Tema 3

Uso de Odoo como BackEnd

1.Odoo como Backend

Odoo ofrece una arquitectura sólida y flexible que te permite desarrollar y exponer tus propias API para interactuar con los datos y la lógica empresarial dentro del sistema.

La API RESTful de Odoo te permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los objetos y modelos de datos existentes en Odoo. Puedes definir tus propios modelos de datos personalizados según las necesidades de tu aplicación y luego exponer estos modelos a través de la API RESTful para que puedan ser consumidos por aplicaciones frontales, sistemas externos u otras herramientas.

Con la API RESTful de Odoo, puedes acceder y manipular datos empresariales como clientes, productos, facturas, órdenes de venta, entre otros, utilizando estándares de protocolo web como HTTP y JSON. Esto te brinda la flexibilidad para integrar Odoo con otros sistemas, construir aplicaciones personalizadas o incluso desarrollar servicios web independientes que se comuniquen con el backend de Odoo.

2.Razones para usar Odoo como Backend

Odoo ofrece una serie de ventajas como backend para aplicaciones empresariales que lo hacen una elección atractiva frente a otras tecnologías. Una de las razones principales es la integración completa de funcionalidades empresariales que ofrece. Con Odoo, obtienes un conjunto completo de módulos integrados que abarcan desde ventas y compras hasta contabilidad, inventario y recursos humanos. Esto significa que puedes construir una aplicación completa sin tener que integrar múltiples sistemas o tecnologías, simplificando el desarrollo y la gestión de tu aplicación.

3.¿Cómo se hace?

Para convertir un módulo de Odoo en un backend con una API RESTful, debes comenzar definiendo los modelos de datos que deseas exponer a través de la API.

Una vez que hayas definido tus modelos de datos, deberás agregar controladores web en Python que manejen las solicitudes HTTP entrantes y generen las respuestas para la API RESTful. Estos controladores serán responsables de procesar las solicitudes y realizar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los modelos de datos.

Después de implementar los métodos CRUD en tus controladores web, necesitarás definir funciones de serialización para convertir los objetos de tus modelos de datos en formato JSON. Esto te permitirá enviar datos estructurados a través de la API RESTful de tu backend Odoo.

Luego, configurarás las rutas de la API RESTful en el archivo de manifestación de tu módulo (__manifest__.py), donde definirás las rutas y las vincularás con los controladores web correspondientes. Es importante seguir las convenciones RESTful para las rutas y los métodos HTTP, asegurándote de que sean descriptivas y coherentes.

Finalmente, no te olvides de documentar tu API RESTful para que los usuarios puedan entender cómo interactuar con ella. Además, considera la seguridad de tu API implementando

autenticación y autorización adecuadas para proteger los datos del sistema y garantizar la integridad y la privacidad de la información.

4.Ejemplo Práctico: Gestión de Tareas

a) Crear Modelos

```python
from odoo import models, fields

class User(models.Model):
    _name = 'custom_app.user'
    _description = 'User Model'

    name = fields.Char(string='Name', required=True)
    tasks = fields.Many2many('custom_app.task', string='Tasks')

class Task(models.Model):
    _name = 'custom_app.task'
    _description = 'Task Model'

    name = fields.Char(string='Name', required=True)
    users = fields.Many2many('custom_app.user', string='Users')
```

b) Crear controladores Web

```python
from odoo import http
from odoo.http import request
from odoo.exceptions import ValidationError

class UserController(http.Controller):

    @http.route('/api/users', auth='public', methods=['POST'], type='json')
    def create_user(self, **kwargs):
        try:
            user = request.env['custom_app.user'].sudo().create(kwargs)
            return {'success': True, 'message': 'User created successfully', 'user_id': user.id}
        except ValidationError as e:
            return {'success': False, 'message': e.name}

    @http.route('/api/users/<int:user_id>', auth='public', methods=['GET'], type='json')
    def get_user(self, user_id, **kwargs):
        user = request.env['custom_app.user'].sudo().browse(user_id)
        if user.exists():
            return {'success': True, 'user': {'id': user.id, 'name': user.name}}
        else:
            return {'success': False, 'message': 'User not found'}

    @http.route('/api/users/<int:user_id>', auth='public', methods=['DELETE'], type='json')
    def delete_user(self, user_id, **kwargs):
        user = request.env['custom_app.user'].sudo().browse(user_id)
        if user.exists():
            user.unlink()
            return {'success': True, 'message': 'User deleted successfully'}
        else:
            return {'success': False, 'message': 'User not found'}
```

Además, debes asegurarte de que el archivo __init__.py dentro del directorio del controlador web (controllers) tenga una importación adecuada del controlador para que Odoo pueda encontrarlo y cargarlo correctamente.

```python
'controllers': [
    'controllers/user_controller.py',
],
```