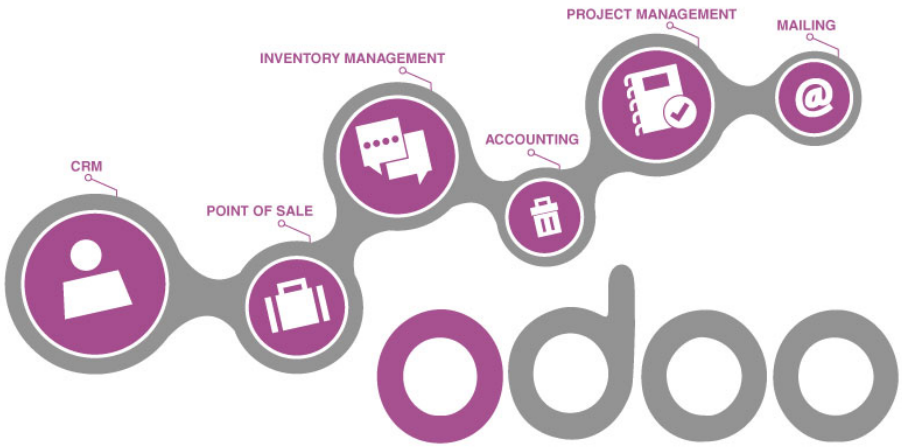
Iniciación a

ODOO

10/11



Recomiendo trabajar con [PyCharm]

Juan Antonio Ramos
JuanRaMar
juanramar@tecnoinfe.com
http://tecnoinfe.com

INDICE:

Sumario

¿Qué es Odoo?

Odoo, anteriormente conocido como **OpenERP** es una completa suite de aplicaciones de gestión empresarial de código abierto, destinadas a cubrir todos los procesos de su empresa, independientemente del sector al que pertenezca: logística, distribución, producción, venta directa a cliente, transporte…

Odoo ofrece a las **PyMEs** el acceso a un software de gestión de sus procesos de negocio completo, fiable, amigable, fácil de usar, dinámico y escalable, que hasta no hace mucho estaba sólo al alcance de grandes organizaciones.

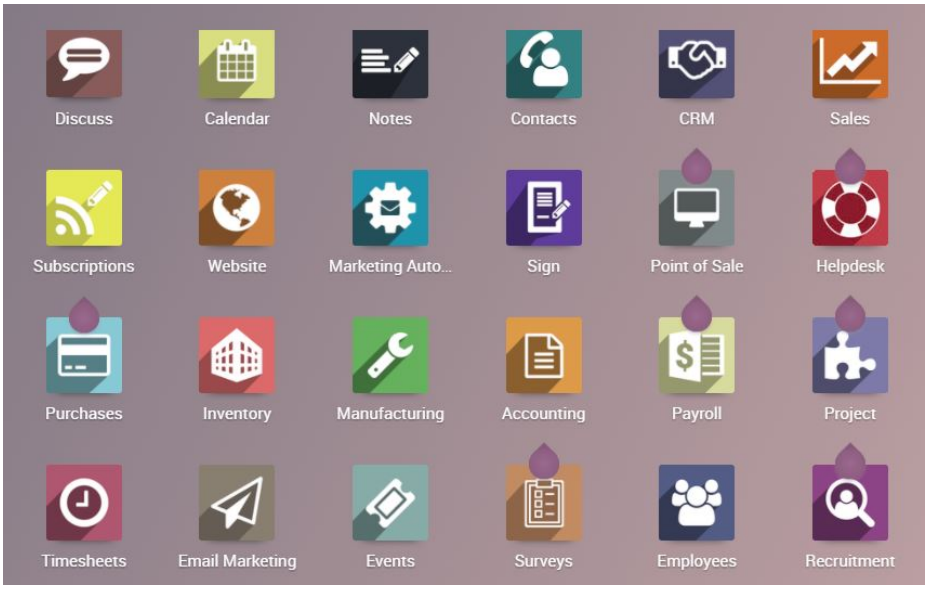
Active solo las aplicaciones que necesite

La herramienta se articula por módulos o aplicaciones, de tal manera que no es necesario instalar todas las aplicaciones, sino que cada empresa implementará sólo aquellos módulos que necesite en cada momento. El software es totalmente escalable, por lo que podemos comenzar a utilizar Odoo con una sola aplicacación e ir añadiendo o eliminando otras en función de nuestras necesidades.

Diferentes formatos de uso

Odoo es un software de código abierto, lo que nos permite, respetando los términos de su licencia, modificar el código para adaptarlo a las necesidades específicas de cada empresa.
Además, esta filosofía nos permite disfrutar de él tanto en formato SaaS (Software como servicio) como instalarlo en los propios servidores de una organización.

