**Bài 3: Tạo Application từ đầu cho Odoo**

Mục lục

[I. Lời mở đầu 1](#_Toc126958305)

[II.Yêu cầu kỹ thuật 1](#_Toc126958306)

[III. Tiến hành 1](#_Toc126958307)

[1. Chuẩn bị 1](#_Toc126958308)

[2. Odoo scaffold 1](#_Toc126958309)

[3. \_\_manifest\_\_.py 3](#_Toc126958310)

**[4. Bắt đầu viết application](#_Toc126958311)** [5](#_Toc126958311)

[5. Models 7](#_Toc126958312)

[6. Tạo menu 10](#_Toc126958313)

[7. Security Groups 11](#_Toc126958314)

[8. Access 15](#_Toc126958315)

[9. Backend view layer 16](#_Toc126958316)

[10. Bussiness Logic 20](#_Toc126958317)

[11. Web Controllers 22](#_Toc126958318)

[12. Chú ý 24](#_Toc126958319)

[IV. Phần kết 24](#_Toc126958320)

[V. Tham Khảo 24](#_Toc126958321)

1. Lời mở đầu

Bạn theo hướng dẫn ở Buổi 2 để tạo dựng Odoo version 16 nha. Tiếp theo bạn theo hướng dẫn trong bài viết này để tạo **application** đầu tiên của mình trong **Odoo**. Cuối bài viết mình sẽ đính kèm source code hoàn chỉnh cho việc tạo **application** trong **Odoo** của Buổi 3 này.

1. Yêu cầu kỹ thuật

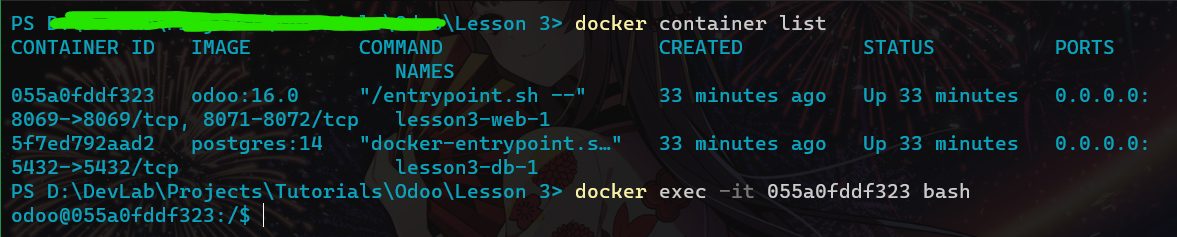
* Biết cơ bản về **Python**
* Biết về **Docker**, **docker-compose.yml**
* Thay đổi thông số config của **Odoo**
* Thêm **addon** vào **Odoo**

1. Tiến hành
2. Chuẩn bị

Một Server Odoo đang chạy.

1. Odoo scaffold

Trước tiên thì ta sẽ tương tác với container có mà Server Odoo đang hoạt động trong terminal

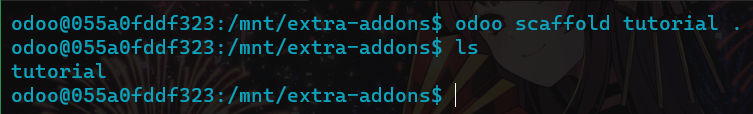


Vào đường dẫn **“ /mnt/extra-addons/ ”**



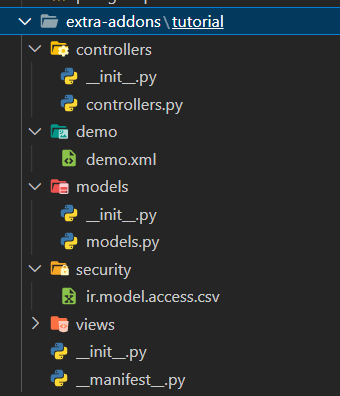
Chạy lệnh “**odoo scaffold <tên\_module\_muốn\_tạo> <đường/dẫn/chứa/module>**“

Trường hợp này, mình đã cd tới **/mnt/extra-addons** rồi nên đường dẫn ở đây là dấu chấm (**.**) (nghĩa là tại chỗ)





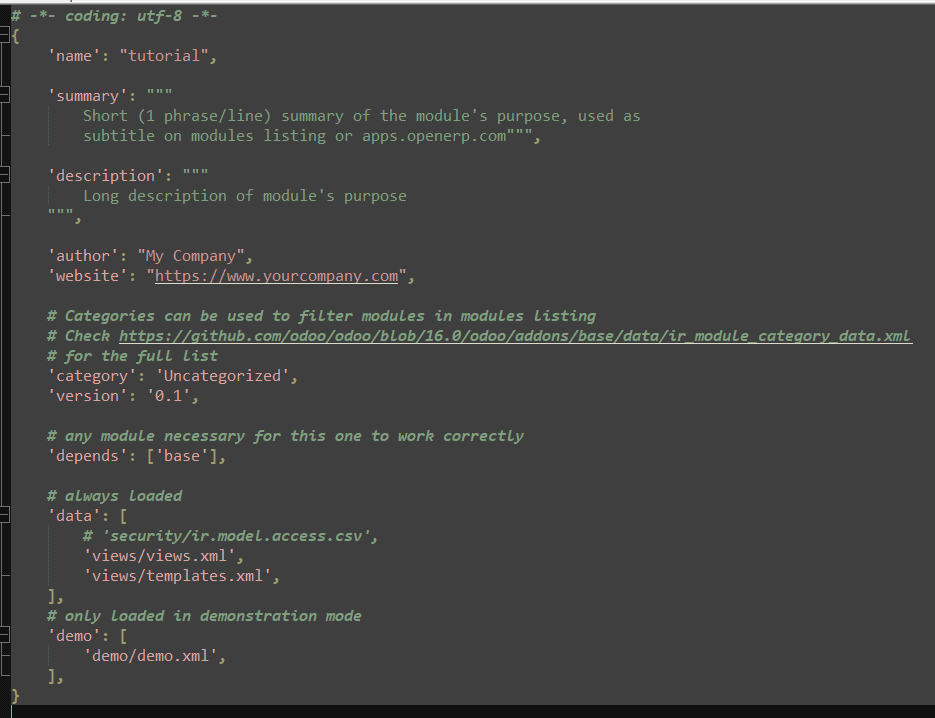
Kiểm tra ở folder máy của bạn



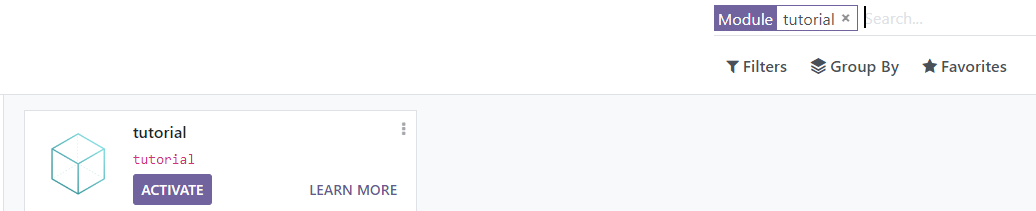
Được rồi, đây là công cụ tiện ích của Odoo giúp chúng ta tạo ra cái **template** của **module**.

1. \_\_manifest\_\_.py

Mở **\_\_manifest\_\_.py** lên ta sẽ có được nội dung như sau:

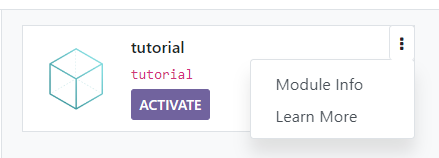


Khởi động lại odoo, cập nhật lại danh sách Apps và tìm module “tutorial”

