Manual modulo custom con scaffold

Estructura del proyecto

```
└── 📁 odoo_docker
        └── 📁 config_odoo
                └── odoo.conf
        └── 📁 dev_addons
        └── docker-compose.yml
        └── 📁 log
        └── notes.txt
```

`odoo.conf`

```
[options]
admin_passwd = 123456
addons_path = /mnt/extra-addons
db_password = odoo_db
db_user = odoo_db
db_host = postgres_16
db_port = 5432
http_port = 8069
```

- **[options]:** Esta línea indica el inicio de la sección de configuración de Odoo. Todas las opciones bajo esta línea son parte de la configuración de Odoo.
-
- **admin_passwd = 123456:** Esta es la contraseña del superusuario de Odoo. Se utiliza para realizar operaciones administrativas importantes, como actualizar la lista de módulos o realizar cambios en la configuración del sistema a través de la interfaz web.
- **addons_path = /mnt/extra-addons:** Especifica la ruta (o rutas, separadas por comas) donde Odoo buscará módulos adicionales. Esto permite a Odoo cargar módulos personalizados o de terceros que no están incluidos en la instalación estándar. En tu caso, apunta a `/mnt/extra-addons`, lo que sugiere que en el contenedor de Odoo, este directorio está montado para contener dichos módulos.
- **db_password = odoo_db:** La contraseña utilizada por Odoo para conectarse a la base de datos PostgreSQL. En este contexto, `odoo_db` es la contraseña que has configurado para el usuario de la base de datos.

- **db_user = odoo_db:** El nombre de usuario utilizado por Odoo para conectarse a la base de datos PostgreSQL. Debe coincidir con un rol (usuario) existente en PostgreSQL que tenga permisos adecuados para realizar operaciones en la base de datos de Odoo.
- **db_host = postgres_16:** El nombre del host donde se está ejecutando el servidor de base de datos PostgreSQL. En un entorno de Docker, este nombre debería coincidir con el nombre del servicio definido en `docker-compose.yml` para PostgreSQL. En tu caso, parece que deberías cambiar esto a `db` para que coincida con el nombre del servicio en tu archivo `docker-compose.yml`.
- **db_port = 5432:** El puerto a través del cual Odoo se conectará al servidor de PostgreSQL. `5432` es el puerto predeterminado para PostgreSQL.
- **http_port = 8069:** Especifica el puerto en el que el servidor web de Odoo escuchará las conexiones entrantes. Por defecto, Odoo utiliza el puerto `8069` para la interfaz web.

En resumen, este archivo `odoo.conf` configura las credenciales de la base de datos, la ruta para módulos adicionales, el puerto del servidor web de Odoo, y la contraseña del superusuario. Para que funcione correctamente en tu entorno Docker, necesitarías ajustar `db_host` para que coincida con el nombre del servicio de PostgreSQL en tu `docker-compose.yml`, que probablemente sea `db` y no `postgres_16`, a menos que tengas una configuración de nombres específica en tu red de Docker.

`docker-compose.yml`

```
version: '3.9'

services:
  web:
    container_name: odoo_16
    image: odoo:16.0
    depends_on:
      - db
    ports:
      - "8069:8069"
    volumes:
      - odoo-web-data:/var/lib/odoo
      - ./config_odoo:/etc/odoo
      - ./dev_addons:/mnt/extra-addons
      - ./log:/var/log/odoo

  db:
    container_name: postgres_16
    image: postgres:16.0
    ports:
      - 5432:5432/tcp
```

```
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_PASSWORD=odoo_db
      - POSTGRES_USER=odoo_db
      - PGDATA=/var/lib/postgres/data/pgdata
    volumes:
      - odoo-db-data:/var/lib/postgresql/data/pgdata
    healthcheck:
      test: ["CMD", "pg_isready"]
      interval: 10s
      timeout: 5s
      retries: 5

volumes:
  odoo-web-data:
  odoo-db-data:
```

Crear un nuevo modulo con el comando scaffold

Paso 1: Acceder al Contenedor de Odoo

Primero, necesitas obtener acceso al terminal dentro del contenedor de Odoo donde quieres crear el módulo. Esto se hace usando el comando `docker exec` junto con la opción `-it` para permitir una interacción interactiva, seguido del nombre o ID del contenedor de Odoo y el comando para ejecutar, que en este caso sería una shell como `/bin/bash`.

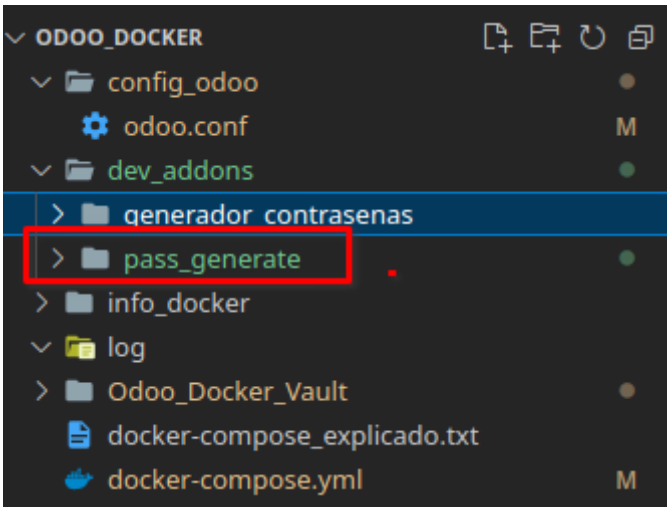
`docker exec -it odoo_16 /bin/bash`

Paso 2: Ejecutar el Comando Scaffold

Con el comando `scaffold`, puedes generar la estructura básica de un nuevo módulo. Odoo proporciona este comando para facilitar el inicio de un nuevo módulo con los archivos y directorios estándar que normalmente necesitarás.

Ejecuta el comando `scaffold` como sigue, reemplazando `pass_generate` con el nombre que deseas darle a tu nuevo módulo:

```
/usr/bin/odoo scaffold pass_generate /mnt/extra-addons
```



Paso extra para Linux:

```
chmod 777 -R /mnt/extra-addons/pass_generate/
```

Explicación de la estructura del modulo

La estructura que has creado con el comando `scaffold` para tu módulo `pass_generate` en Odoo sigue una estructura estándar que facilita el desarrollo, la organización y la distribución de módulos en Odoo. Aquí te explico el propósito de cada componente dentro de tu módulo:

`__init__.py`

Los archivos `__init__.py` son archivos especiales de Python que permiten que Python trate los directorios que contienen como paquetes. Esto significa que para poder importar cualquier archivo de Python como un módulo, el directorio debe contener un archivo `__init__.py`. En el contexto de Odoo, estos archivos se utilizan para importar los módulos Python de las subcarpetas como `models` y `controllers`, para que Odoo los reconozca y los cargue correctamente.

- En la raíz del módulo, `__init__.py` típicamente importa los modelos y los controladores usando líneas como `from . import models, controllers`.
- Dentro de `models` y `controllers`, `__init__.py` importa los archivos específicos, por ejemplo, `from . import models.py` o `from . import controllers.py`.

`__manifest__.py`

El archivo `__manifest__.py` (anteriormente conocido como `__openerp__.py` en versiones antiguas de Odoo) contiene metadatos sobre tu módulo. Esto incluye información como el nombre del módulo, descripción, autor, versión, dependencias de otros módulos de Odoo que tu módulo requiere para funcionar, datos que deben ser instalados o actualizados con el

módulo (como archivos XML o CSV), y otras configuraciones. Es esencial para que Odoo sepa cómo manejar tu módulo.

`models/models.py`

Aquí defines los modelos de datos de tu módulo. Los modelos en Odoo son representaciones de estructuras de datos, y cada modelo corresponde generalmente a una tabla en la base de datos de Odoo. Este archivo es donde defines los campos, métodos y comportamientos de tus modelos, utilizando la API de Odoo.

`controllers/controllers.py`

Este archivo define los controladores para tu módulo, que son puntos de entrada para las solicitudes HTTP. Los controladores pueden ser utilizados para crear páginas web front-end, responder a solicitudes AJAX, etc. En este archivo, puedes definir rutas (URLs) y cómo deben ser manejadas.

`demo/demo.xml`

Los archivos en la carpeta `demo` contienen datos de demostración que pueden ser cargados para demostrar las funcionalidades de tu módulo sin necesidad de configuración adicional. Estos datos son especialmente útiles durante el desarrollo o para demostraciones a clientes potenciales.

`security/ir.model.access.csv`

Este archivo CSV define las reglas de acceso para los modelos definidos en tu módulo. En Odoo, las reglas de acceso son importantes para asegurar que solo los usuarios autorizados puedan crear, leer, actualizar o eliminar registros en los modelos. Este archivo lista esas reglas, especificando qué grupos de usuarios tienen qué permisos sobre cada modelo.

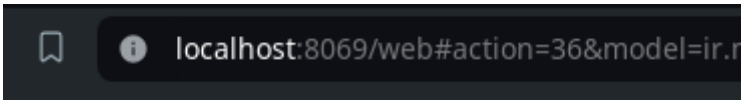
`views/templates.xml` y `views/views.xml`

Los archivos en la carpeta `views` definen cómo se presentan los modelos en la interfaz de usuario.

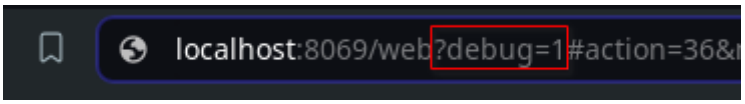
- `views.xml` suele contener la definición de las vistas de formulario, árbol, búsqueda y otras vistas que determinan cómo se muestran y se organizan los registros de modelos en la interfaz web de Odoo.
- `templates.xml` puede contener definiciones de plantillas QWeb, que son plantillas del lado del servidor utilizadas por Odoo para generar HTML dinámicamente. Estas plantillas pueden ser usadas para crear reportes, páginas web o componentes específicos de la UI.

Esta estructura está diseñada para ser modular y extensible, lo que permite que Odoo sea altamente personalizable y adaptable a una amplia gama de necesidades empresariales. Cada componente tiene su propósito específico y juntos forman la base sobre la cual puedes construir funcionalidades complejas dentro de tu módulo `pass_generate`.

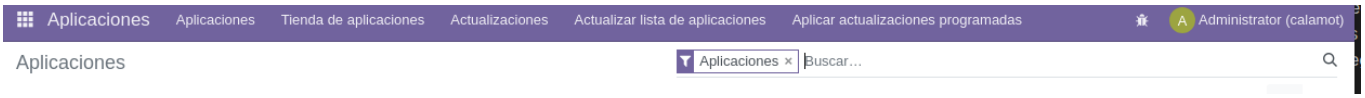
Activar el modo debug para desarrollar el modulo



Debemos modificar la url de esta forma:

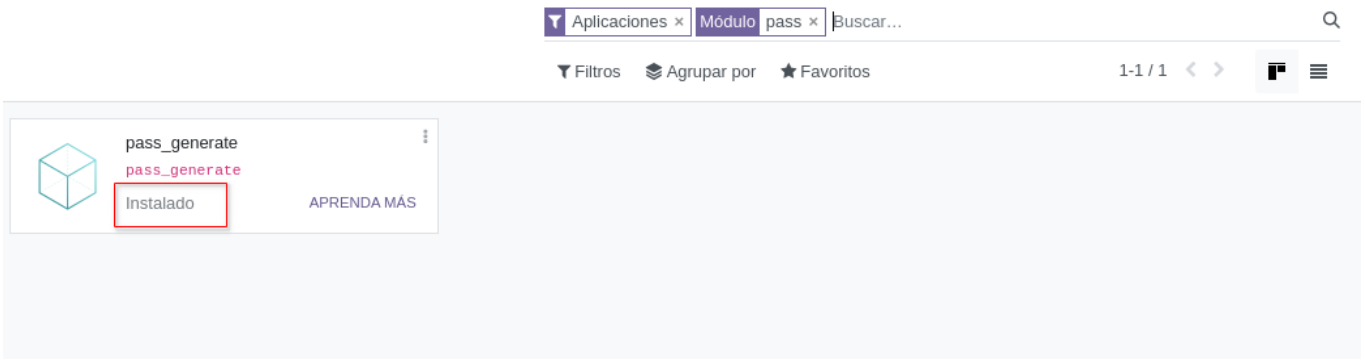


Ahora ya tenemos todas las opciones de desarrollador



Hacemos click en actualizar aplicaciones y ya podemos buscar el modulo por el nombre en la barra de busqueda:

Después de Activar el modulo ya podemos buscarlo dentro de la tienda de aplicaciones:



Modificar el modulo:

Vamos a modificar tu módulo `pass_generate` para que muestre un formulario simple. Este formulario podría ser para cualquier propósito que necesites, pero como no especificaste uno, vamos a crear un ejemplo básico que podría extenderse según tus necesidades.

Paso 1: Definir un Modelo

Primero, necesitarás definir un modelo (objeto de negocio) que tu formulario va a utilizar para mostrar y almacenar los datos. Por simplicidad, crearemos un modelo muy básico.

