

# PASOS PARA LA RESOLUCIÓN

1. Se levanta el Docker.

```
PS C:\Users\Usuario\Desktop\Alicia\2. DAM\4. SGE\SGE_practicas> docker compose up -d  
[+] Running 2/2  
✓ Container db Started  
✓ Container odoo_16 Started
```

2. Establecimiento de credenciales.

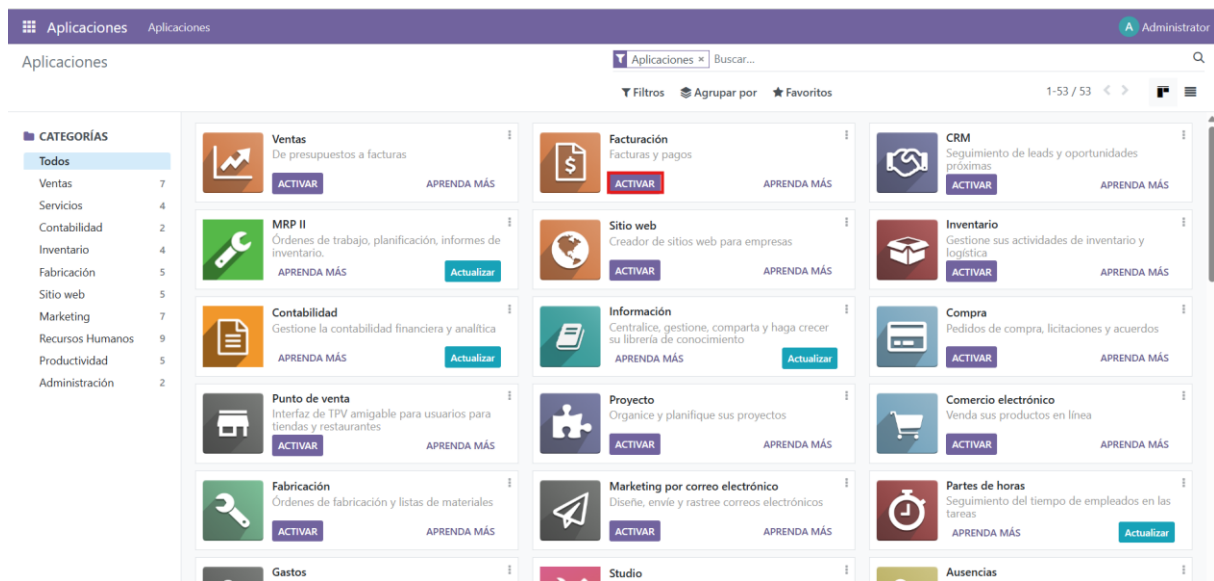
Warning, your Odoo database manager is not protected. To secure it, we have generated the following master password for it:

**qgcy-cdq2-gjtw**

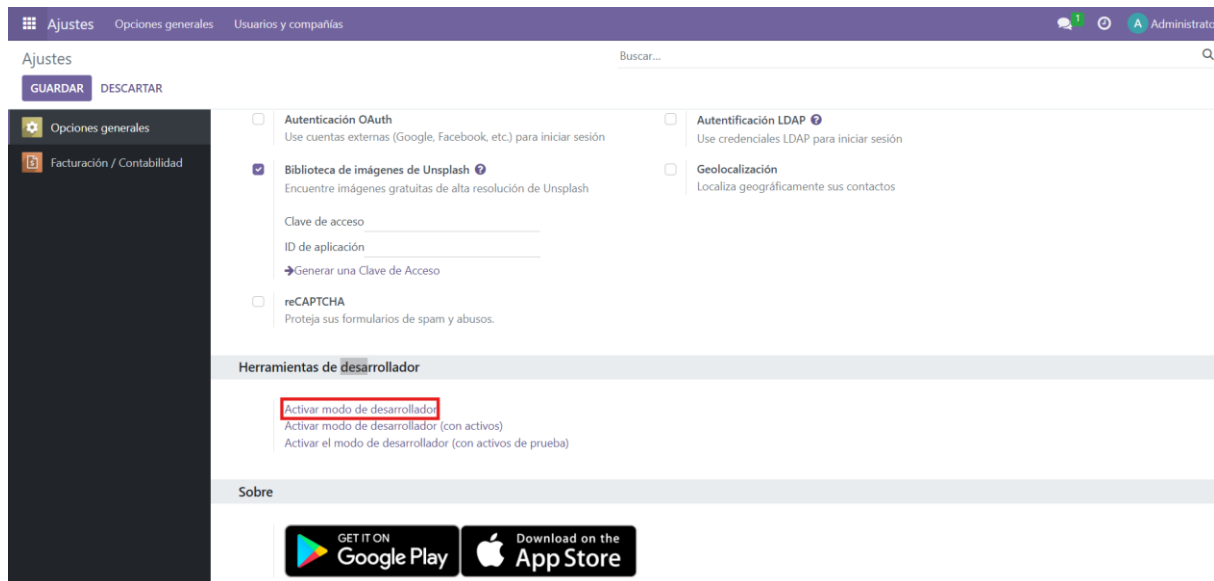
You can change it below but be sure to remember it, it will be asked for future operations on databases.

Master Password	<input type="password" value="...."/>
Database Name	<input type="text" value="db_gestion_biblioteca"/>
Email	<input type="text" value="alicia@alicia.com"/>
Password	<input type="password" value="...."/>
Phone number	<input type="text" value="666666666"/>
Language	<input type="text" value="Spanish / Español"/>
Country	<input type="text" value="Spain"/>
Demo data	<input type="checkbox"/>

### 3. Instalación de una aplicación para poder activar el modo de desarrollador.



### 3. Activación del modo de desarrollados en *Ajustes*.



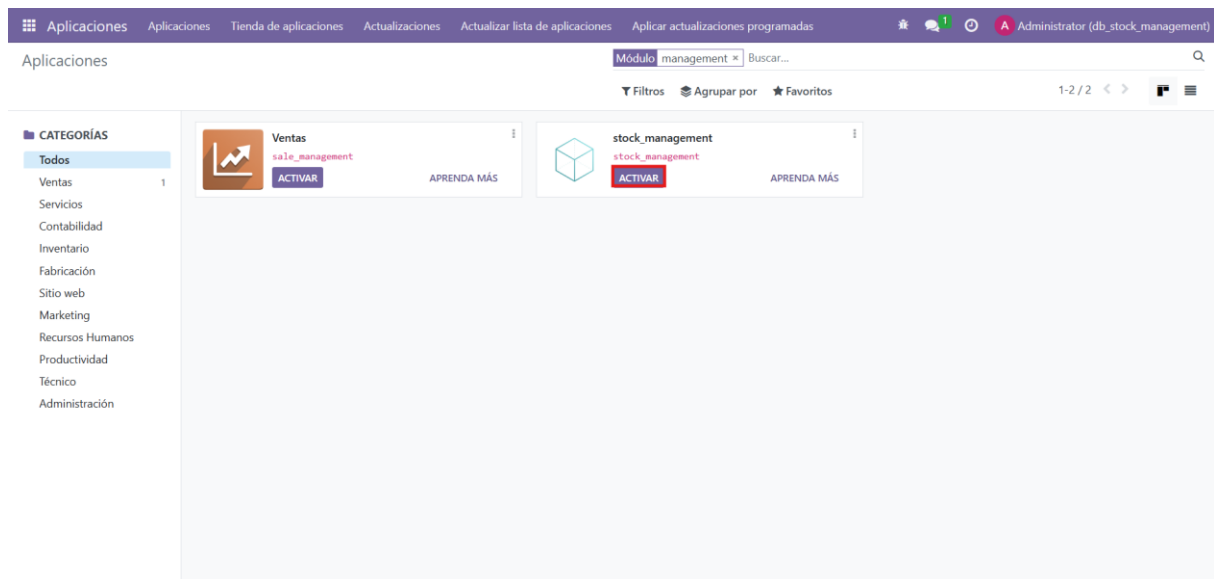
### 4. Acceso, mediante la línea de comandos, al contenedor Odoo.

```
PS C:\Users\Usuario\Desktop\Alicia\2. DAM\4. SGE\SGE_practicas> docker exec -it f385905d6a2a bash
odoo@f385905d6a2a:/$
```

### 5. Creación de la estructura básica del módulo con scaffold.

```
odoo@f385905d6a2a:/$ odoo scaffold stock_management /mnt/extra-addons/
odoo@f385905d6a2a:/$
```

## 6. Instalación del módulo.



## 7. Edición de los archivos necesarios para la correcta implementación del módulo.

### \_\_manifest.py\_\_

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

    'summary': 'Gestión de stock de productos',

    'description': 'Módulo para gestionar el stock de productos, incluyendo categorías, precios y cantidades.',

    'author': 'Alicia',
    'website': 'https://www.alicia.com',

    # Categories can be used to filter modules in modules listing
    # Check
https://github.com/odoo/odoo/blob/16.0/odoo/addons/base/data/ir\_module\_category\_data.xml
    # for the full list
    'category': 'Uncategorized',
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base'],

    # always loaded
    'data': [
```

```

        # 'security/ir.model.access.csv',
        'security/ir.model.access.csv',
        'views/stock_product_views.xml',
    ],
    # only loaded in demonstration mode
    'demo': [
        'demo/demo.xml',
    ],
}

```

## \_\_init.py\_\_

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

```

from . import stock_product
ir.model.access.csv

```

	id	name	model_id:id	group_id:id	perm_read	perm_write	perm_create	perm_unlink
1	access_stock_product	access_stock_product	model_stock_product_stock_product	base.group_user	1	1	1	1
2								

## stock\_product.py

```

from odoo import models, fields, api
from odoo.exceptions import ValidationError

class StockProduct(models.Model):
    _name = 'stock_product.stock_product'
    _description = 'Gestión de Stock'

    _sql_constraints = [
        ('unique_name', 'unique(name)', 'El nombre debe ser único'),
        ('check_quantity', 'CHECK(quantity >= 0)', 'La cantidad debe ser positiva'),
    ]

    name = fields.Char(
        string='Nombre producto',
        required=True)
    category = fields.Selection([
        ('television', 'Televisión'),
        ('pc', 'PC'),
        ('monitor', 'Monitor'),
        ('smartphone', 'Smartphone'),
    ],
        string='Categoria',
        required=True)

    price = fields.Float(string='Precio unitario')
    quantity = fields.Float(string='Cantidad de stock')

```

```

total_value = fields.Float(
    string='Valor total',
    compute='_compute_total', inverse='_inverse_total',
    store=True)
minimun_quantity = fields.Integer(
    string='Cantidad minima')

stock_status = fields.Selection([
    ('normal', 'Normal'),
    ('low stock', 'Low Stock'),
],
string="Estado",
compute='_compute_stock_status')

full_name = fields.Char(
    string='Nombre Completo', compute='_compute_full_name')

@api.depends('price', 'quantity')
def _compute_total(self):
    for record in self:
        record.total_value = record.price * record.quantity

def _inverse_total(self):
    for record in self:
        if record.quantity and record.quantity != 0:
            record.price = record.total_value / record.quantity

@api.depends('quantity', 'minimun_quantity')
def _compute_stock_status(self):
    for rec in self:
        if rec.quantity < rec.minimun_quantity:
            rec.stock_status = 'low stock'
        else:
            rec.stock_status = 'normal'