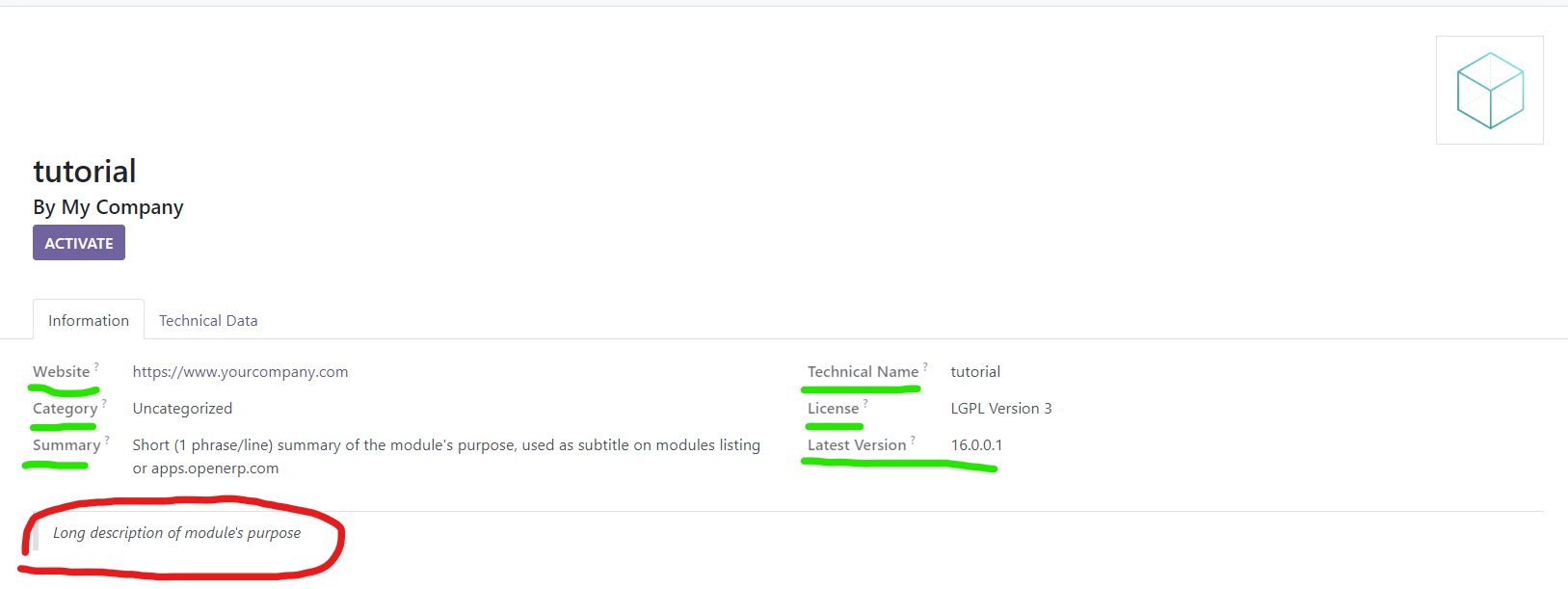


Lưu ý rằng đây mới là **module** chứ chưa phải **application**

Xem Module Info và đối chiếu với nội dung trong \_\_manifest\_\_.py



Cái hình trên (chữ tutorial đậm là “**name**” trong **\_\_manifest\_\_.py**, còn chữ **tutorial dưới** là cái tên **folder module** của chúng ta)

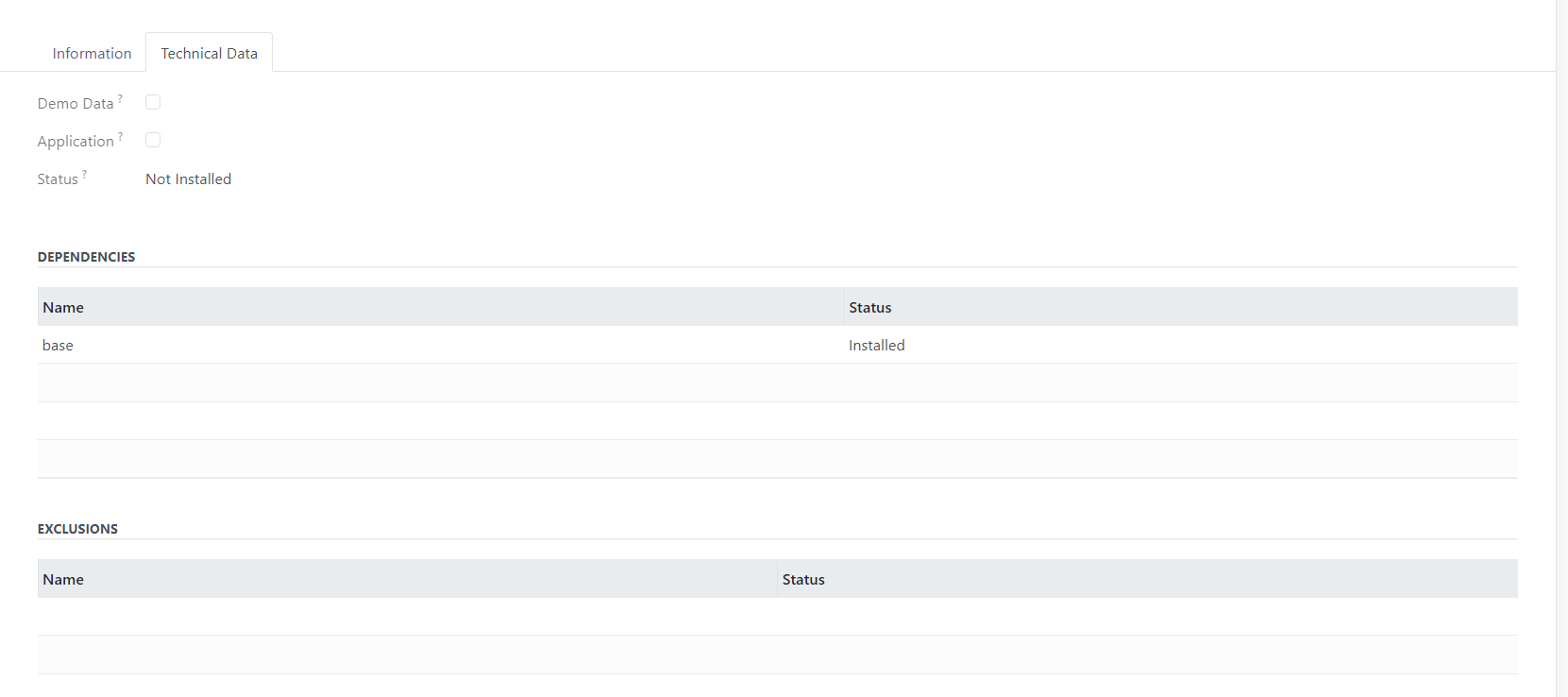


Bạn có thể thấy các thông tin như Website, Category, Summary, Description, Version, Technical Name (là cái folder), License



Ở đây License không có trong **\_\_manifest\_\_.py** nên Odoo sẽ mặc định là **LGPL-3**

Bên thẻ Technical Data



Bạn sẽ thấy cái “**depends**” nó hiện ra với tên **DEPENDENCIES**

Chi tiết về \_\_manifest\_\_.py bạn có thể xem ở đây: (link)

1. Bắt đầu viết applications

Bây giờ chúng ta sẽ viết application đầu tiên.

Vào \_\_manifest\_\_.py sửa lại thông số như sau(Hoặc tùy bạn):



Ở đây:

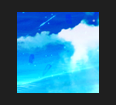
* **“version”** thì bạn nên để theo **<odoo\_version\_của\_bạn>**.**<module\_version\_của\_bạn>**
* **“application”** nếu đặt là **True** thì **odoo** sẽ xem **module** này là **application**
* **“category”** thì đặt theo nhu cầu thể loại của bạn

Vì là application, nên bạn hãy cho nó 1 cái icon bằng cách:

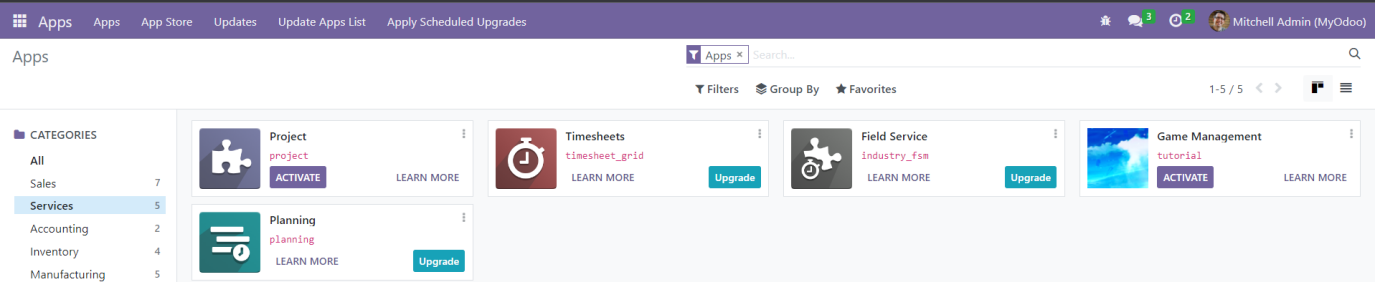
Trong folder gốc, tạo folder **static**, trong folder **static** tạo folder **description,** ta sẽ đặt file **icon.png** vào trong đây

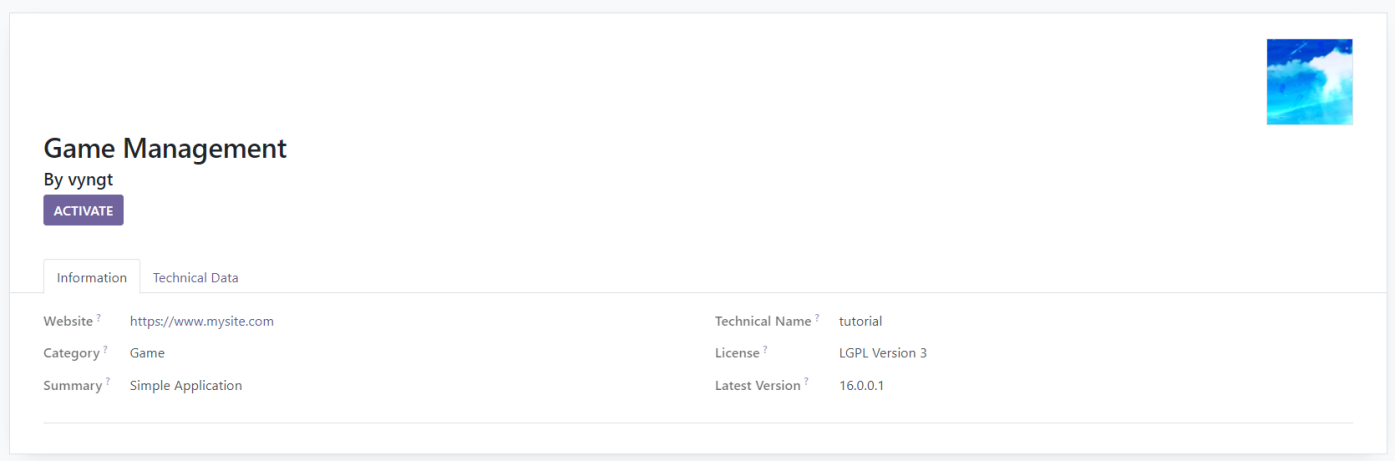


Ở đây icon của mình là cái này



Được rồi, khởi động lại Odoo và cập nhật lại danh sách Apps:

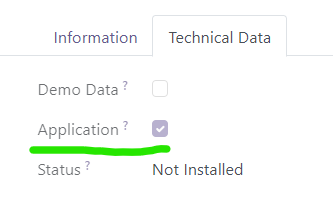




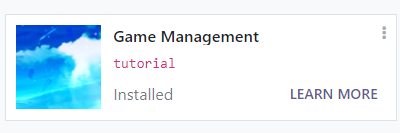
Bạn sẽ thấy Category đã thay đổi, hình đại diện cũng đổi.

Sang thẻ **Technical Data,** bạn sẽ thấy checkbox **Application** đã được tích vào.

Bạn có thể hiểu **Application** đơn giản là một **Module** có checkbox **Application**.



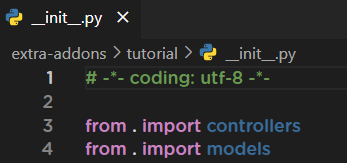
Được rồi, hãy **Active** nó và sang phần tiếp theo.



1. Models

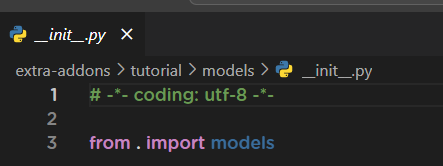
Trước tiên ở file **/tutorial/\_\_init\_\_.py** ta phải kiểm tra import của models

Như sau

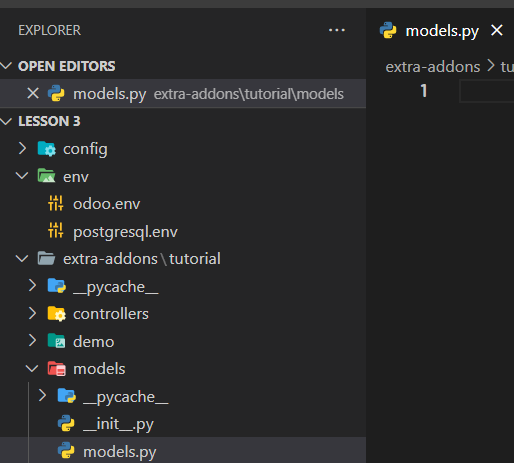


Ở dòng 4

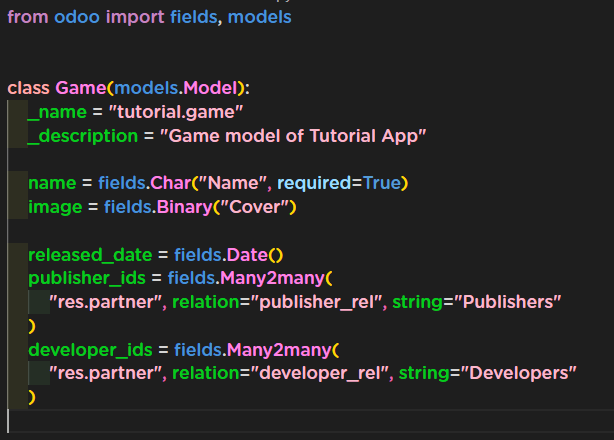
Sau đó, file **/tutorial/models/\_\_init\_\_.py** ta cũng làm tương tự



Rồi tại thư mục **models/** này tạo file **models.py** (Nếu chưa có)

****

Ta sẽ viết model mới như sau:



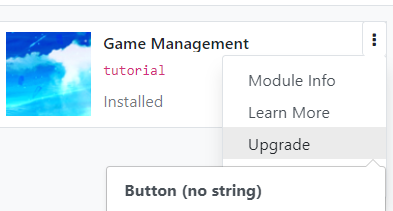
Ở đây, mình sẽ cho kế thừa **Model** của Odoo

Thuộc tính **\_name** ở đây là tên bảng trong Database (**Odoo** sẽ đổi dấu chấm “**.**” Thành dấu “**\_**”)

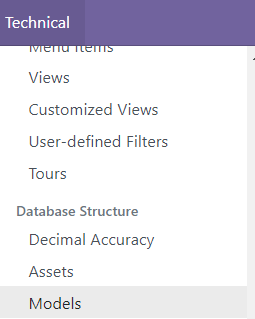
Thuộc tính **\_description** là mô tả model

Những thuộc tính còn lại sẽ do chúng ta tuỳ ý viết, ở đây thì mình dựa trên Steam, một Game sẽ có các thông tin như: *tên, ảnh bìa, ngày ra mắt, nhà phát hành và nhà phát triển*.

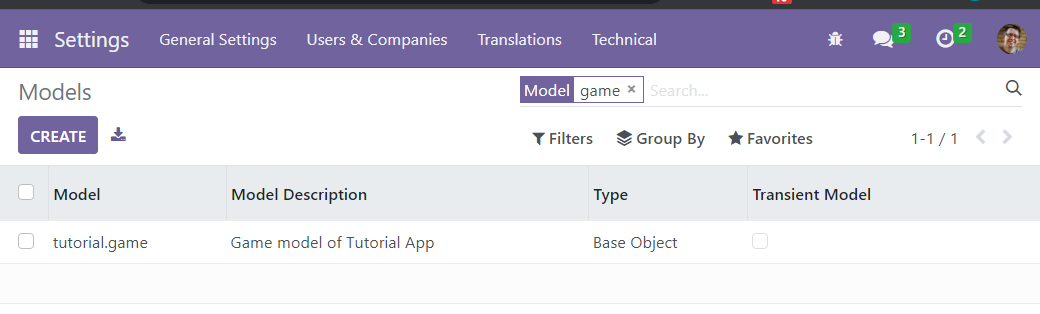
Lưu lại, khởi động lại Odoo và **Upgrade.**



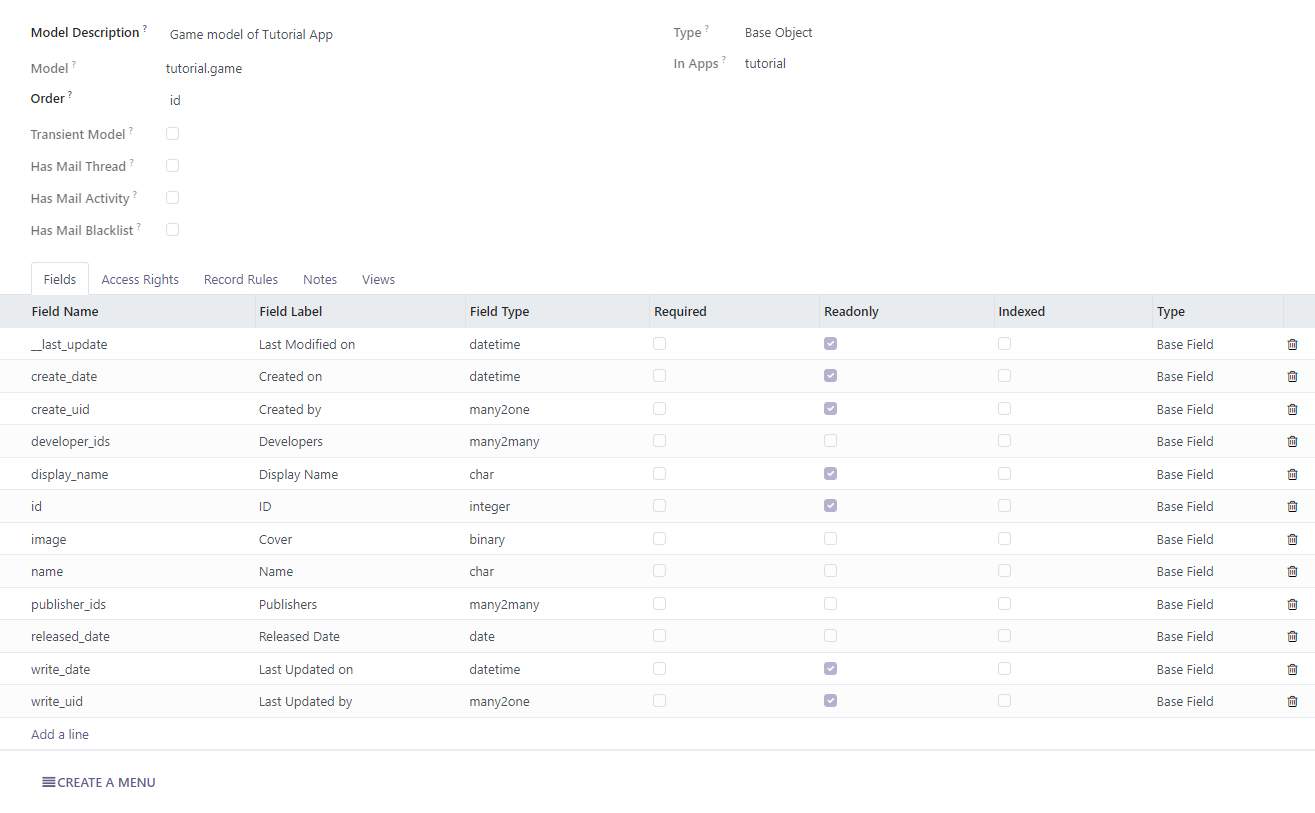
Sau đó vào **Settings | Technical | Models**



Bạn sẽ tìm thấy model của mình



Ấn vào sẽ thấy các fields

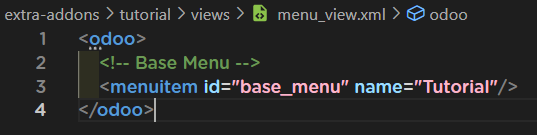


1. Tạo menu

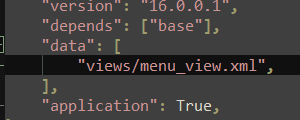
Tiếp theo chúng ta hãy tạo menu cho App

Tạo file **tutorial/views/menu\_view.xml**

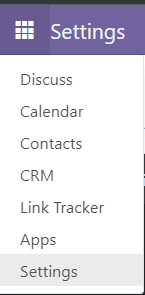
Với nội dung đơn giản như sau



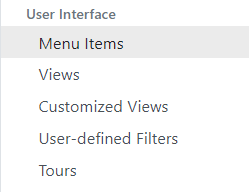
Vào **\_\_manifest\_\_.py** thêm vào **data**



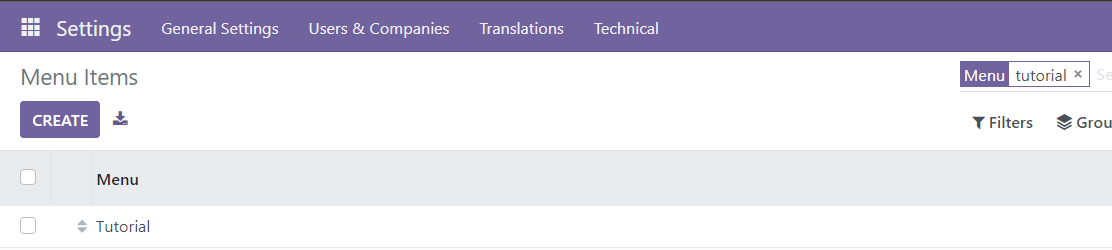
Sau khi làm xong bước này, chúng ta vẫn sẽ chưa thấy **Tutorial** trên thanh menu đâu (Vì chưa có submenu hoặc những thứ cần hiển thị trong này)



Chúng ta có thể kiểm tra nhanh(Trước đó hãy khởi động lại Odoo và Upgrade đã nhé) ở **Settings | Technical | Menu Items**



Và ta sẽ tìm được cái tên **Tutorial** ở đây



1. Security Groups

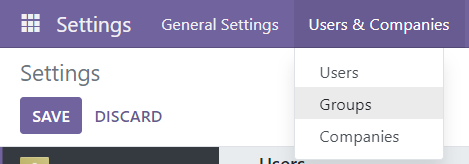
Trước khi có thể dùng các tính năng của **App,** user phải có quyền truy cập. Trong Odoo, ta có thể làm điều này với các **security group.** Các đặc quyền được cấp cho các **security group**. Các user được chỉ định vô các **security group.**

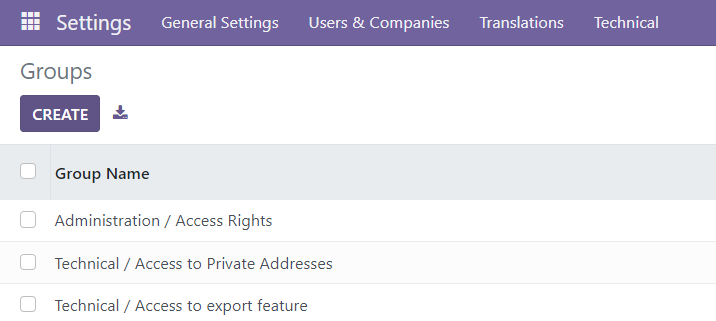
Các App của Odoo thường cung cấp 2 loại truy cập:

* + Cấp độ người dùng, cho phép người dùng thực hiện các tác vụ hằng ngày.
  + Cấp độ quản trị, đầy đủ quyền truy cập toàn bộ tính năng, bao gồm cả cấu hình.

Bây giờ chúng ta hãy tạo các **security group**, mà trước đó bạn hãy làm việc sau đây

Vào **Settings | Users & Companies | Groups**

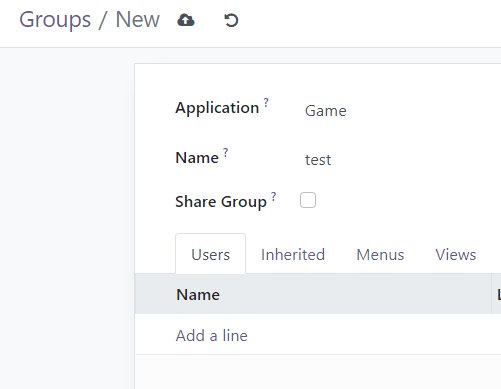




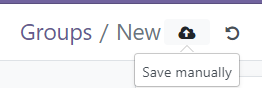
Hãy thử tạo một cái **group** như sau:

