV-340-2017.pdf

9.2. Deberá crear la data por defecto a importar, basada en el PDF antes proveído.
**Nota:** Esta tarea es necesaria para el siguiente laboratorio: