ARQUITECTURA DE SOFTWARE

**Guía 3**

# MODELO C4 Y PRINCIPIOS SOLID

SEBASTIAN LOPEZ

JUAN SEBASTIAN MENDOZA MONCADA

JOSEPH IMANOL REYES CHAPARRO

DOCENTE:
CARLOS EDUARDO MUJICA REYES

UNIVERSIDAD MANUELA BELTRAN

2025

**Descripción ERP**

El ERP Obleas la Villa está diseñado para gestionar los procesos de producción, distribución y venta de productos derivados de la leche. Sus funcionalidades principales incluyen:

**Gestión de compras e inventario**: Control de materias primas y productos terminados.

**Gestión de clientes y proveedores:** Registro, seguimiento y relaciones comerciales.

**Recursos humanos y nómina:** Administración de empleados, cálculo de salarios y pagos.

**Contabilidad:** Registro de transacciones, ingresos, egresos y generación de reportes financieros.

**Ventas y pedidos**: Procesamiento de órdenes, control de pagos y facturación.

**Módulo de pagos:** Integración con sistemas de pago para transacciones seguras.

## 2. Trade-offs (Compromisos y decisiones de diseño)

Simplicidad vs. Complejidad:

Se priorizó una arquitectura modular para facilitar la escalabilidad, sacrificando un diseño monolítico más simple.

Costos vs. Rendimiento:

Se optó por tecnologías de código abierto para reducir costos, aunque esto puede implicar una mayor curva de aprendizaje y configuración.

Seguridad vs. Facilidad de acceso:

Se implementó autenticación y control de permisos para proteger los datos, aunque esto requiere una mayor gestión de usuarios.

Flexibilidad vs. Especialización:

Se eligió una arquitectura flexible basada en microservicios, lo que permite personalización, pero aumenta la complejidad del mantenimiento.

## 3. Tecnologías seleccionadas

**Backend:**

Lenguaje: Python

Framework: Django / FastAPI (según necesidad de rendimiento)

Base de datos: PostgreSQL / SQLite (según entorno)

**Frontend:**

Aplicación web: React.js

Estilización: Tailwind CSS

**Infraestructura:**

Hosting: Hostinger / VPS privado

Seguridad: Certificado SSL y autenticación JWT

Contenedores: Docker para despliegue escalable

**Módulos adicionales:**

Pagos: Integración con Stripe o PayPal

Autenticación: OAuth 2.0 / Firebase Auth

## Patrones de Diseño Aplicados en el ERP Obleas la Villa

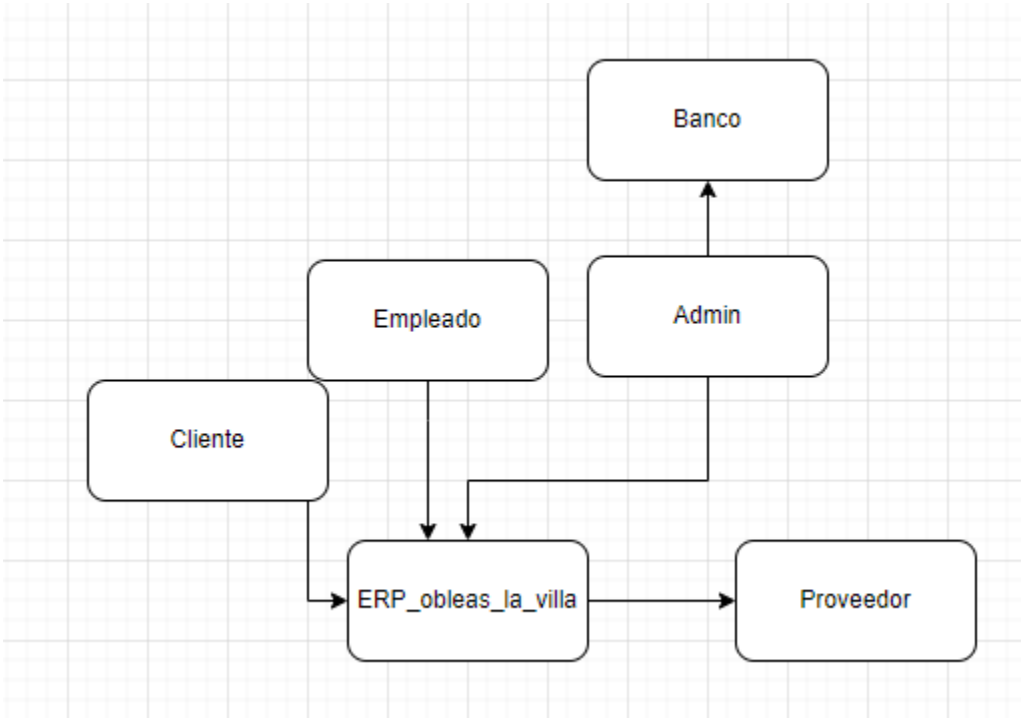
1. **Patrón MVC (Modelo-Vista-Controlador)**
   - **Aplicación en el ERP:**
     - **Modelo:** Representa entidades como `Empleado`, `Cliente`, `Producto`, `Pedido`.
     - **Vista:** Interfaz web donde los usuarios pueden gestionar compras, ventas, inventario, etc.
     - **Controlador:** API en Python que maneja la comunicación entre la base de datos y la vista.
2. **Patrón Singleton**
   - Se aplica para la gestión de la conexión a la base de datos y configuración del sistema.
   - **Ejemplo en el ERP:**
     - La instancia de la base de datos es única y reutilizada en todo el sistema para evitar múltiples conexiones innecesarias.
3. **Patrón Observador**
   - Se usa para notificar módulos cuando ocurre un evento importante.
   - **Ejemplo en el ERP:**
     - Cuando un pedido se completa, el sistema notifica al módulo de inventario para actualizar el stock automáticamente.
4. **Patrón Factory Method**
   - Se emplea para la creación de objetos sin acoplarse a clases específicas.
   - **Ejemplo en el ERP:**
     - Una fábrica de productos que genera diferentes tipos de productos (`Obleas`, `Dulces`, `Lácteos`) según la categoría.

