PORTAL SAAS DE ODOO - Plan Estratégico y Arquitectura Lógica

Proyecto: Sistema Automatizado de Venta y Provisión de Instancias Odoo
Metodología: Scrum Adaptado
Duración Estimada: 8-12 semanas
Equipo: 2 desarrolladores (Backend/DevOps + Frontend/Odoo)
Versión del Documento: 1.0

📋 ÍNDICE EJECUTIVO

---

1. VISIÓN GENERAL

1.1 ¿Qué Estamos Construyendo?

Un **sistema de venta automatizada de instancias Odoo** que funciona como un servicio SaaS. El cliente entra a una página web, compra una suscripción, y automáticamente recibe su propia instalación completa de Odoo sin intervención humana.

1.2 El Problema que Resolvemos

Situación Actual:

- Las empresas necesitan Odoo pero no tienen conocimiento técnico
- Contratar servidores y configurar Odoo es complejo y costoso
- El Odoo SaaS oficial es caro ($24+ por usuario/mes)
- Las alternativas requieren contratar consultores

Nuestra Solución:

- Portal web simple donde cualquiera puede comprar
- Precio accesible (modelo de suscripción mensual)
- Instancia propia e independiente (no compartida)
- Cero conocimiento técnico requerido
- Todo automatizado desde el pago hasta la entrega

1.3 Modelo de Negocio

Cliente paga $10/mes → Recibe su Odoo completo → Usa el sistema para su negocio
↓
Si no paga el segundo mes → Sistema se suspende automáticamente
↓
Paga de nuevo → Sistema se reactiva inmediatamente

Escalabilidad:
Con 100 clientes = $1,000/mes de ingresos recurrentes
Con 1,000 clientes = $10,000/mes de ingresos recurrentes
Sin aumentar personal (todo es automático)

---

2. ANÁLISIS DEL MODELO DE REFERENCIA

2.1 Estudio del Webkul SaaS Kit

Características Principales de un SaaS Kit Profesional:

1. **Multi-Tenancy (Multi-Inquilino)**
    - Cada cliente tiene su propia base de datos
    - Aislamiento total entre clientes
    - Gestión centralizada desde un panel maestro
2. **Provisión Automática**
    - Creación instantánea de instancias
    - Configuración automática de base de datos
    - Asignación de subdominios o URLs únicas
3. **Gestión de Planes**
    - Múltiples niveles de suscripción (Básico, Pro, Enterprise)
    - Límites configurables (usuarios, almacenamiento, módulos)
    - Upgrades/Downgrades automáticos
4. **Control de Ciclo de Vida**
    - Monitoreo de estado de pago
    - Suspensión automática por mora
    - Reactivación al confirmar pago
    - Eliminación de datos después de X días de impago
5. **Panel de Administración**
    - Dashboard para ver todas las instancias

    - Métricas de uso y rendimiento
    - Gestión manual de instancias (si es necesario)

2.2 Lo que Vamos a Replicar (Versión Gratuita)

Basándonos en ese modelo, nuestra versión construida con herramientas gratuitas incluirá:

Funcionalidad	Implementación Propia
Multi-Tenancy	Docker containers (un contenedor por cliente)
Provisión Automática	Scripts Python + comandos Docker
Gestión de Planes	Productos en eCommerce de Odoo
Control de Ciclo de Vida	Cron Jobs de Odoo + subprocess Python
Panel de Administración	Vistas personalizadas en Odoo
Portal del Cliente	Portal integrado de Odoo

2.3 Diferencias Clave

Webkul SaaS Kit (Pago):

- Todo integrado en un solo módulo
- Interfaz gráfica para todo
- Soporte técnico incluido
- Actualizaciones automáticas

Nuestra Solución (Gratuita):

- Construimos cada componente nosotros
- Más trabajo inicial, más control
- Sin costos de licencia
- Aprendizaje profundo del sistema

---

3. ARQUITECTURA CONCEPTUAL

3.1 Los Tres Niveles del Sistema

NIVEL 1: PRESENTACIÓN (Lo que ve el cliente)

↓ Interacción

NIVEL 2: LÓGICA DE NEGOCIO (El cerebro)

↓ Ejecuta acciones

NIVEL 3: INFRAESTRUCTURA (Donde vive todo)

3.2 Arquitectura de Dos Mundos

MUNDO 1: El Odoo Maestro (Sistema de Control)