Acceso móvil y acceso multiplataforma



La interfaz de Odoo permite acceder desde cualquier equipo independientemente del sistema operativo (Windows, Linux o Mac). Dispone asimismo de versión para dispositivos móviles (smartphones y tablets) que simplifica las vistas para acceder tanto desde iOS como desde Android.

Integración con aplicaciones externas

Conpas ofrece servicios de consultoría, configuración, formación, desarrollos y soporte sobre este software de código libre. Consulte nuestra **integración de Zoho CRM con Odoo**.

Aplicaciones. Odoo Apps.

Odoo cuenta con una completa suite de módulos, Apps, que facilitan la gestión de todo el ciclo empresarial.

A continuación, veremos en detalle cada una de estos módulos.

Apps Odoo ventas

Gestione todo el ciclo de ventas de sus clientes con Odoo CRM y TPV. Con este sistema podrá monitorizar los resultados de sus comerciales y el desempeño de sus productos y servicios.

TPV

Completísimo software para Terminal Punto de Venta compatible con todo tipo de software (ordenadores de sobremesa, portátiles, tablets o hardware específico).

Extremadamente flexible y fácil de configurar y adatable a todo tipo de empresas minoristas.

CRM

Gestión de leads, cuentas, contactos y oportunidades de manera visual, intuitiva y sencilla.

Control de la agenda diaria: tareas, llamadas, reuniones, emails…

Completos informes de la situación de la organización.

Ventas

Herramienta online para la creación de propuestas que permite el envío directo al cliente, negociación de detalles y aceptación y pago por parte de este sobre la propia plataforma.

Suscripciones

Simplifica la gestión de las suscripciones: facturación recurrente y gestión de abonados de manera sencilla.

Facilita la creación de relaciones duraderas de con los clientes, optimizando el desempeño de la fuerza de ventas proporcionando herramientas de análisis para optimizar el negocio.

Apps Odoo finanzas

Odoo incluye el plan General Contable Español (PGCE) 2008 y el de los principales países.

Contabilidad

Gestión de la contabilidad adaptada a la legislación española.

Sincronización bancaria, facturación, control de cuentas, gastos…

Realice facturas electrónicas, informe de ganancias , balances o estados de flujos de efectivo.

Facturación

Facturación online sencilla: control de contactos, creación de facturas recurrentes, conversión de presupuestos en facturas, análisis de ventas…

Gastos

Control de gastos de los diarios empleados (gastos de viaje, dietas…) en una sola herramienta.

Apps Odoo par la gestión de operaciones logísticas y de desarrollo del negocio.

Inventario

Maximización de la eficiencia del almacén: mejora del rendimiento, minimización del stock, aumento de la rotación.

Automatización de funciones, control de la trazabilidad del proceso, generación de informes…

Imputación de horas

Control de las actividades realizadas y obtención de pronosticos de productividad.

Proyectos

Gestión de proyectos: organiza, programa, planea y analiza sus proyectos.

Realiza pronósticos de necesidades y recursos.

Compras

Gestión de proveedores y ordenes de compra: automatización del flujo de trabajo de compras, control de productos y facturas.

Apps Odoo operaciones con productos

MRP (Planificación de requerimientos de material)

Sistema de planificación y administración, orientado a tener los materiales requeridos en el momento oportuno para cumplir con las demandas de los clientes.

PLM (Administración del ciclo de vida de los productos)
Gestión del ciclo completo de vida del producto, desde la concepción, pasando por el análisis y la optimización, llegando al análisis de cómo se va a producir y dar mantenimiento.

Mantenimiento

Automatización preventiva del mantenimiento y organización del mismo. Aumente la eficiencia del proceso.

Control de calidad

Control de calidad con hitos y alertas.

Apps Odoo para sitios web

Integre Odoo ERP con su red de sitios web: páginas, blogs, intranet…

Sitio web

Odoo posibilita la creación de páginas optimizadas para SEO de manera sencilla, simple e intuitiva.

Tienda online

Creación de ecommerce atractivos, optimizados y conectados con todas las aplicaciones de Odoo.

Blogs y foros

Diseño de portales web para blogs y foros. Atractivos, optimizados e integrados con las RRSS.

Apps Odoo para marketing

El oro está en los datos. Monitorice todas sus acciones y obtenga su ROI de manera sencilla e intuitiva.

Email Marketing

Solución para el envío de campañas de email a nuestros contactos concetada con todas las aplicaciones de Odoo.

Eventos

Creación y gestión de eventos online.

Promociones y venta de entradas online.

Chat en vivo

Aplicación con la que incluiremos una ventana para atender en tiempo real a nuestros clientes desde nuestra página web.

Encuentas

Diseño de atractivas encuestas: satisfacción del cliente, campañas de marketing…

Apps Odoo RRHH

Mejore sus equipo de trabajo con una completa suite de apps de recursos humanos.

Reclutamiento

Portal de reclutamiento personalizable, adaptado a las diferentes necesidades de las empresas.

Empleados

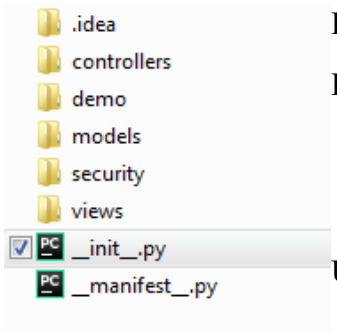
Gestión de los RRHH de la empresa: centralización de la información en una única herramienta. Retención del talento y control del desempeño.

Vacaciones y ausencias

Gestión de las vacaciones de toda la organización. Solicitudes, aceptación o rechazo. Control de las ausencias.

Fuente: https://www.conpas.net/odoo.html#ventas

CREACIÓN DE MODULOS EN ODOO 10/11



Para crear un módulo con una estructura vacía:

Primero he de situarme en:

C:\Program Files (x86)\Odoo 10.0\server\
C:\Program Files (x86)\Odoo 11.0\server\

Una vez ahí escribir la sentencia:

odoo-bin scaffold nombremodulo odoo/addons

odoo-bin scaffold nombremodulo d:\proyectos/addons

Para usar el comando scaffolding en la versión 11 esta es la orden, escrita tal cual (con comillas y todo):

"c:\Program Files (x86)\Odoo 11.0\python\python.exe" "c:\Program Files (x86)\Odoo 11.0\server\odoo-bin" scaffold *module directory*

"c:\Program Files (x86)\Odoo 11.0\python\python.exe" "c:\Program Files (x86)\Odoo 11.0\server\odoo-bin" scaffold nombremodulo d:\proyectos/addons

Los modulos por defecto se guardan dentro de:

C:\Program Files (x86)\Odoo 10.0\server\odoo\addons

Si quiero guardarlos en otra ubicación y que luego odoo los reconozca, he de modificar el archivo:

C:\Program Files (x86)\Odoo 10.0\server\odoo.conf

```
[options]
addons_path = C:\Program Files (x86)\Odoo 10.0\server\odoo\addons,C:\Program Files
(x86)\Odoo 10.0\server\odoo\addons\custom-addons,D:\Proyectos\addons
admin_passwd = admin
.....
```

Teniendo claro todo eso, es hora de ponerse a trabajar.....

Para llevar un mayor control de lo que hacemos, tenemos el archivo **odoo.log** que se encuentra en la carpeta Odoo/server

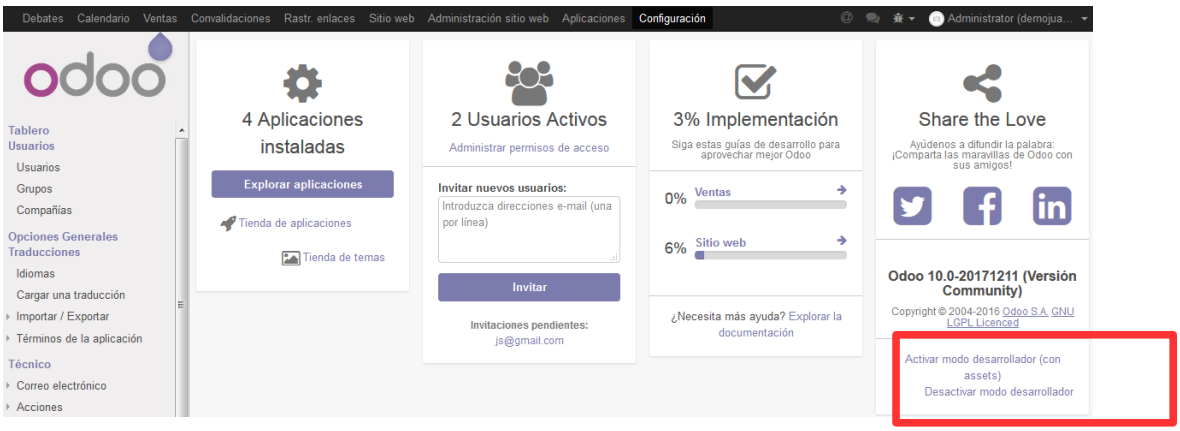
Esto de pende de los servicios de odoo (odoo-server-11.0). Para controlarlo mejor, vamos a detener este servicio en windows, nos vamos a la carpeta server de odoo y suponiendo que previamente tengo instalado Git, le doy al botón derecho en un lugar vacío de la ventana y ejecuto: **Git Bash Here** seme abre una ventana en la que tendré que poner: **./odoo-bin** y apartir de ahí en dicha ventana se me muestra el log del servicio odoo....