**Application**: Game

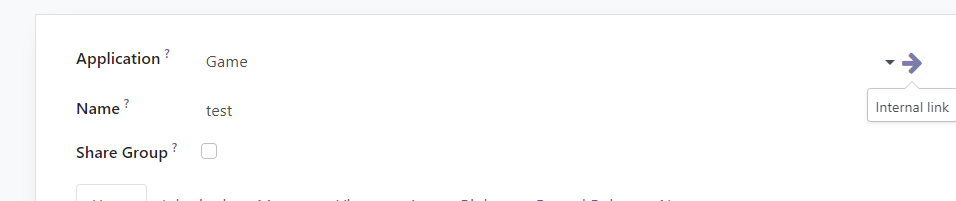
**Name**: test



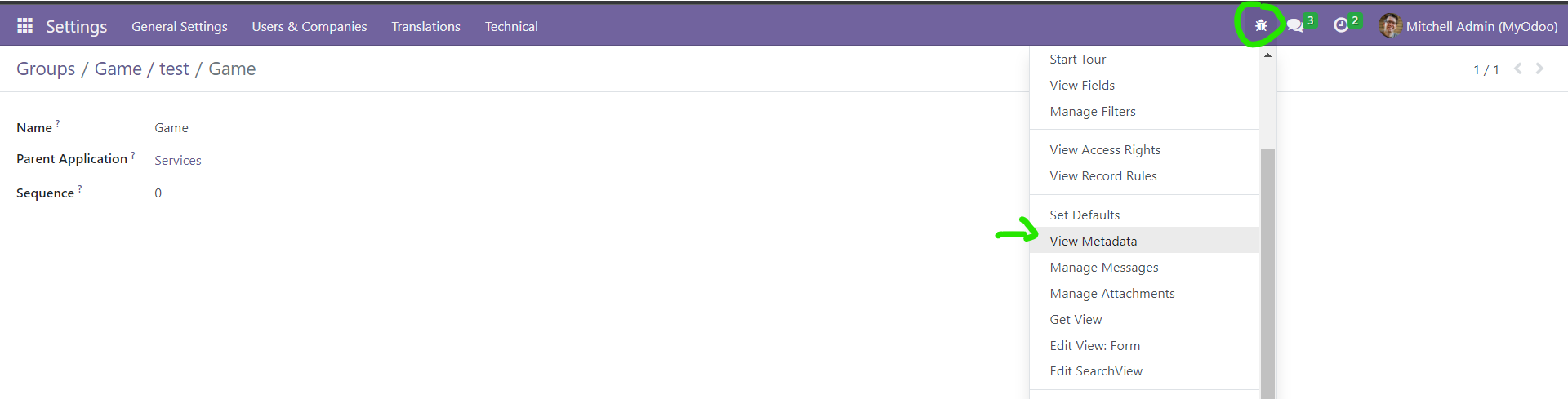
Sau khi lưu

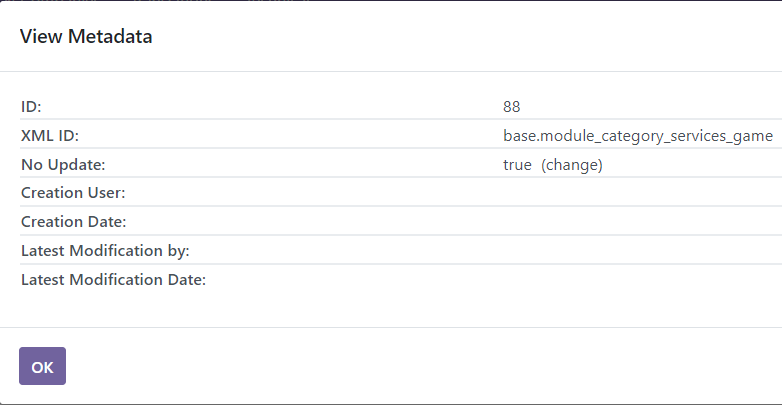


Bên cạnh trường Application, chọn dấu mũi tên Internal link



Ở nút hình **con bọ**, chọn **View Metadata**





Ở đây, ta sẽ thấy **XML ID**: **base.module\_category\_services\_game**

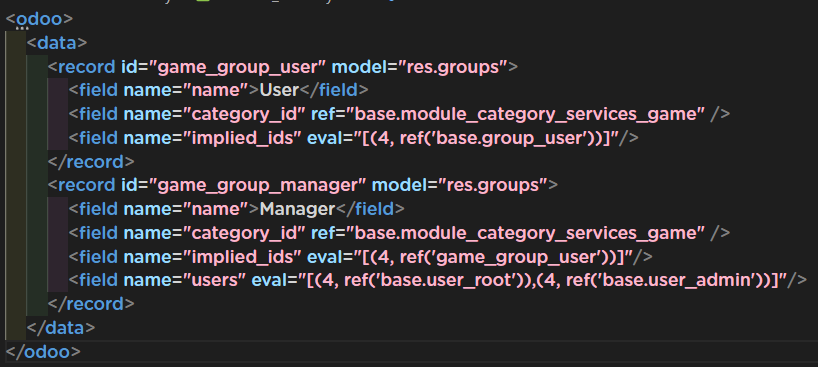
Xem lại **\_\_manifest\_\_.py** phần category ta thấy: “**Services/Game**”

Như vậy, ta đã biết với **module** có **category** là “**Services/Game**”

**XML ID** sẽ được sinh ra thành **base.module\_category\_services\_game**

Cái group test giờ bạn có thể xóa đi vì chúng ta sẽ không cần nữa.

Bây giờ, chúng ta hãy tạo file **/tutorial/security/tutorial\_security.xml**

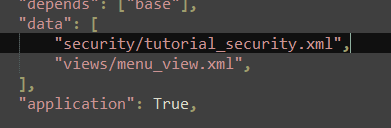
Với nội dung như sau:  


Ở đây, chúng ta tạo 2 group:

* + Trường “**name**” thì đơn giản là tên của group
  + Trường “**category\_id**” thì ta sẽ link tới XML ID
  + Còn “**implied\_ids**” là trường mối quan hệ **one-to-many**(1-n, một-nhiều) và bao gồm danh sách các nhóm mà sẽ áp dụng cho các user thuộc nhóm này.

Ở group **Manager,** implied\_ids link tới cái group **User** ở trênvà kế thừa các đặc quyền của group **User,** còn trường “**users**” trong này là đặt là “**base.user\_root**” là *superuser* và “**base.user\_admin**” là *Administrator*

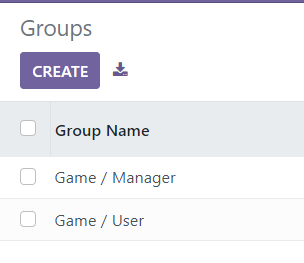
Lưu lại và sửa file **\_\_manifest\_\_.py**



Lưu ý:

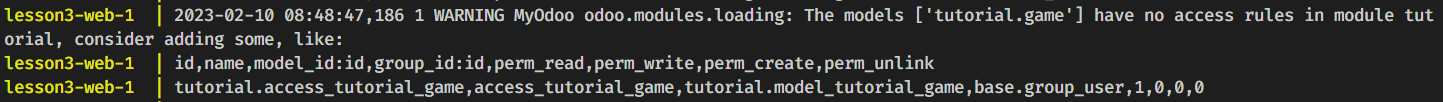
* + Theo thứ tự trong **data,** thường thì **security/** phải ở đầu danh sách.

Khởi động lại **Odoo** và **Upgrade,** Sau đó vào   
**Settings | Users & Companies | Groups** ta sẽ thấy 2 group mới.



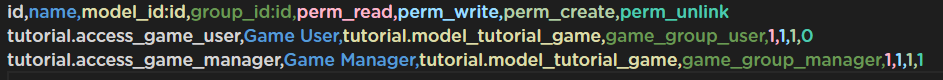
1. Access

Khi bạn viết xong Model đầu tiên, Odoo khởi động lên và Upgrade App, bạn sẽ để ý những dòng này từ Odoo

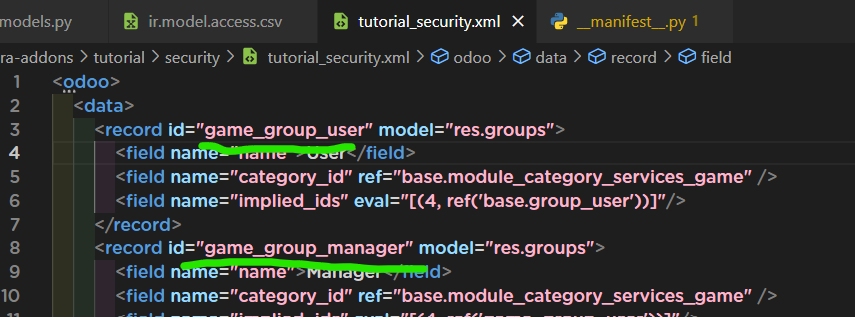


Odoo gợi ý rằng bạn chưa có **access rules** cho model này. Odoo thậm chí đưa ra giải pháp cho bạn.

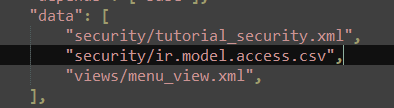
Hãy tạo file **/tutorial/security/ir.model.access.csv** với nội dung như sau