## Principios SOLID Aplicados en el ERP Obleas la Villa

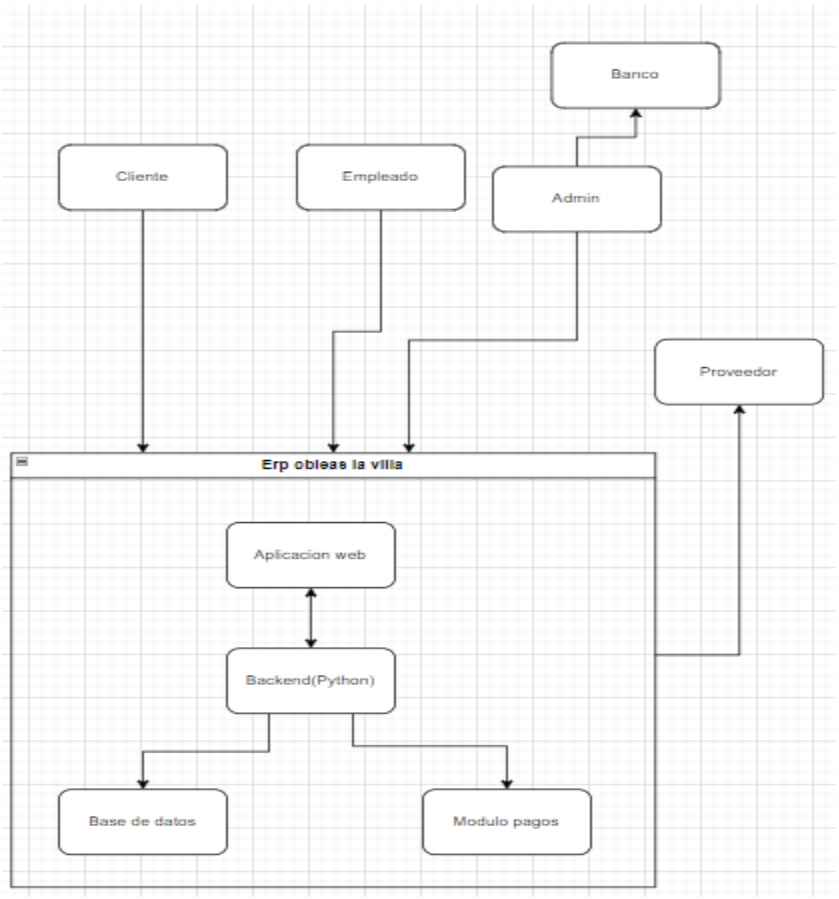
1. **S - Principio de Responsabilidad Única (SRP)**
   - Cada módulo y clase tiene una única responsabilidad.
   - **Ejemplo en el ERP:**
     - `InventarioService` solo gestiona la lógica de inventario.
     - `PagoService` maneja los pagos sin interferir con la contabilidad.
2. **O - Principio de Abierto/Cerrado (OCP)**
   - El sistema es extensible sin modificar código existente.
   - **Ejemplo en el ERP:**
     - Si se agrega un nuevo método de pago, se implementa una nueva clase que extiende `PagoService` sin modificar las clases existentes.
3. **L - Principio de Sustitución de Liskov (LSP)**
   - Las clases hijas pueden reemplazar a las clases base sin afectar el comportamiento del sistema.
   - **Ejemplo en el ERP:**
     - `Empleado` es una clase base y `Gerente` y `Cajero` son subclases que pueden usarse sin afectar la lógica del sistema.
4. **I - Principio de Segregación de Interfaces (ISP)**
   - Se crean interfaces específicas en lugar de una interfaz gigante.
   - **Ejemplo en el ERP:**
     - `IReporteVentas` para la generación de reportes de ventas.
     - `IGestionInventario` para la administración del stock.
5. **D - Principio de Inversión de Dependencias (DIP)**
   - Se usan abstracciones en lugar de depender de implementaciones concretas.
   - **Ejemplo en el ERP:**
     - Se implementan interfaces para `BaseDatosService`, permitiendo cambiar de SQLite a PostgreSQL sin modificar la lógica del sistema.

