V20221130

Aprende a programa en Odoo ERP 16 con ejercicios prácticos

Marlon Falcón Hernández

Dedicación.

A ...

Índice

Capítulo 1 - Sobre el autor.

Marlon Falcón Hernandez con más de 20 años de experiencia en programación. Ha trabajado por 6 años en más de 200 proyectos y ha participado en el desarrollo de más de 3000 aplicaciones de **Odoo ERP**.

Se especializa en el desarrollo de verticales como Odoo Sis, Odoo Bim o CrediPro

Valencia, España.
Página web: www.marlonfalcon.com
Email: mfalconsoft@gmail.com

Capítulo 2 - ¿Qué veremos en este libro?

En este libro veremos conceptos básicos de programación enfocados en preparar un programador de Odoo. Haremos un conjunto de ejercicios de los problemas más comunes que día a día encontramos los programadores cuando trabajamos en un proyecto.

Capítulo 3 - Entendiendo a Odoo ERP.

Odoo ERP es un sistema de gestión empresarial de código abierto, desarrollado por la empresa Odoo. Utiliza distintos lenguajes de programación como son Python, Javascript, CSS, HTML, SQL, etc.

Capítulo 4 - Git simplificado.

Git es un software de control de versiones diseñado por Linux Torvalds, es un sistema que se destaca por eficiencia, confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Cuando programas para Odoo, casi siempre tendrás que alojar el código fuente de tu aplicación en un repositorio de git, los gestores de repositorio más conocido son:

www.github.com
www.bitbucket.com
www.gitlab.com

En este libro veremos el caso de Github, para lo cual vamos a ir a la web de github.com y utilizando un correo y contraseña nos vamos a crear una cuenta.

Un **repositorio** es un espacio de trabajo donde almacenaremos los módulos en github.
Una **rama** es una versión de ese repositorio. Ejemplo rama 14.0 o rama 16.0.

Comandos utilizados en git

Permite clonar un repositorio de github.

```
$ git clone git@github.com:falconsoft3d/docker-odoo-16.git
```

```
$ git pull
```

Permite bajar los cambios del repo para saber que otro programador ha realizado cambios en alguna parte del repo.

```
$ git add .
$ git commit -m "Subir cambios"
$ git push
```

Estos 3 comandos seguidos así como está te permite subir los cambios locales al repositorio de github.

```
$ git status
```

Permite ver los cambios locales.

```
$ git branch
```

Te muestra en qué rama del proyecto estás.

```
$ git checkout 14.0
```

Nos cambiamos de rama, en este ejemplo a la rama 14.

Capítulo 5 - Docker simplificado

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Podemos utilizar docker para entornos de desarrollo y de programación con Odoo.

Docker Compose es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores. En Compose, se usa un archivo YAML para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios de la configuración.

Comandos

Instalar docker en linux.
```
$ apt-get install docker
```

Listar docker que estén corriendo.
```
$ docker ps
```

Parar un docker.
```
$ docker stop 777799021212
```

Eliminar un docker.
```
$ docker rm 777799021212
```

Correr Odoo con docker compose con estos 3 comandos, donde clonamos el repo de ejemplo y con “docker compose” o con “docker-compose” lo levantaremos.
```
$ git clone https://github.com/falconsoft3d/docker-odoo-16.git
$ cd docker-odoo-16
$ docker compose up -d
```

Paramos el docker.
```
$ docker stop 777799021212
```

Entramos en un contenedor existente.
```
$ docker exec -it CONTAINER_ID bash
```

Historial de la imagen.
```
$ docker image history f70b4e424946
```

Ejemplos de Dockers

Ejecutamos un contenedor de MySQL

```
$ docker run -e MYSQL_ROOT_PASSWORD=x1234567890 -d mysql:8.0.13
```

Ejecutamos un contenedor de PgAdmin 4

```
$ docker pull dpage/pgadmin4
docker run -p 5050:80 --link db:db --name pgadmin4dev -e
"PGADMIN_DEFAULT_EMAIL=mfalcon@ynext.cl"   -e
"PGADMIN_DEFAULT_PASSWORD=secret" -d dpage/pgadmin4
```

Subimos nuestro docker a un repositorio pero de Dockers llamado Docker Hub.

```
$ docker login
$ docker tag mfalconsoft/odoo:10.0 mfalconsoft/odoo:10.0
$ docker push mfalconsoft/odoo:10.0
$ docker run -d  mfalconsoft/odoo:10.0
```

Ejecutamos un contenedor FTP

```
$ docker run -d -p 21:21 -p 21000-21010:21000-21010 -e USERS="one|1234" -e
ADDRESS=127.0.0.1 delfer/alpine-ftp-server
```

Capítulo 6 - GNU/Linux simplificado.

GNU/Linux, es un sistema operativo tipo Unix compuesto por software libre y de código abierto. **GNU/Linux** surge de las contribuciones de varios proyectos de software, entre los cuales destacan **GNU** y el kernel **«Linux»**

Comandos que más utilizaremos con Odoo

Conectarse a un servidor con el comando ssh.

```
$ ssh-copy-id root@159.89.29.221
```

Convertirse en superusuario.

```
$ sudo su
```

Actualizar el sistema operativo.

```
$ apt-get update && apt-get upgrade -y
```

Crear una carpeta y dar permiso.

```
$ mkdir /opt/extra-addons
$ chown odoo: /opt/extra-addons/ -R
```

Editamos un fichero con Nano.

```
$ nano /etc/odoo/odoo.conf
```

Reiniciamos un servicio.

```
$ service odoo restart
```

Paramos un servicio.

```
$ service odoo stop
```

Iniciamos un servicio.

```
$ service odoo start
```

Instalamos un programa

```
$ apt-get install htop
```

Vemos los recursos del sistema

```
$ htop
```

Generamos una clave pública.

```
$ ssh-keygen
$ cat ~/.ssh/id_rsa.pub
```

Actualizamos Odoo

```
$ /etc/init.d/odoo stop
$ su - odoo -s /bin/bash
$ odoo -d db11-spain -u all --stop-after-init --logfile=/dev/stdout
$ odoo -d db16-spain -u sis_base --i18n-overwrite --stop-after-init
--logfile=/dev/stdout
$ /etc/init.d/odoo start
```

Capítulo 7 - HTML y Bootstrap simplificado

HTML, siglas en inglés de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.

Ejemplo del código fuente de una página web.
Etiquetas más usadas <h1>, <p>, <img>, <link>, <html>, <body>
la palabra reservada **class** se utiliza para asignar una clase que permite ponerles estilos.
La palabra reservada **id** se utiliza como identificador.

```
$ nano index.html
```

```
<!doctype html>
<html lang="es">
<head>
    <meta charset ="UTF8">
    <title>My Blog</title>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel="stylesheet" type="text/css" href="style.css">
    <script
src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
</head>
<body>
   <h1 class="primera">Esto es una texto grande</h1>
   <p id="myid">Esto es un párrafo</p>
</body>
<html>
```

Ejemplo del código fuente de una página web usando Bootstrap
Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios web y aplicaciones web.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset ="UTF8">
  <title>Ejemplo Bootstrap</title>
  <meta name="viewport" content="width = device-width">

 <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstra
p.min.css" rel="stylesheet"
integrity="sha384-rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEd
K2Kadq2F9CUG65" crossorigin="anonymous">
```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.
bundle.min.js"
integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rY
XK/7HAuoJl+0I4" crossorigin="anonymous"></script>


</head>
 <body>
  <div class="container">
       <h1>Hello</h1>
  </div>
   </body>
</html>
```

Container

Permite crear un contenedor responsivo para su página web

```
 <div class="container">
       <h1>Hello</h1>
  </div>
```

Row y Col

Permite crear columnas separadas por filas.

```
<div class="container text-center">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>
```

Otra forma utilizando espacios fijos

```
<div class="container text-center">
  <div class="row">
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>

    <!-- Force next columns-->
    <div class="w-100 d-none d-md-block"></div>

    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
    <div class="col-6 col-sm-4">.col-6 .col-sm-4</div>
  </div>
```