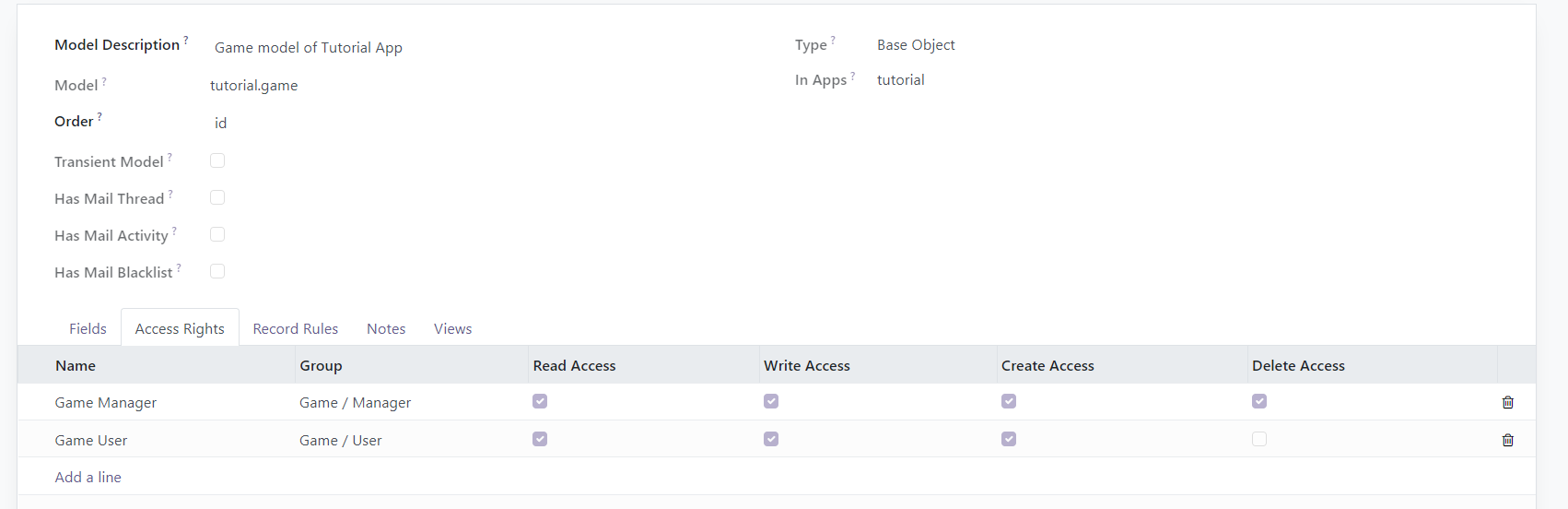
Ở đây:

* + **id**: ID của Access rule, bạn đặt tuỳ ý, miễn không trùng tên của các rule khác
  + **name**: Tên của Access rule
  + **model\_id:id**: Áp dụng vào **model** nào, ví dụ model có trường **\_name** = “a.b.c” thì ta sẽ đặt là <tên\_module>.model\_a\_b\_c
  + **group\_id:id**: là phần id của **Group**
  + **perm\_read**: Quyền đọc
  + **perm\_write**: Quyền ghi
  + **perm\_create**: Quyền tạo
  + **perm\_unlink**: Quyền xóa

Sửa nội dung trong **\_\_manifest\_\_.py** ở “**data**”



Khởi động lại Odoo và Upgrade, vào kiểm tra nhanh model   
(**Settings | Technical | Models** và tìm model của bạn).

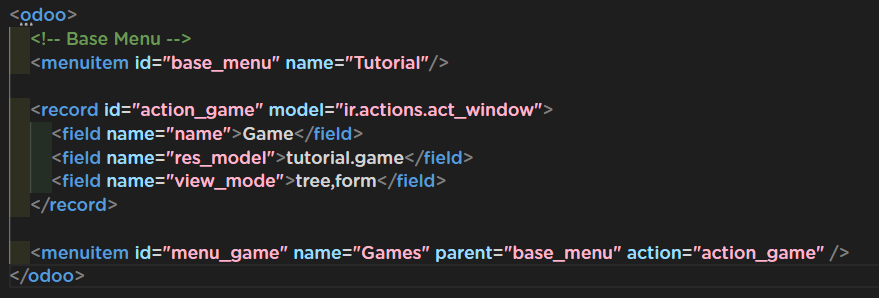


Nếu đã thành công rồi, hãy sang phần tiếp theo.

1. Backend view layer

Chúng ta đã hoàn thành **Model** và **Access Rights.** Bây giờ là lúc thao tác dữ liệu.

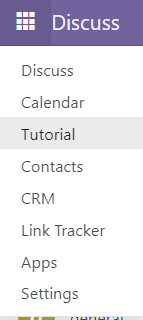
Sửa nội dung trong file “**tutorial/views/menu\_views.xml**”

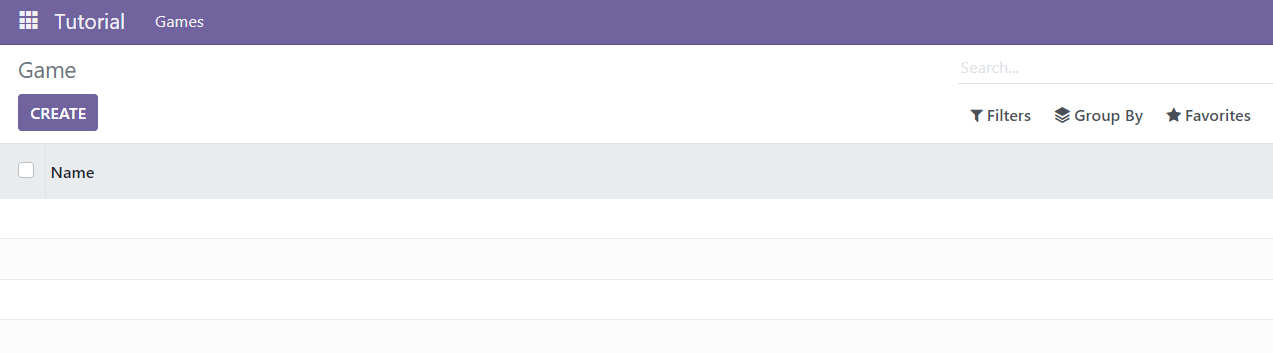


Ở đây, ta thêm action windows có **id**=”action\_game” nối với **model** là **tutorial.game** và có chế độ hiển thị là **tree** và **form** và một **menuitem** có **parent** là **base\_menu** và **action** là **action\_game**

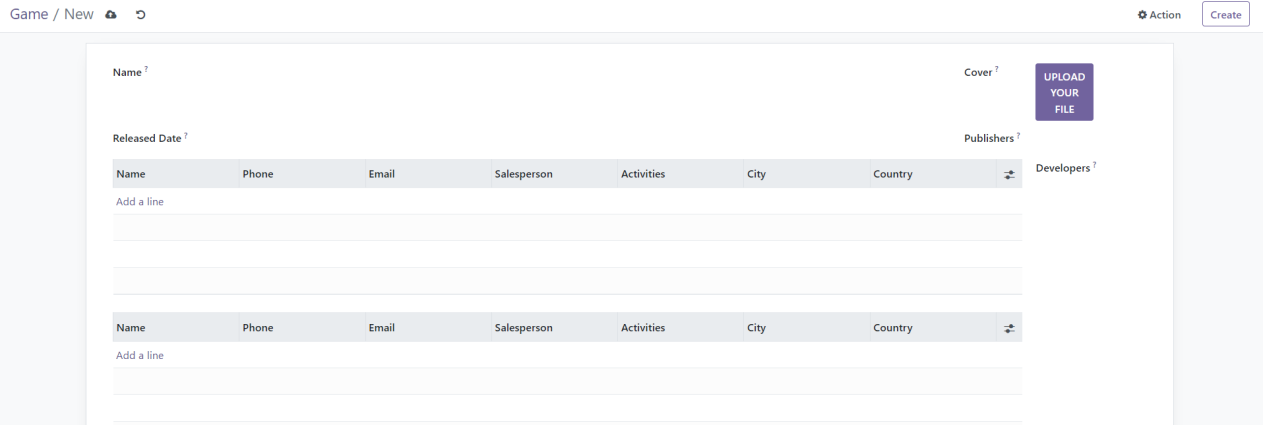
Lưu lại và chỉ cần **Upgrade** app là được

Đến lúc này ta đã có thể thấy được **Tutorial**





*TreeView (Danh sách các game)*



*Form tạo mới*

Chúng ta hãy custom lại những form này như sau

Tạo file **tutorial/views/game\_view.xml** với nội dung như sau

<odoo>

<!-- Form -->

<record id="view\_form\_game" model="ir.ui.view">

<field name="name">Game Form</field>

<field name="model">tutorial.game</field>

<field name="arch" type="xml">

<form string="Game">

<group name="group\_top">

<group name="group\_left">

<field name="name"/>

<field name="publisher\_ids" widget="many2many\_tags"/>

<field name="developer\_ids" widget="many2many\_tags"/>

</group>

<group name="group\_right">

<field name="released\_date"/>

<field name="image" widget="image"/>

</group>

</group>

</form>

</field>

</record>

<!-- Tree List -->

<record id="view\_tree\_game" model="ir.ui.view">

<field name="name">Game List</field>

<field name="model">tutorial.game</field>

<field name="arch" type="xml">

<tree>

<field name="name"/>

<field name="publisher\_ids" widget="many2many\_tags"/>

<field name="developer\_ids" widget="many2many\_tags"/>

<field name="released\_date"/>

</tree>

</field>

</record>

<!-- Search, Filter -->

<record id="view\_search\_game" model="ir.ui.view">

<field name="name">Game Filters</field>

<field name="model">tutorial.game</field>

<field name="arch" type="xml">

<search>

<field name="publisher\_ids" />

<field name="developer\_ids"/>

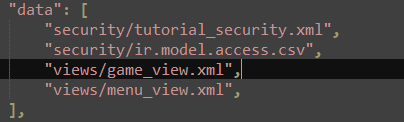
</search>

</field>

</record>

</odoo>

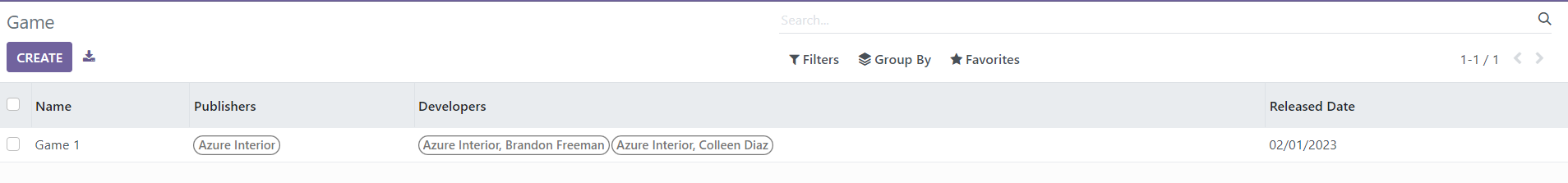
Lưu lại và thêm file này vào **data** trong **manifest**