Para parar el servicio, he de emplear la combinación de teclas **Ctrl + C** (con eso se para) y para volverlo a iniciar, vuelvo a iniciarlo con el comando de antes.

Si no lo hiciera así, tendría que abrir el archivo odoo.log con un editor de texto e ir actualizando para ver los errores.

Para Odoo 11 tengo que hacerlo desde la carpeta Python

Cuando trabajemos con Odoo, para cargar un nuevo modulo, hay que darle a actualizar lista de aplicaciones dentro del apartado Aplicaciones, pero previamente, hay que activar el modo desarrollador dentro del apartado Configuración (situado en la parte inferior derecha)



ESTRUCTURA DE UN MODELO DE APLICACIÓN DE ODOO

Los modelos describen los objetos de negocio, como una oportunidad, una orden de venta, o un socio (cliente, proveedor, etc). Un modelo tiene una lista de atributos y también puede definir su negocio específico.

Los modelos son implementados usando clases Python derivadas de una plantilla de clase de Odoo. Estos son traducidos directamente a objetos de base de datos, y Odoo se encarga de esto automáticamente cuando el módulo es instalado o actualizado.

__init__.py

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

from . import controllers
from . import models
```

__manifest__.py

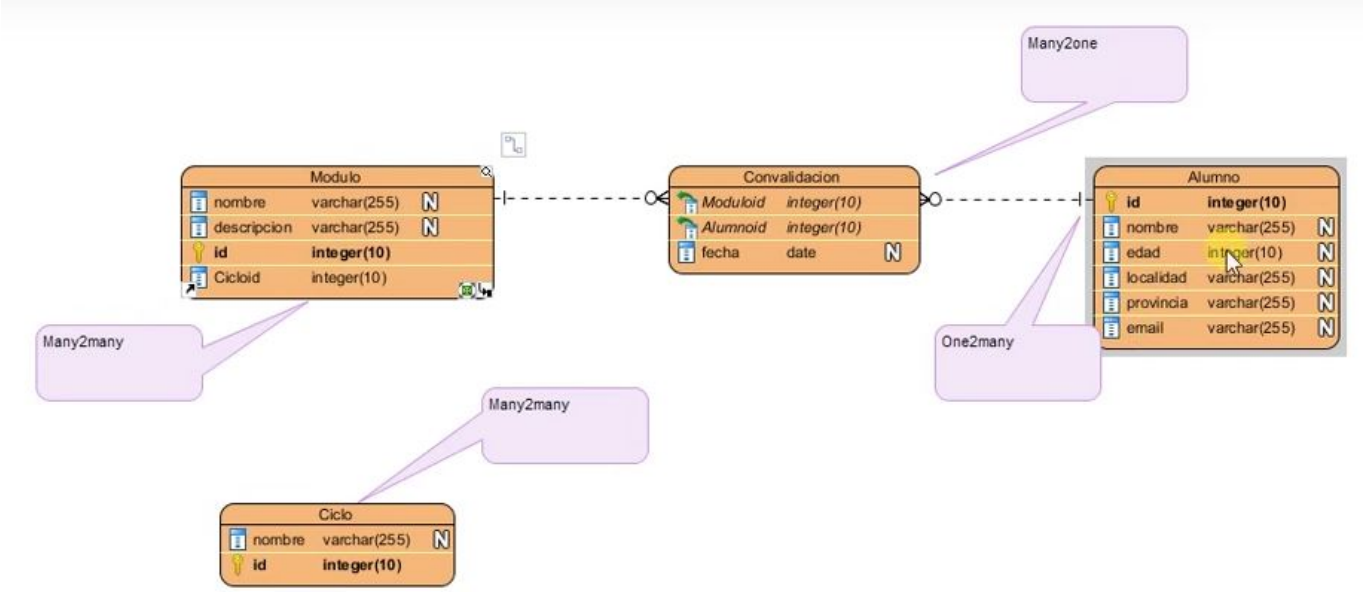
```
# -*- coding: utf-8 -*-
{
    'name': "Convalidaciones",

    'summary': """
        Permite gestionar las convalidaciones en un Instituto""",

    'description': """
        Este módulo permite gestionar las convalidaciones de un centro educativo
    """,