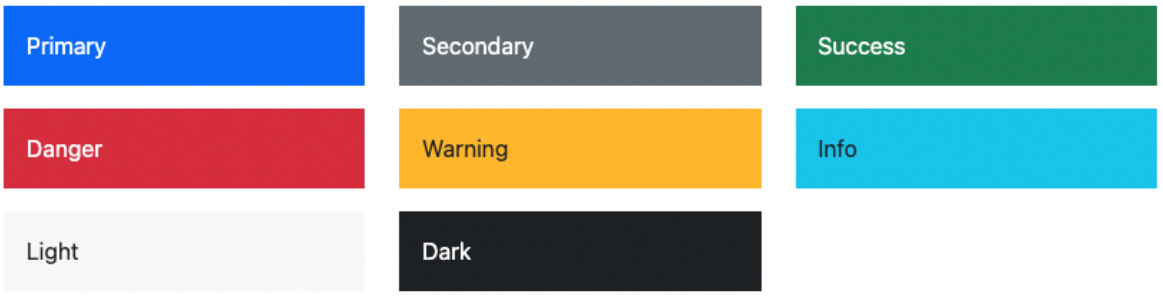
```
</div>
```

Crear una tabla con html y bootstrap. Se utilizan las etiquetas table, thead, tr, tbody.

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
  </tbody>
</table>
```

Colores más usados.

```
$theme-colors: (
  "primary":    $primary,
  "secondary":  $secondary,
  "success":    $success,
  "info":       $info,
  "warning":    $warning,
  "danger":     $danger,
  "light":      $light,
  "dark":       $dark
);
```


Primary
Secondary
Success
Danger
Warning
Info
Light
Dark

Capítulo 8 - CSS simplificado

CSS, en español **Hojas de estilo en cascada**, es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

Se crea un archivo con extensión css ejemplo, main.css. y se vincula con la página web.

estilos más usados.

```
html {
        background-color: #f9f9f9;
      }
```

Dándole color a una etiqueta que tenga una clase.

```
.clase-demo {
        color: #f9f9f9;
      }
```

Dándole color a una etiqueta que tenga un id llamado id-demo.

```
#id-demo {
        color: #f9f9f9;
      }
```

Ancho mínimo 225px , centrando los elementos.

```
#end.ytd {
         min-width: 225px;
         align-items: center;
         justify-content: flex-end;
}
```

Configuramos el margen y el padding de un elemento con id: id-demo.

```
#id-demo {
        margin: 0;
        padding: 0;
      }
```

Asignamos el tamaño de texto a los elementos tipo p

```
p {
      font-size: 13px;
  }
```

Capítulo 9 - SQL simplificado

SQL, es un lenguaje de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

Sentencias de manipulación de datos DML (Lenguaje de Modificación de Datos)

SELECT Listar los usuarios del sistema.

```
$ SELECT * FROM res_users;
$ SELECT name, email FROM res_partner LIMIT 10
$ SELECT id,login FROM res_users WHERE id=4
$ SELECT name,move_type FROM public.account_move
ORDER BY id ASC LIMIT 100
```

INSERT insertamos un contacto.

```
 $ INSERT INTO res_partner (name, email)
VALUES ('Juan', 'Tom B. juan@marlonfalcon.com');
```

UPDATE modificamos el password del admin.

```
$ UPDATE res_users SET password='123' WHERE login='admin';
$ UPDATE account_move SET name = concat('F',old_id) WHERE old_id <>
0
UPDATE empleados SET edad = 23 WHERE sueldo > 12;
```

DELETE borramos las facturas canceladas.

```
$ DELETE from account_move WHERE old_id IN (22001733,#,#)
$ DELETE FROM account_invoice WHERE state = 'cancel';
```

Capítulo 11 - Python 3 simplificado.

Python, Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo.

Qué debo saber antes de empezar con Python.

1- Python utiliza la identificación para delimitar la estructura no utiliza { }.
2- Para instalar una librería externa se utiliza **pip install nombre-libreria.**
3- Cuando utilizamos librerías externas es mejor usar.

```
import logging
_logger = logging.getLogger(__name__)

try:
    import dropbox
    from tqdm import tqdm
except:
    pass
    logging.info("Missing dropbox or tqdm")
```

Conectándose a una API Rest.