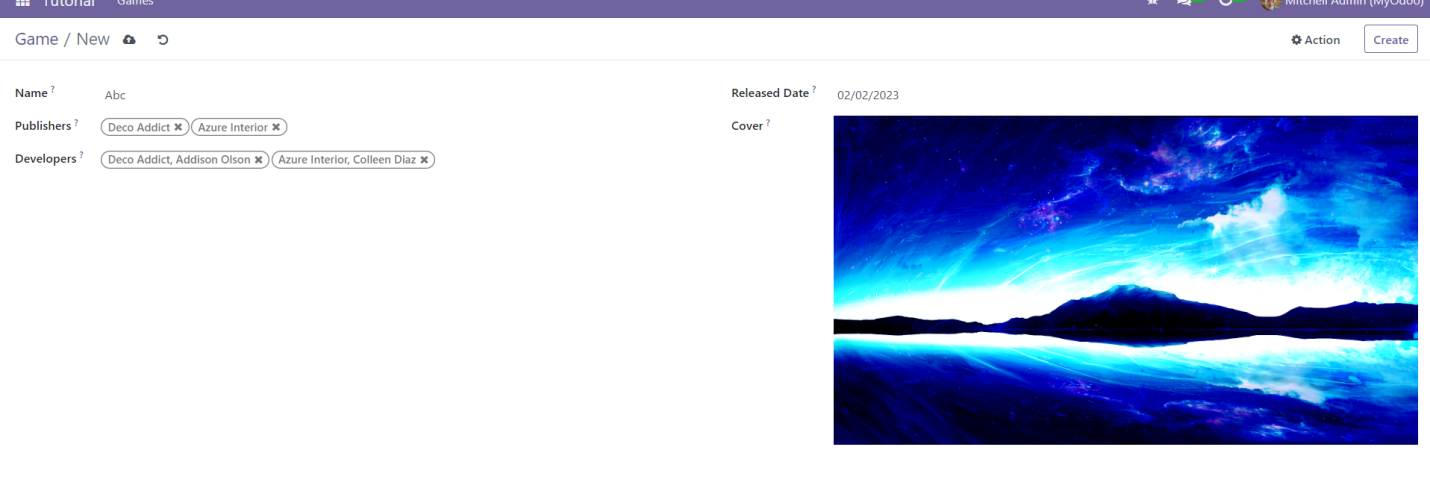
Thứ tự thì hãy đặt trên menu\_view

Khởi động lại Odoo và Upgrade app, chúng ta được

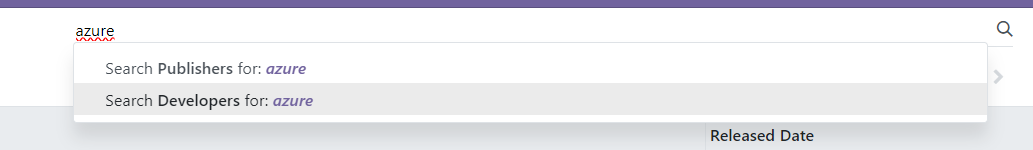
*Tree View*



*Form*



*Search Filter*



1. Bussiness Logic

**Bussiness logic layer** hỗ trợ các **bussiness rules** của **application**, như là **validations** hay **automation**.

Ở đây ta chỉ đơn giản làm bật/tắt ẩn hiện thôi để làm ví dụ cho Search Filter.

Vào chỉnh sửa file **tutorial/models/models.py**

from odoo import fields, models

class Game(models.Model):

\_name = "tutorial.game"

\_description = "Game model of Tutorial App"

name = fields.Char("Name", required=True)

image = fields.Binary("Cover")

**hidden = fields.Boolean("Hidden", default=False, readonly=True)**

released\_date = fields.Date()

publisher\_ids = fields.Many2many(

"res.partner", relation="publisher\_rel", string="Publishers"

)

developer\_ids = fields.Many2many(

"res.partner", relation="developer\_rel", string="Developers"

)

**def toggle\_hidden(self):**

**self.ensure\_one()**

**for game in self:**

**game.hidden = False if game.hidden else True**

**return True**

Ở đây ta thêm thuộc tính **hidden,** ta để là **readonly** (chỉ đọc), nghĩa là thao tác giao diện, điền form các thứ đều không được, nhưng có thể thông qua con đường khác để chỉnh sửa.

(*Cái này là ví dụ thôi nhé, có nhiều ví dụ hay hơn để biểu diễn thứ này,* như tính toán thứ gì đó, tạo số random, thêm thắt chuỗi như nào đó, format lại cái gì đó,…).

Tiếp theo, **ở /tutorial/views/game\_view.xml** cần thêm vài dòng ở đoạn form

<record id="view\_form\_game" model="ir.ui.view">

<field name="name">Game Form</field>

<field name="model">tutorial.game</field>

<field name="arch" type="xml">

<form string="Game">

**<header>**

**<button name="toggle\_hidden" type="object" string="Toggle Hidden" />**

**</header>**

**<sheet>**

<group name="group\_top">

<group name="group\_left">

<field name="name"/>

<field name="publisher\_ids" widget="many2many\_tags"/>

<field name="developer\_ids" widget="many2many\_tags"/>

</group>

<group name="group\_right">

<field name="released\_date"/>

**<field name="hidden"/>**

<field name="image" widget="image"/>

</group>

</group>

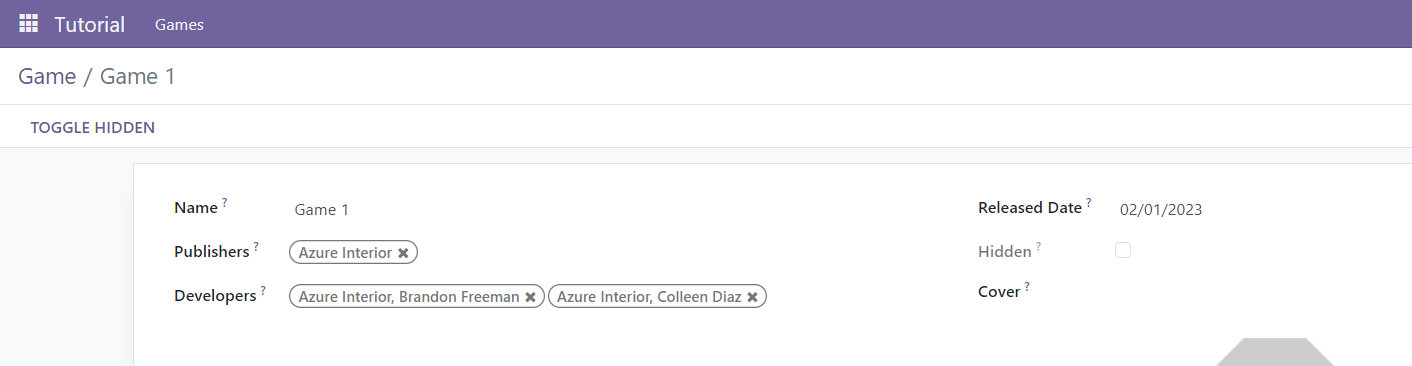
**</sheet>**

</form>

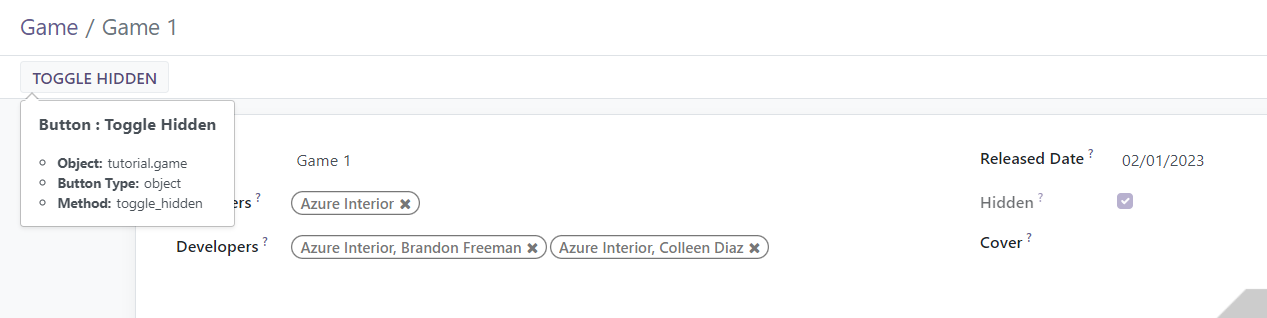
</field>

</record>

Sau đó khởi động lại **Odoo** và **Upgrade** module



Vào form, bạn sẽ thấy trường **Hidden** bị nhạt màu đi, không thể tích trực tiếp vào đó, và bạn để ý có cái nút “**Toggle Hidden**” ở bên góc trái bên phía trên dưới thanh **menu**



