Taller:

Análisis Comparativo de ERP con Integración de Datos Climáticos

Presentado por:

Juliana Parra Caro
Frank Kenner Olmos Prada
Daniel Aguilar Castro
Jenny Catherine Herrera Garzón
Eduar Mauricio Mendez Mendez

Profesor:
Sergio Enrique Vargas Pedraza

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Sistemas de Información
2025 - II
30 de Septiembre de 2025

Análisis Comparativo de ERP con Integración de Datos Climáticos

Los sistemas de planificación de recursos empresariales (ERP) tradicionales, aunque robustos en su funcionalidad central, no siempre ofrecen la flexibilidad ni la capacidad de integración necesaria para incorporar datos climáticos o de polución de manera ágil y económica. Esto limita la capacidad de las empresas para tomar decisiones informadas que optimicen su cadena de suministro, reduzcan el impacto ambiental y mejoren la resiliencia operativa.

Este trabajo tiene como objetivo realizar un análisis comparativo de sistemas ERP con capacidad de integración de datos climáticos, evaluando criterios funcionales, técnicos, económicos y organizativos que permitan identificar la plataforma más adecuada.

1.Criterios Funcionales

Módulos disponibles para todas las áreas
Los ERPs empresariales tradicionales (SAP, Oracle, Dynamics) ofrecen cobertura completa para todas las áreas funcionales, pero con complejidad significativa. Odoo proporciona una suite modular integral que cubre ventas, inventario, manufactura, contabilidad y RRHH, con la ventaja de una arquitectura unificada y coherente.

Capacidad de integración con otros sistemas
Mientras los ERPs corporativos requieren middleware complejo y costoso, Odoo ofrece APIs REST nativas y un framework de integración simplificado. La integración con APIs climáticas se realiza directamente mediante Python, sin capas adicionales.

Desarrollos a medida
Los ERPs tradicionales implican costos prohibitivos para personalizaciones. Odoo, siendo open-source, permite desarrollos personalizados ilimitados con Python, framework nativo y comunidad de desarrolladores accesible.

2.Criterios Técnicos

Plataformas y bases de datos
Odoo soporta PostgreSQL, Linux y despliegues cloud/híbridos con menor overhead que soluciones empresariales que requieren infraestructura especializada.

Lenguajes y herramientas
Python como lenguaje nativo de Odoo versus ABAP (SAP), Java (Oracle) o .NET (Dynamics). Python ofrece:

- Curva de aprendizaje más suave
- Ecosistema robusto para integraciones API

- Mantenimiento más económico
- Comunidad activa y recursos abundantes

Compatibilidad con estándares web
Odoo maneja nativamente XML, JSON, EDI y APIs REST, simplificando la conexión con servicios climáticos como OpenWeatherMap.

3.Criterios Económicos

Costos de licencias e implementación

- ERPs tradicionales: Licencias por usuario (+$100-300/usuario/mes), implementación 6-18 meses, consultoría especializada costosa
- Odoo Community: $0 en licencias, implementación 2-4 meses
- Odoo Enterprise: Costos significativamente menores que competidores

Mantenimiento y actualizaciones
Actualizaciones en Odoo son menos disruptivas y más económicas, con migraciones simplificadas versus procesos complejos en ERPs corporativos.

Análisis ROI
La integración clima-ERP en Odoo ofrece:

- Tangible: Optimización logística, reducción de mermas, eficiencia energética
- Intangible: Toma de decisiones proactiva, ventaja competitiva, resiliencia operativa

4.Criterios Organizativos

Impacto en procesos
Odoo implica menor disruptividad organizacional versus transformaciones radicales de ERPs corporativos.

Facilidad de uso
Interfaz intuitiva y moderna de Odoo versus interfaces complejas de sistemas tradicionales.

Gestión del cambio
Curva de aprendizaje más suave, capacitación más accesible y resistencia al cambio reducida

5.Evaluación de Proveedores

Dentro de los proveedores considerados se encuentran Microsoft Dynamics 365, SAP/Oracle, NetSuite y Odoo. La mayoría de de ERP modernos tienen herramientas integradas para el clima, pero luego de realizar un balance de varios Servicios de API del clima integrados en diferentes ERPs, se decidió que el proveedor que más se ajusta a la necesidad es Odoo.

Experiencia en empresas similares
Odoo cuenta con implementaciones exitosas en PYMEs y medianas empresas, con casos de uso relevantes para el objetivo climático.

Servicio de soporte
Ecosistema de partners locales, documentación extensa y soporte comunitario robusto.

Referencias del sector
Numerosas implementaciones en logística, distribución y manufactura con integraciones climáticas similares.