```
var request = require('request');
var options = {
 'method': 'GET',
 'url': 'https://randomuser.me/api/',
 'headers': {
 }
};
request(options, function (error, response) {
 if (error) throw new Error(error);
 console.log(response.body);
});
```

Tipos de datos en Python.
En **Python** no hace falta declarar el tipo de datos, él reconocerá el tipo según la información que tenga.

Entero
```
a = -1
```

Flotante
```
real = 1.1 + 2.2
```

Tipo string
```
hola = 'Hola "Pythonista"'
```

Booleano
```
mydata = True
```

Listas
```
lista = [1, 2, 3, 8, 9]
```

Diccionario
```
diccionario = {'a': 1, 'b': 3, 'z': 8}
```

Sentencia IF ELSE
Permite tomar un camino u otro.
```
if condition:
    if_body()
elif other_condition:
    elif_body()
elif another_condition:
    another_elif_body()
else:
    else_body()
```

Sentencia For
Permite recorrer listas.
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
fruits = ["a", "b", "c"]
for x in fruits:
  if x == "a":
    break
  print(x)
```

Control de excepciones
Con python podemos capturar errores que ocurran para tratarlos con Try y Except.

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

Funciones en Python
Podemos reutilizar código con las funciones.

```
def multiplica(val1, val2):
  return val1 * val2

multiplica(2, 2)  # muestra 4 en la consola
```

Decoradores
Los decoradores son funciones que modifican el comportamiento de otras funciones, ayudan a acortar nuestro código.

Ejemplos: `@api.constrains('account_type'), @api.depends('code'), @api.model`

```
@api.model
def _search_new_account_code(self, company, digits, prefix):
     for num in range(1, 10000):
          new_code = str(prefix.ljust(digits - 1, '0')) + str(num)
          rec = self.search([('code', '=', new_code), ('company_id',
'=', company.id)], limit=1)
            if not rec:
                return new_code
        raise UserError(_('Error'))
```

POO en Python
Python permite el uso de clases para simplificar la programación.

```
class Alumno:
    def __init__(self):
        self.nombre = "Pablo"

    def saludar(self):
        """Imprime un saludo en pantalla."""
        print(f"¡Hola, {self.nombre}!")
```

```
alumno = Alumno()
alumno.nombre = "Pablo"
```

POO en Python

Una clase en Python puede heredar de otras clases.

```
class ClaseA(ClaseB, ClaseC, ClaseD):
    pass
```

Capítulo 12 - Tipos de campos en los modelos de Odoo

Char: se utilizan para guardar datos de tipo texto.

```
name = fields.Char(string="Account Name", required=True,
index='trigram', tracking=True)
```

Boolean: Campo booleano o binario.

```
deprecated = fields.Boolean(default=False, tracking=True)
```

Text: Campo de tipo texto.

```
note = fields.Text('Internal Notes', tracking=True)
```

Float: Campo de tipo flotante.

```
current_balance = fields.Float(string='Current Balance')
```

Integer: Campo de tipo Entero.

```
current_balance2 = fields.Integer(string='Current Balance')
```

Many2one: se utilizan para guardar datos de tipo de muchos a uno.

```
currency_id = fields.Many2one('res.currency', string='Account',
tracking=True, help="Forces all journal")

company_id = fields.Many2one('res.company', required=True,
readonly=True, default=lambda self: self.env.company)
```

Many2many: se utilizan para guardar datos de tipo de muchos a muchos.

```
tax_ids = fields.Many2many('account.tax',
'account_account_tax_default_rel',
        'account_id', 'tax_id', string='Default Taxes',
        check_company=True,
        context={'append_type_to_tax_name': True})
```

Monetary: de tipo moneda.

```
opening_debit = fields.Monetary(string="Opening Debit",
inverse='_set_opening_debit')
```

Computados: campos que dependen de una función para calcularse.

```
reconcile = fields.Boolean(string='Allow Reconciliation',
tracking=True,
        compute='_compute_reconcile', store=True, readonly=False,
        help="Check this box if this account allows invoices &
payments matching of journal items.")


@api.depends('account_type')
    def _compute_reconcile(self):
        for account in self:
            account.reconcile = account.account_type in
('asset_receivable', 'liability_payable')
```