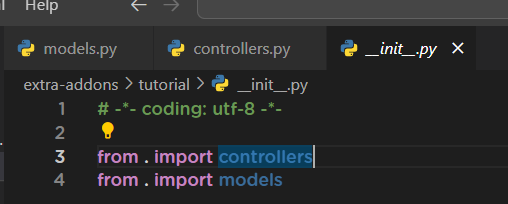
Phần này ta chỉ giới thiệu qua thế thôi, có rất nhiều điều cần làm với thứ này. Hãy qua phần tiếp theo nào

1. Web Controllers

Phần này khá gần gũi với những bạn nào về phát triển web.

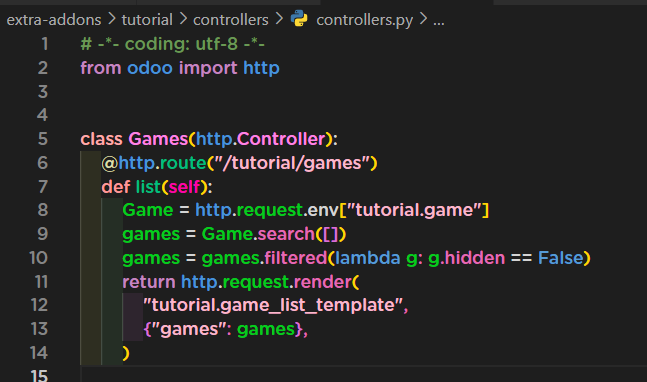
Bây giờ ta muốn khi vào web với URL: **http(s)://<Server\_name>/tutorial/games** thì Server sẽ trả về danh sách các game không bị ẩn. Và endpoint **/tutorial/games** là thứ mà ta cần triển khai.

Trước đó tại **tutorial/\_\_init\_\_.py** ta import **controllers** vào



Ở file **tutorial/controllers/\_\_init\_\_.py** ta import controllers

Rồi tạo file **/tutorial/controllers/controllers.py** với nội dung như hình dưới



Sau đó ta tạo một **template(**Cái này là QWeb) cho cái **controller** này, bằng cách tạo một file  
**/tutorial/views/game\_list\_template.xml** với nội dung như sau

<odoo>

<template id="**game\_list\_template**" name="Game List">

<div id="wrap" class="container">

<h1>Games</h1>

<t t-foreach="games" t-as="game">

<div class="row">

<span t-field="game.name" />,

<span t-field="game.released\_date" />

<span>Publishers:

<t t-foreach="game.publisher\_ids" t-as="pub">

<span t-field="pub.name" />,

</t>

</span>

</div>

</t>

</div>

</template>

</odoo>

Lưu lại thêm file này vào manifest

"data": [

"security/tutorial\_security.xml",

"security/ir.model.access.csv",

"views/game\_view.xml",

"views/menu\_view.xml",

**"views/game\_list\_template.xml",**

],

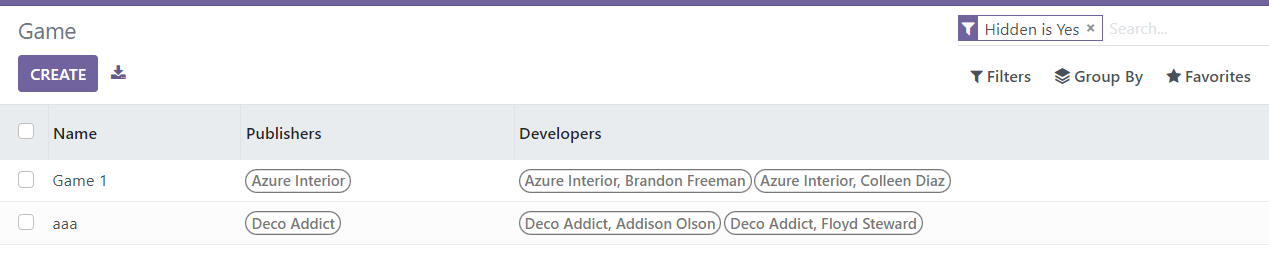
Khởi động lại **Odoo** và **Upgrade** app

Ta sẽ tạo vài record có **hidden** và không **hidden**

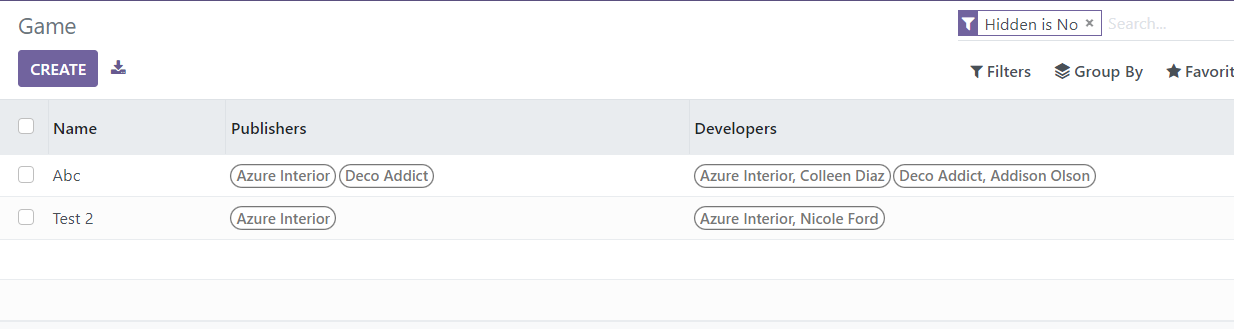
Dưới đây là toàn bộ record



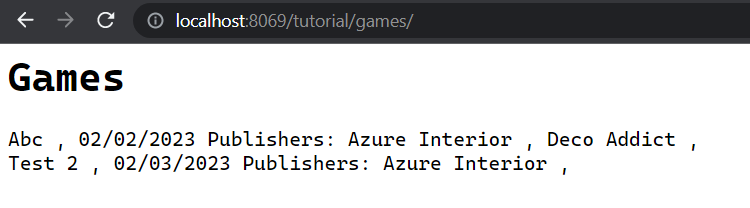
Record bị ẩn theo Search Filter



Record không bị ẩn theo Search Filter



Vào **URL** như đã nói ở ban đầu



Như bạn thấy đấy, hình trên chỉ hiện danh sách các **record** **không** bị **hidden**

1. Chú ý

Khi **module** chỉ có sự thay đổi ở các file **xml** thì chỉ cần Upgrade module là xong

Khi **module** có sự thay đổi ở các file **.py** thì hãy khởi động lại **Server**, tốt nhất là **Upgrade** luôn

Khi có sự nghi ngờ, cứ chạy lại **Odoo** và **Upgrade**

Bạn nên đọc logs của Server, sẽ giúp ích rất nhiều

1. Phần kết

Qua đây, tổng kết lại, chúng ta đã:

* Rõ hơn về **\_\_manifest\_\_.py**
* Cơ bản về **Model**
* Cách tạo **menu**, **submenu**
* **Form**, **List**, **Filter** view
* Một chút về **Security Group, Access Rules**
* Biết qua về tầng bussiness logic
* **Web Controller** và **QWeb Template**

1. Tham Khảo

* Link Github:

<https://github.com/vyngt/my-odoo-tutorial/tree/master/l3/tutorial>

* Odoo manifest: <https://www.odoo.com/documentation/16.0//developer/reference/backend/module.html>
* ORM: <https://www.odoo.com/documentation/16.0/developer/reference/backend/orm.html>
* Web Controllers: <https://www.odoo.com/documentation/16.0/developer/reference/backend/http.html>
* XML: <https://www.odoo.com/documentation/16.0/developer/reference/backend/data.html>
* QWeb Template: <https://www.odoo.com/documentation/16.0/developer/reference/frontend/qweb.html>