@api.depends('name', 'category')
def _compute_full_name(self):
    for rec in self:
        rec.full_name = f"{rec.name} ({rec.category})" if rec.name and
rec.category else rec.name

@api.constrains('price')
def _check_price(self):
    for rec in self:
        if rec.price < 0:
            raise ValidationError('El precio debe ser mayor que 0')

```

stock\_product\_views.xml

```

<odoo>
  <record id="view_stock_product_tree" model="ir.ui.view">
    <field name="name">stock.product.tree</field>
    <field name="model">stock_product.stock_product</field>
    <field name="arch" type="xml">
      <tree>
        <field name="full_name"/>
        <field name="category"/>
        <field name="price"/>
        <field name="quantity"/>
        <field name="total_value"/>
        <field name="stock_status"/>
      </tree>
    </field>
  </record>

  <record id="action_stock_product" model="ir.actions.act_window">
    <field name="name">Productos de Stock</field>
    <field name="res_model">stock_product.stock_product</field>
    <field name="view_mode">tree,form</field>
  </record>

  <menuitem id="menu_stock_root" name="Stock Management"/>
  <menuitem id="menu_stock_product" name="Productos"
parent="menu_stock_root" action="action_stock_product"/>
</odoo>

```

## 8. Agregación de registros para comprobar el correcto funcionamiento

Stock Management		Productos	Administrator (db_stock_management)	