    'author': "Victor",
    'website': "http://www.example.com",

    'category': 'Uncategorized',
    'version': '0.1',

    'depends': ['base'],
```

```
    'data': [
        'views/alumnos.xml',
        'views/modulos.xml',
        'views/ciclos.xml',
        'views/convalidaciones.xml',
    ],
}
```

Muy Importante!!!!! cada mez que haga un cambio en model.py he re reiniciar el servicio de odoo desde services.msc... porque sino me dará error al compilar el modulo.

Si hago cambio en las vistas (templates.xml), simplemente actualizo el módulo.

DIAGRAMA DE RELACIONES Y DE CLASES



models/models.py

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

from odoo import models, fields, api

class Alumno(models.Model):
    _name = 'convalidaciones.alumno'

    name = fields.Char(string='Nombre y apellidos')
    edad = fields.Integer(string='Edad')
    localidad = fields.Char(string='Localidad')
    provincia = fields.Char(string='Provincia')
    email = fields.Char(string='Email')
    convalidacion_ids = fields.One2many('convalidaciones.convalidacion', 'alumno_id', string='Convalidaciones del alumno')

class Modulo(models.Model):
    _name = 'convalidaciones.modulo'

    name = fields.Char(string='Nombre')
    description = fields.Text(string='Descripción')
    ciclo_ids = fields.Many2many('convalidaciones.ciclo')
```

```
class Convalidacion(models.Model):
    _name = 'convalidaciones.convalidacion'

    name = fields.Char()
    fecha_convalidacion = fields.Date(string='Fecha de la convalidación')
    modulo_id = fields.Many2one('convalidaciones.modulo', string='Módulo')
    alumno_id = fields.Many2one('convalidaciones.alumno', string='Alumno')

class Ciclos(models.Model):
    _name = 'convalidaciones.ciclo'

    name = fields.Char(string='Nombre')
    description = fields.Text(string='Descripción')
    modulo_ids = fields.Many2many('convalidaciones.modulo')
```

En models es donde se definen las clases que va a llevar mi módulo.... **Importante!!!!** cuando defino una clase, tan solo ha de haber una tabulación entre class y los objetos que contienen dicha clase, sino dará error.

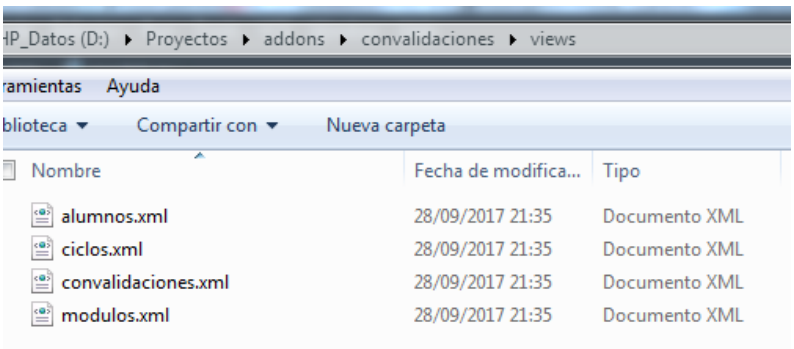
Puedo o definirlas todas dentro del archivo models.py o crear archivos independientes para mis clases por ejemplo: alumno.py, modulo.py, convalidación.py

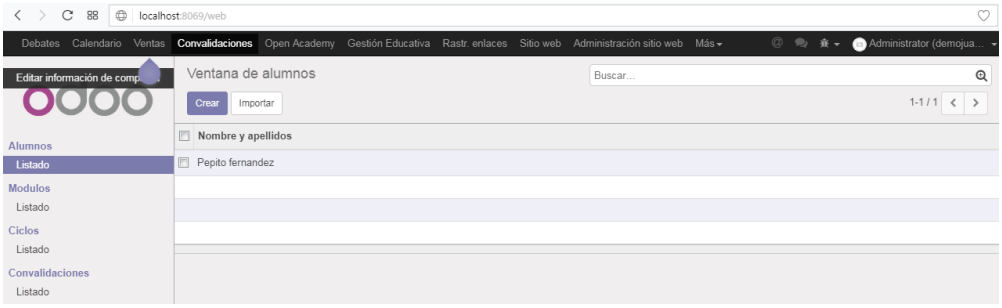
Si hago esto, tengo último, tengo que tener la precaución de añadir la clase al __init__.py que está dentro de la carpeta models.