Diagrama C4

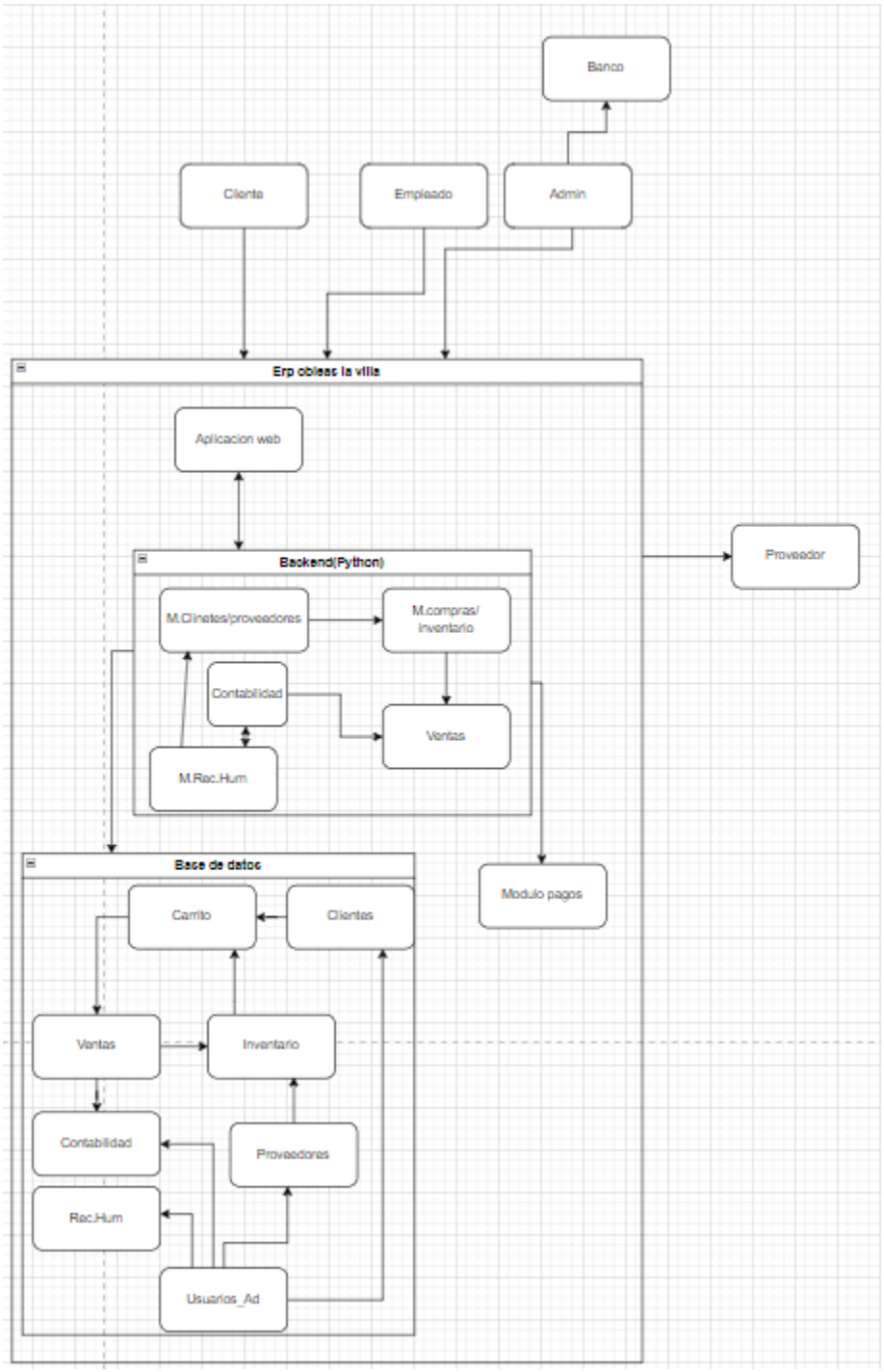
- Contexto



- Contenedores



-Componentes



-Código