1. **Editar** `models/models.py`: Vamos a definir un modelo llamado `pass.generate`.

```
from odoo import models, fields

class test(models.Model):
    _name = 'pass_generate.test'
    _description = 'test'

    name = fields.Char()
    value = fields.Integer()
    value2 = fields.Float(compute="_value_pc", store=True)
    description = fields.Text()

    @api.depends('value')
    def _value_pc(self):
        for record in self:
            record.value2 = float(record.value) / 100
```

Este código define un modelo con dos campos: un `name`, que es un campo de texto corto y obligatorio, y una `description`, que es un campo de texto largo y opcional.

Paso 2: Definir una Vista

Después de definir el modelo, necesitas crear una vista para mostrar un formulario basado en ese modelo.

1. **Editar** `views/views.xml`: Define una vista de formulario para el modelo `pass.generate`.

```
<odoo>
        <record model="ir.ui.view" id="pass_generate.test_list">
                <field name="name">pass_generate test list</field>
                <field name="model">pass_generate.test</field>
                <field name="arch" type="xml">
                <tree>
                        <field name="name"/>
```

```
                        <field name="value"/>
                        <field name="value2"/>
                        </tree>
                </field>
        </record>
</odoo>
```

Esta vista define un formulario simple para tu modelo, mostrando campos para `name` y `description`.

Paso 3: Definir Acciones y Menús

Para acceder a tu formulario desde la interfaz de usuario de Odoo, necesitas definir una acción y un elemento de menú.

1. **Editar** `views/views.xml` para añadir una acción y un menú:

```
      <record model="ir.actions.act_window"
id="pass_generate.test_action_window">
      <field name="name">pass_generate test window</field>
      <field name="res_model">pass_generate.test</field>
      <field name="view_mode">tree,form</field>
    </record>

        <!-- Top menu item -->
    <menuitem name="pass_generate" id="pass_generate.menu_root"/>

        <!-- menu categories -->
        <menuitem name="Menu 1" id="pass_generate.test_menu_1"
parent="pass_generate.menu_root"/>

        <!-- actions -->
        <menuitem name="Test" id="pass_generate.test_test_menu_1_list"
parent="pass_generate.test_menu_1"

        action="pass_generate.test_action_window"/>
```

Esto crea una acción que abre la vista de formulario de tu modelo `pass.generate` y añade un elemento de menú en el menú principal de Odoo (puedes cambiar `parent="base.menu_custom"` por el identificador del menú donde quieras colocarlo).

Paso 4: Acceso a la Seguridad

Es importante definir reglas de acceso para tu nuevo modelo para asegurarte de que solo los usuarios autorizados puedan ver y modificar los datos.

1. **Editar** `security/ir.model.access.csv` para añadir reglas de acceso:

```
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_pass_generate_test,pass_generate.test,model_pass_generate_test,base.g
roup_user,1,1,1,1
```

Esto proporciona acceso CRUD (Crear, Leer, Actualizar, Eliminar) básico a todos los usuarios. Puedes restringir esto más adelante según sea necesario.

Paso 5: Actualiza --manifest__.py

```
'data': [
        'security/ir.model.access.csv',
        'views/views.xml',
        'views/templates.xml',
],
```

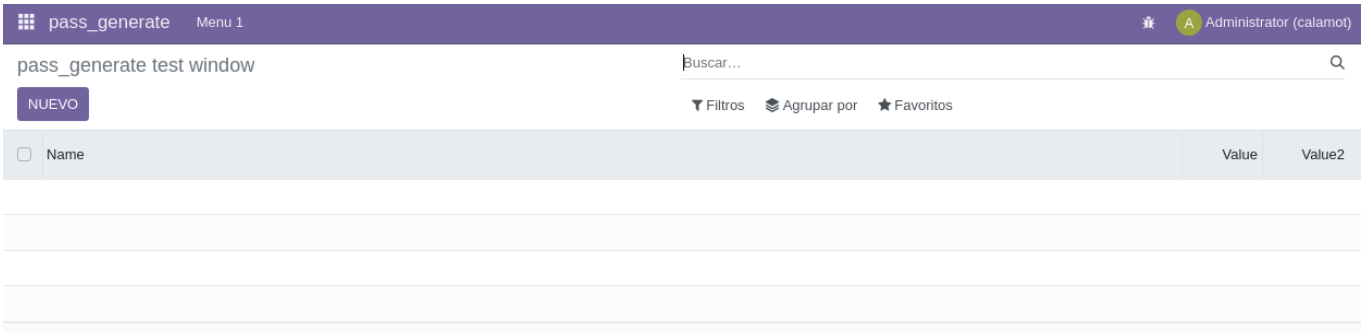
Paso 6: Actualizar el Módulo

Reinicia el contenedor :