Conlusiones

La App Climática en Odoo con API Python es la mejor opción por los siguientes puntos:

1. Arquitectura Nativa Óptima
    a. Python como lenguaje nativo de Odoo permite integración seamless con APIs climáticas
    b. Desarrollo de módulo específico sin dependencias externas complejas
2. Costo-Beneficio Superior
    a. Elimina costos de licencias de ERP corporativo
    b. Implementación acelerada (semanas vs meses)
    c. Mantenimiento con recursos Python accesibles vs especialistas SAP/Oracle
3. Flexibilidad Funcional
    a. Módulo climático personalizable para necesidades específicas
    b. Integración directa con módulos de logística, inventario y planeación
4. Sostenibilidad Técnica
    a. Stack tecnológico moderno y futuro-proof
    b. Comunidad activa de desarrollo y soporte
    c. Actualizaciones no disruptivas
5. ROI Acelerado
    a. Beneficios operativos desde el primer día de implementación
    b. Costo total de propiedad significativamente inferior
    c. Escalabilidad progresiva según necesidades

La combinación Odoo + Python + API Climática Meteomatics ofrece la solución más elegante, económica y técnicamente sólida para integrar datos meteorológicos en los

procesos empresariales, superando ampliamente las alternativas corporativas tradicionales en agilidad, costo y eficiencia.

EJEMPLO DE IMPLEMENTACIÓN [1]

manifest.py

```
{
    'name': 'Weather Integration',
    'version': '1.0',
    'category': 'Tools',
    'description': 'Fetch and store weather data via API',
    'depends': ['base'],
    'data': [
        'security/ir.model.access.csv',
        'views/weather.xml',
    ],

    'installable': True,
    'application': True,
    'license': 'LGPL-3',
}
```

weather.py

```python
from odoo import models, fields, api
import meteomatics.api as meteomatics_api
import datetime as dt
import logging

_logger = logging.getLogger(__name__)

class WeatherData(models.Model):
    Create Views | Create Report | Import Model | Import ir.model.access.csv
    _name = 'weather.data'
    _description = 'Weather Data'

    location = fields.Char(string='Location')
    temperature = fields.Float(string='Temperature')
    condition = fields.Char(string='Condition', default='Clear')

    def fetch_and_store_weather_data(self):
        # Fetch weather data from the Meteomatics API
        weather_data = self._fetch_weather_data()
        if weather_data:
            # Log the data to be created
            print(f"Creating record with data: {weather_data}")
            # Create a new record with the fetched data
            self.create({
                'location': weather_data['location'],
                'temperature': weather_data['temperature'],
                'condition': weather_data['condition']
            })
```

```python
def _fetch_weather_data(self):
    # Meteomatics API credentials
    username = 'odooistic_rajput_farooq'
    password = '7HRQ9smnz9'

    coordinates = [(51.5073219, -0.1276474)]  # London coordinates
    parameters = ['t_2m:C']  # Temperature at 2 meters
    model = 'mix'

    # Set start and end date for the request
    startdate = dt.datetime.utcnow().replace(minute=0, second=0, microsecond=0)
    enddate = startdate + dt.timedelta(hours=1)  # Forecast for the next hour
    interval = dt.timedelta(hours=1)

    try:
        # Query the time series from Meteomatics
        df = meteomatics_api.query_time_series(coordinates, startdate, enddate, interval, parameters, username, password, model=model)

        # Log the entire response to check its structure
        print(f"API response: {df}")

        # Parse the temperature value from the DataFrame
        temperature = df['t_2m:C'].iloc[0]  # Correct way to access column data by column name and row index

        # Log the extracted temperature to verify it's correct
        print(f"Extracted temperature: {temperature}")

        return {
            'temperature': temperature,
            'location': 'London',
            'condition': 'Clear'  # Default to 'Clear'
        }
    except Exception as e:
        _logger.error(f"Error fetching weather data: {str(e)}")
        return None
```