Productos de Stock / TV Samsung		Acción 1 / 1		Nuevo
Nombre producto ?	TV Samsung	Categoría ?	Televisión	
Precio unitario ?	500,00	Cantidad de stock ?	2,00	
Valor total ?	1.000,00	Cantidad mínima ?	2	
Estado ?	Normal	Nombre Completo ?	TV Samsung (television)	

Stock Management

Productos

1

Administrator (db\_stock\_management)

Productos de Stock

Buscar...

NUEVO

Filtros

Agrupar por

Favorites

1-1 / 1

<input type="checkbox"/>	Nombre Completo	Categoría	Precio unitario	Cantidad de stock	Valor total	Estado
<input type="checkbox"/>	TV Samsung (television)	Televisión	500,00	2,00	1.000,00	Normal

Stock Management

Productos

1

Administrator (db\_stock\_management)

Productos de Stock / Nuevo

Guardar manualmente

Nombre producto ?

Precio unitario ?

Valor total ?

Estado ?

Categoría ?

Cantidad de stock ?

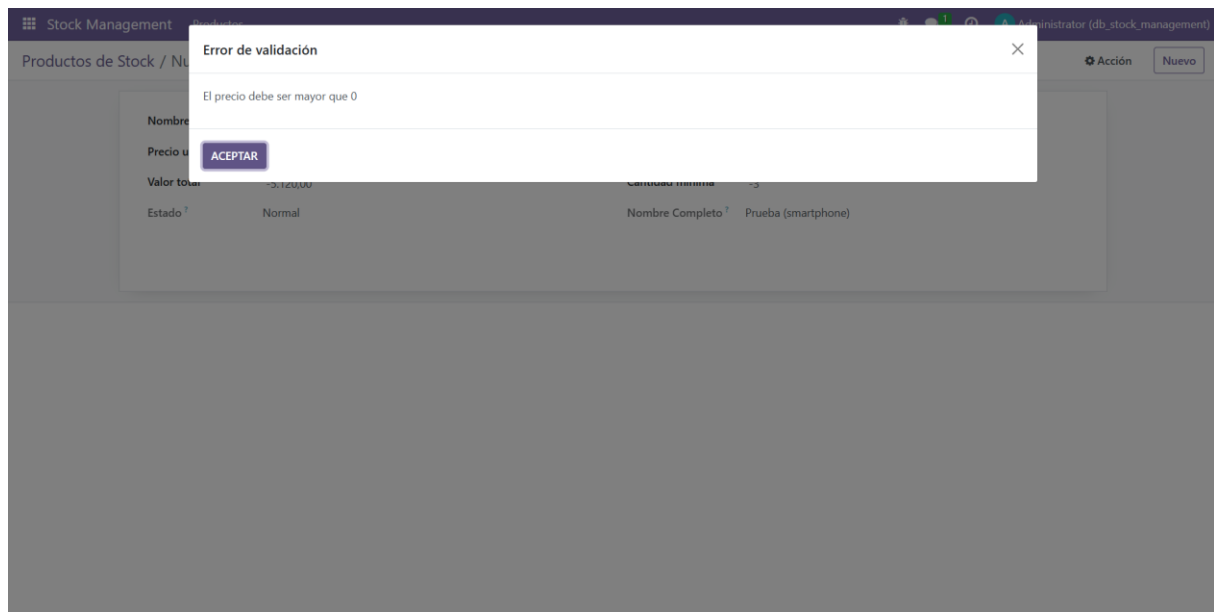
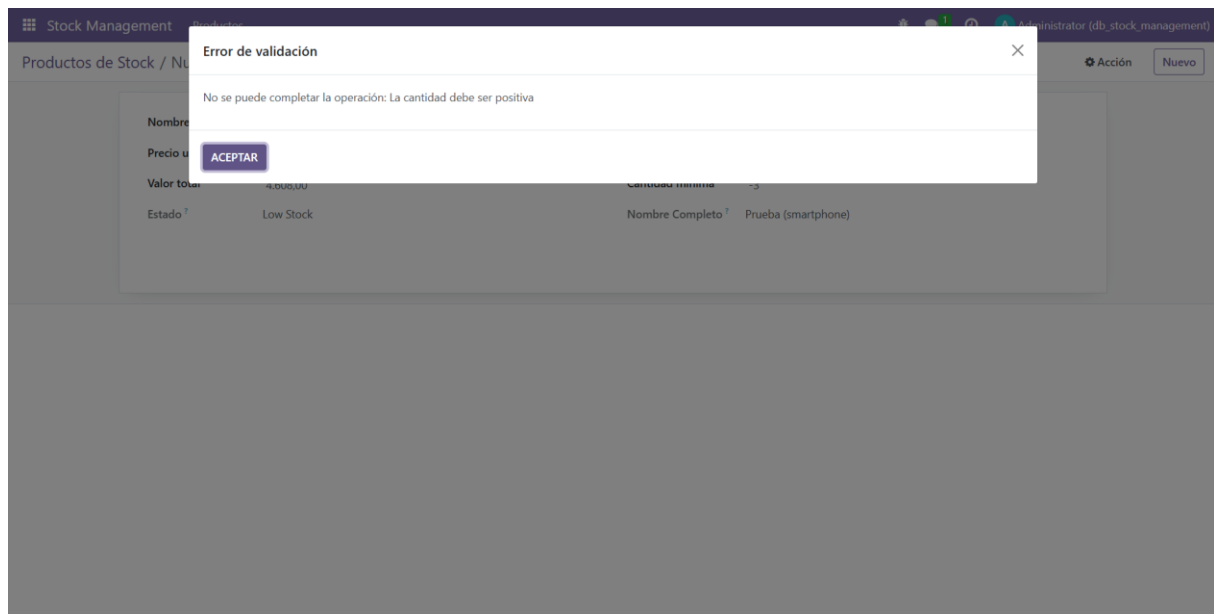
Cantidad mínima ?

Nombre Completo ?

Campos no válidos:

Nombre producto

Categoría





Buscar...

Q

 Filtros
  Agrupar por
  Favoritos

1-2/2 < >

<input type="checkbox"/> Nombre Completo	Categoría	Precio unitario	Cantidad de stock	Valor total	Estado
<input type="checkbox"/> TV Samsung (television)	Televisión	500,00	2,00	1.000,00	Normal
<input type="checkbox"/> Prueba (smartphone)	Smartphone	700,00	10,00	7.000,00	Normal