Una vez que tengo mis clases creadas, tengo que definir la apariencia que estas van a tener. Eso se hace dentro de la carpeta **views**. Si tengo varias clases independientes, deberé de crear una vista para cada una. Siempre me puedo basar en el modelo de ejemplo que me crea odoo con la orden **scaffold.**

Una vez creadas mis vistas, tendré que añadirlas al fichero __manifest__.py para que se tengan en cuenta, dentro de:

```
data': [
        'views/alumnos.xml',
        'views/modulos.xml',
.....................
```

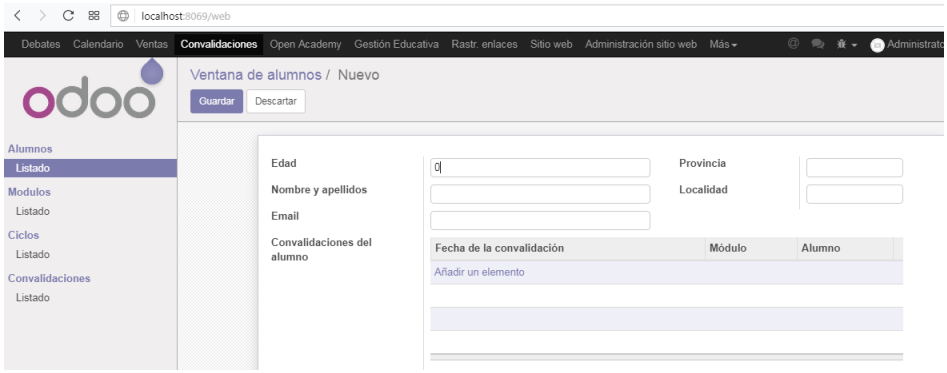




view/alumnos.xml

```
<odoo>
  <data>

    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="convalidaciones.list_alumnos_tree">
      <field name="name">convalidaciones.alumno.tree</field>
      <field name="model">convalidaciones.alumno</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name"/>
          <field name="edad"/>
          <field name="localidad"/>
          <field name="provincia"/>
          <field name="email"/>
          <field name="convalidacion_ids">
            <tree string="Convalidaciones del alumno">
              <field name="modulo_id" />
              <field name="alumno_id" />
              <field name="fecha_convalidacion" />
            </tree>
          </field>
        </tree>
      </field>
    </record>
    <record model="ir.ui.view" id="convalidaciones.list_alumnos_form">
      <field name="name">convalidaciones.alumno.form</field>
      <field name="model">convalidaciones.alumno</field>
      <field name="arch" type="xml">
        <form>
          <sheet>
            <div class="oe_title">
              <label for="name" class="oe_edit_only" string="Nombre y
apellidos">Nombre y apellidos</label>
              <h1><field name="name"/></h1>
            </div>
            <separator string="Datos personales" colspan="2" />
            <group colspan="2" col="2">
              <field name="edad"/>
              <field name="localidad"/>
              <field name="provincia"/>
              <field name="email"/>
            </group>
            <field name="convalidacion_ids"/>
          </sheet>
        </form>
      </field>
    </record>
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="convalidaciones.action_window">
      <field name="name">Ventana de alumnos</field>
      <field name="res_model">convalidaciones.alumno</field>
      <field name="view_mode">tree,form</field>
    </record>
    <!-- Top menu item -->
    <menuitem name="Convalidaciones" id="convalidaciones.menu_root"/>
    <!-- menu categories -->
    <menuitem name="Alumnos" id="convalidaciones.alumnos"
parent="convalidaciones.menu_root"/>
    <!-- actions -->
    <menuitem name="Listado" id="convalidaciones.alumnos_list"
parent="convalidaciones.alumnos"
```

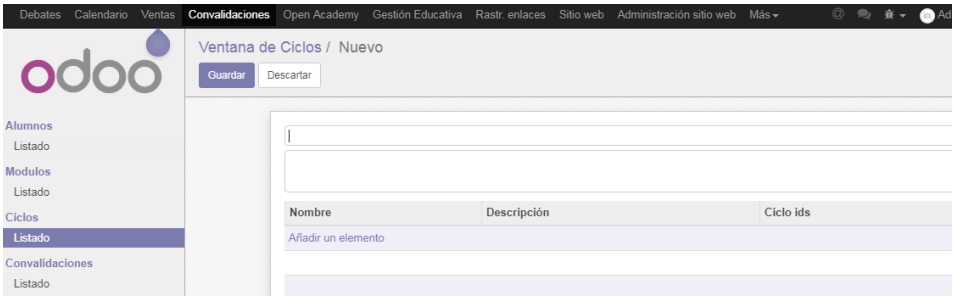
```
            action="convalidaciones.action_window"/>
  </data>
</odoo>
```