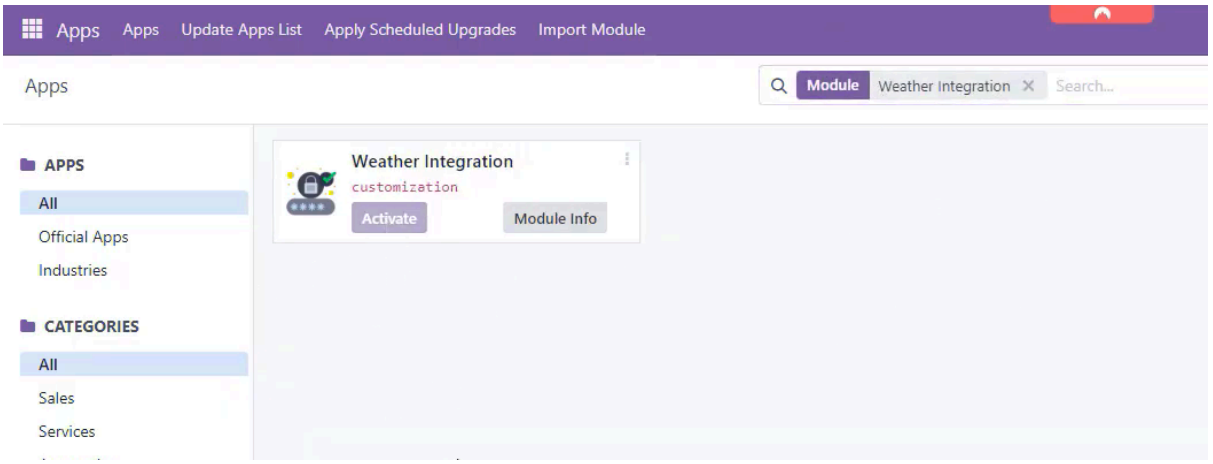
weather.xml

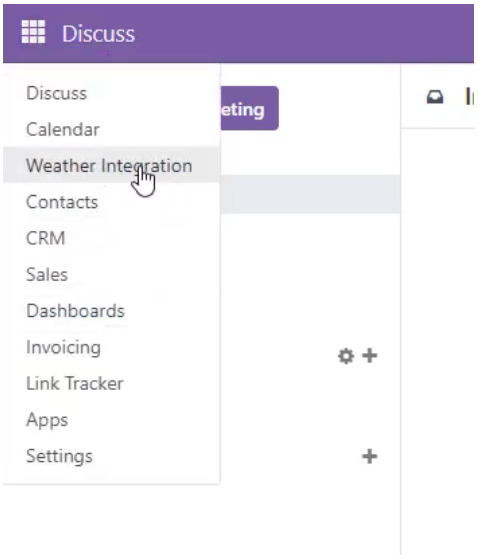
```xml
<odoo>
    <!-- Define Form View for Weather Data -->
    <record id="view_weather_data_form" model="ir.ui.view">
        <field name="name">weather.data.form</field>
        <field name="model">weather.data</field>
        <field name="arch" type="xml">
            <form string="Weather Data">
                <sheet>
                    <div class="oe_button_box" name="button_box">
                        <!-- Corrected Button with Combined Classes -->
                        <button class="oe_stat_button btn-primary" name="fetch_and_store_weather_data"
                                type="object" icon="fa-cloud">
                            <div class="o_stat_info">
                                <span class="o_stat_value">Fetch</span>
                                <span class="o_stat_text">Weather</span>
                            </div>
                        </button>
                    </div>

                    <group>
                        <field name="location"/>
                        <field name="temperature"/>
                        <field name="condition"/>
                    </group>
                </sheet>
            </form>
        </field>
    </record>


    <!-- Define Tree View for Weather Data -->
    <record id="view_weather_data_tree" model="ir.ui.view">
        <field name="name">weather.data.tree</field>
        <field name="model">weather.data</field>
        <field name="arch" type="xml">
            <tree string="Weather Data">
                <field name="location"/>
                <field name="temperature"/>
                <field name="condition"/>
            </tree>
        </field>
    </record>

    <!-- Define the Action to open the Weather Data Views -->
    <record id="action_weather_data" model="ir.actions.act_window">
        <field name="name">Weather Data</field>
        <field name="res_model">weather.data</field>
        <field name="view_mode">tree,form</field>
    </record>

    <!-- Define the Menu Item -->
    <menuitem id="menu_weather_data_root" name="Weather Integration" sequence="10"/>
    <menuitem id="menu_weather_data" name="Weather Data" parent="menu_weather_data_root" action="action_weather_data" sequence="10"/>
</odoo>
```

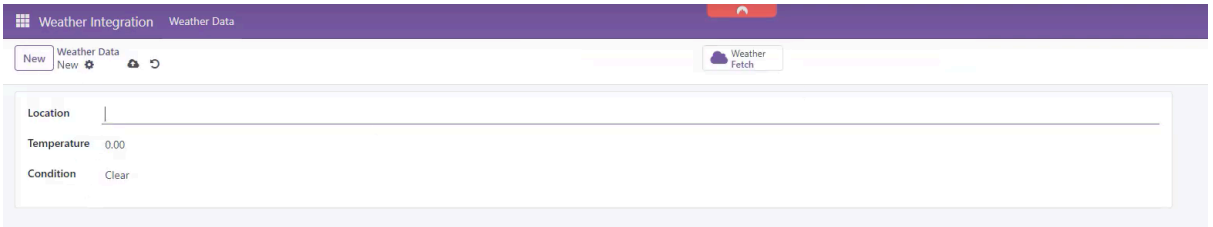
ir.model.access.csv

```csv
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
model_weather_weather_data,model.weather.weather.data,model_weather_data,,1,1,1,1
```

Resultado en odoo







Referencias

[1]"'Build an Odoo Weather App with API Integration | Step-by-Step Guide' ," *YouTube*. [Online]. Disponible en: https://www.youtube.com/watch?v=RMsOKv6OTGQ. [Revisado: 28-Sep-2025]