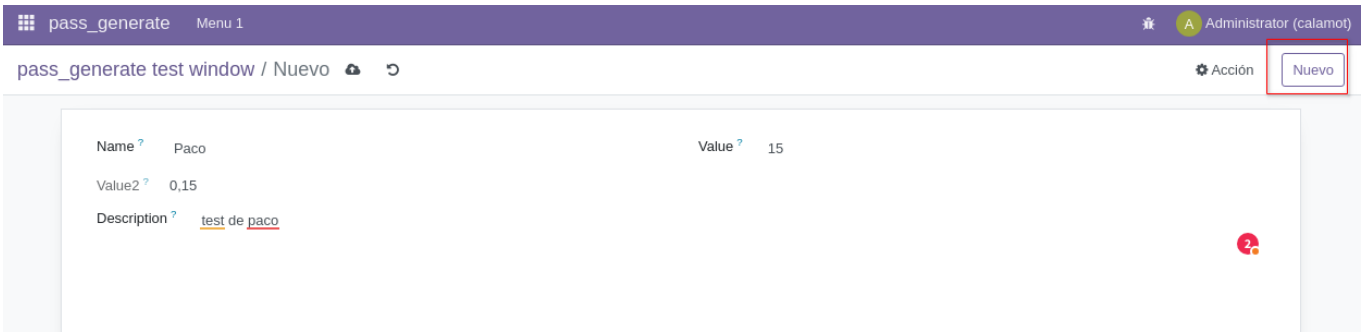
```
docker-compose up
```

Para que estos cambios surtan efecto, necesitas actualizar tu módulo en Odoo. Esto se puede hacer desde la interfaz de usuario en el menú "Aplicaciones" buscando tu módulo y haciendo clic en "Actualizar".

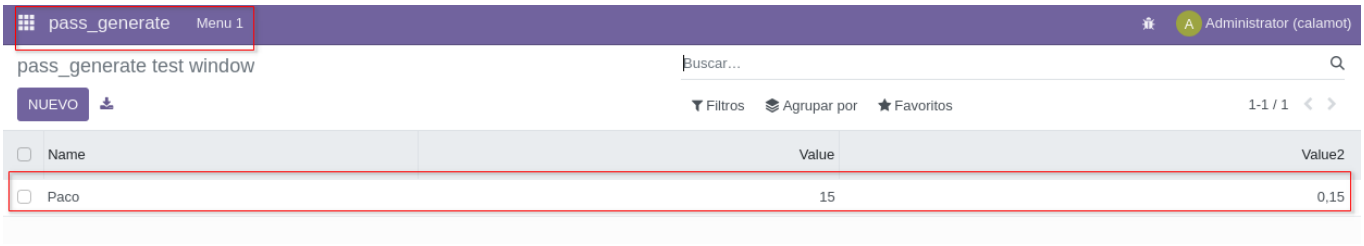
Si todo funciona deberías ver esta pantalla:



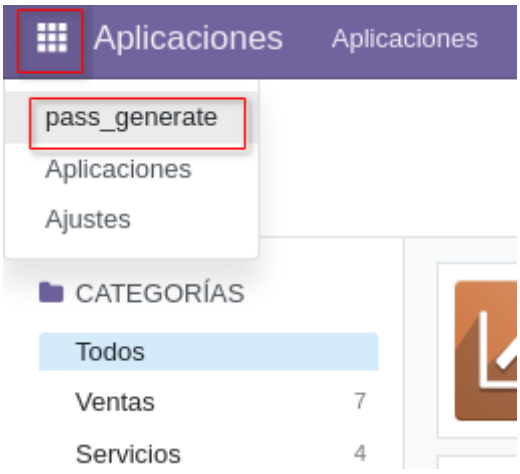
Haz click en el boton nuevo y ya puedes añadir tu primer registro



Comprueba que el registro se creo correctamente:



Tu aplicación instalada ya estará presente en tu conjunto de aplicaciones:



Conclusión

¡Y eso es todo! Ahora tienes un modelo básico `pass.generate` con un formulario para crear y editar registros. Puedes acceder a este formulario desde el menú que has definido. Este es un

punto de partida simple para cualquier funcionalidad que desees construir.

http://castilloinformatica.com/

https://www.youtube.com/playlist?list=PL5ESsgnHGfa8d3EetmuUA8quawtJjEiH4