view/ciclos.xml

```
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="convalidaciones.list_alumnos_tree">
      <field name="name">convalidaciones.ciclos.listado.tree</field>
      <field name="model">convalidaciones.ciclo</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name"/>
          <field name="description"/>
          <field name="modulo_ids">
            <tree string="Modulos en los que se imparten este ciclo">
              <field name="name" />
              <field name="description" />
            </tree>
          </field>
        </tree>
      </field>
    </record>
    <record model="ir.ui.view" id="convalidaciones.list_alumnos_form">
      <field name="name">convalidaciones.ciclos.listado.form</field>
      <field name="model">convalidaciones.ciclo</field>
      <field name="arch" type="xml">
        <form string="Ventana del ciclo">
          <sheet>
            <field name="name"/>
            <field name="description"/>
            <field name="modulo_ids"/>
          </sheet>
        </form>
      </field>
    </record>
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window"
id="convalidaciones.action_window_ciclos">
      <field name="name">Ventana de Ciclos</field>
      <field name="res_model">convalidaciones.ciclo</field>
      <field name="view_mode">tree,form</field>
    </record>
    <!-- menu categories -->
    <menuitem name="Ciclos" id="convalidaciones.ciclos"
parent="convalidaciones.menu_root"/>
```

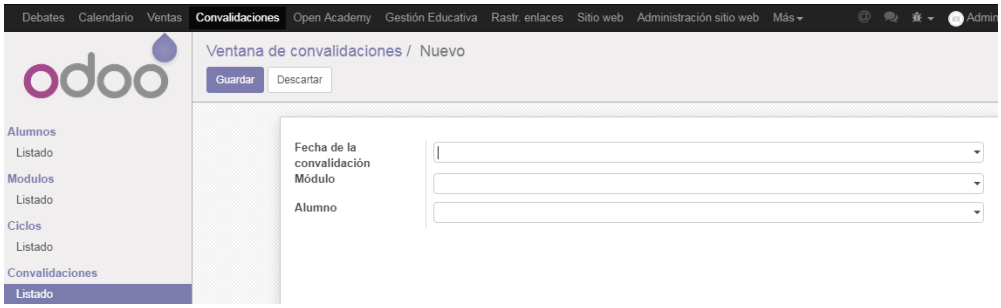
```
    <!-- actions -->
    <menuitem name="Listado" id="convalidaciones.ciclos_list"
parent="convalidaciones.ciclos"
              action="convalidaciones.action_window_ciclos"/>
  </data>
</odoo>
```



view/convalidaciones.xml

```
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view"
id="convalidaciones.list_convalidaciones_tree">
      <field name="name">convalidaciones.convalidaciones.tree</field>
      <field name="model">convalidaciones.convalidacion</field>
      <field name="arch" type="xml">
        <tree>
          <field name="fecha_convalidacion"/>
          <field name="modulo_id"/>
          <field name="alumno_id"/>
        </tree>
      </field>
    </record>
    <record model="ir.ui.view"
id="convalidaciones.list_convalidaciones_form">
      <field name="name">convalidaciones.convalidaciones.form</field>
      <field name="model">convalidaciones.convalidacion</field>
      <field name="arch" type="xml">
        <form>
          <sheet>
            <group colspan="2" col="2">
              <field name="fecha_convalidacion"/>
              <field name="modulo_id"/>
              <field name="alumno_id"/>
            </group>
          </sheet>
        </form>
      </field>
    </record>
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window"
id="convalidaciones.action_window_convalidaciones">
      <field name="name">Ventana de convalidaciones</field>
      <field name="res_model">convalidaciones.convalidacion</field>
      <field name="view_mode">tree,form</field>
    </record>
    <!-- menu categories -->
    <menuitem name="Convalidaciones"
id="convalidaciones.menu_convalidaciones"
parent="convalidaciones.menu_root"/>
```

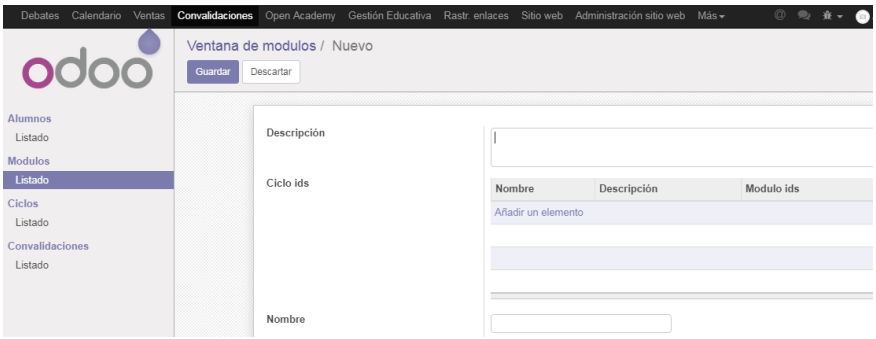
```
    <!-- actions -->
    <menuitem name="Listado" id="convalidaciones.convalidaciones_list"
parent="convalidaciones.menu_convalidaciones"
              action="convalidaciones.action_window_convalidaciones"/>
  </data>
</odoo>
```



view/modulos.xml

```
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="convalidaciones.list_modulos_tree">
      <field name="name">convalidaciones.modulos.listado.tree</field>
      <field name="model">convalidaciones.modulo</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name"/>
          <field name="description"/>
          <field name="ciclo_ids">
            <tree string="Ciclos en los que se imparte este módulo">
              <field name="name"/>
              <field name="description"/>
            </tree>
          </field>
        </tree>
      </field>
    </record>
    <record model="ir.ui.view" id="convalidaciones.list_modulos_form">
      <field name="name">convalidaciones.modulos.listado.form</field>
      <field name="model">convalidaciones.modulo</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name"/>
          <field name="description"/>
          <field name="ciclo_ids"/>
        </tree>
      </field>
    </record>
    <!-- actions opening views on models -->
    <record model="ir.actions.act_window"
id="convalidaciones.action_window_modulos">
      <field name="name">Ventana de modulos</field>
      <field name="res_model">convalidaciones.modulo</field>
      <field name="view_mode">tree,form</field>
    </record>
    <!-- menu categories -->
    <menuitem name="Modulos" id="convalidaciones.modulos"
parent="convalidaciones.menu_root"/>
    <!-- actions -->
```

```
    <menuitem name="Listado" id="convalidaciones.modulos_list"
parent="convalidaciones.modulos"
              action="convalidaciones.action_window_modulos"/>
  </data>
</odoo>
```



Para lanzar tests unitarios en ODOO

Versión 10

Añadiendo pruebas automatizadas

Las mejores prácticas de programación incluyen tener pruebas automatizadas para tu código. Esto es aún más importante para lenguajes dinámicos como Python. Como no hay ningún paso de compilación, no puede estar seguro de que no haya errores sintácticos hasta que el intérprete realmente ejecute el código. Un buen editor puede ayudarnos a detectar estos problemas con antelación, pero no puede ayudarnos a asegurar que el código se ejecute como lo desean las pruebas automatizadas.

Odoo soporta dos formas de describir las pruebas: ya sea utilizando archivos de datos YAML o utilizando código Python, basado en la biblioteca `Unittest2`. Las pruebas YAML son un legado de versiones anteriores, y no se recomiendan. Preferiremos usar pruebas de Python y añadiremos un caso básico de prueba a nuestro módulo.

Los archivos de código de prueba deben tener un nombre que empiece por `test_` y se debe importar desde `tests / __ init__.py`. Pero el directorio de `test` (o submódulo Python) no se debe importar desde la parte superior del módulo `__init__.py`, ya que se descubrirá y cargará automáticamente sólo cuando se ejecuten pruebas.

Las pruebas deben colocarse en un subdirectorio `test/`. Añade un archivo `tests / __ init__.py` con lo siguiente:

```
from . import test_todo
```

Ahora, añade el código de prueba real disponíble en el archivo `tests/test_todo.py`:

```
# -*- coding: utf-8 -*-
from odoo.tests.common import TransactionCase

class TestTodo(TransactionCase):

    def test_create(self):
        "Create a simple Todo"
        Todo = self.env['todo.task']
```

```
        task = Todo.create({'name': 'Test Task'})
        self.assertEqual(task.is_done, False)
```

Esto agrega un caso simple de prueba para crear una nueva tarea y verifica que el campo **Is Done?**Tiene el valor predeterminado correcto.

Ahora queremos hacer nuestras pruebas. Esto se hace agregando la opción `--test-enable` durante la instalación del módulo:

`Para Linux:`

```
$ ./odoo-bin -d todo -i todo_app –test-enable --stop-after-init
```

`Para Windows:`

```
odoo-bin.exe -d todo -i todo_app –test-enable
```

```
odoo-bin.exe -d gestioneducativa -i gestioneducativa_v1 –test-enable
```

El servidor Odoo buscará un subdirectorio tests/ en los módulos actualizados y los ejecutará. Si alguna de las pruebas falla, el registro del servidor te mostrará eso.

Versión 11

Ver web oficial:
https://www.odoo.com/documentation/11.0/reference/testing.html

Tests are automatically run when installing or updating modules if **`--test-enable`** was enabled when starting the Odoo server.

```
51  test_commit = False
52  test_enable = True
53  test_file = False
54  test_report_directory = D:\Proyectos\Tests
```