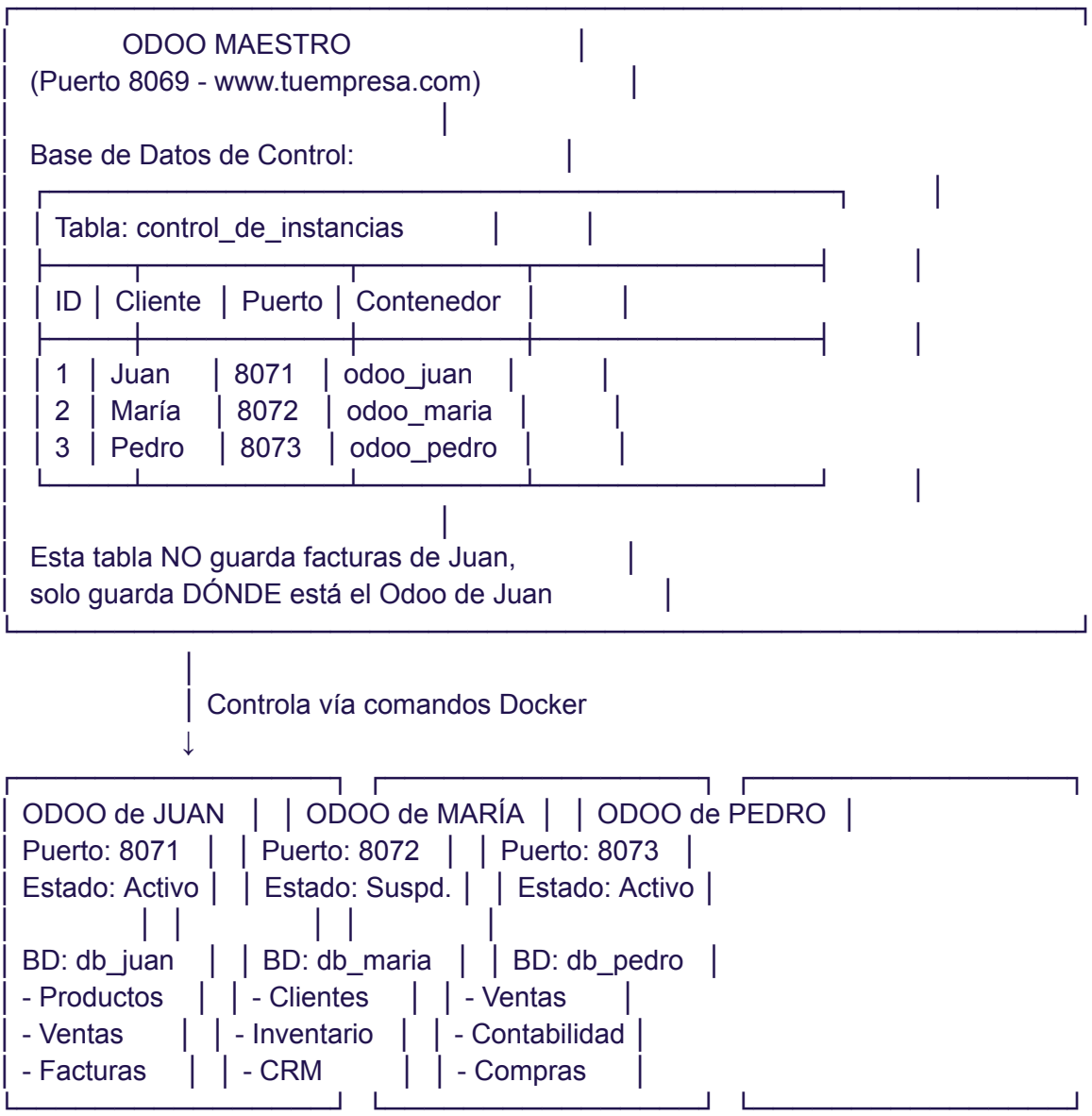
- **Función:** Es la "fábrica" que crea y gestiona las instancias
- **Ubicación:** Un solo contenedor Docker permanente
- **Contiene:**
    - El sitio web público
    - El eCommerce
    - La base de datos de control (NO los datos de los clientes)
    - El módulo personalizado que automatiza todo

MUNDO 2: Los Odoo de Clientes (Productos)

- **Función:** Son los "productos" que se entregan a cada cliente
- **Ubicación:** Múltiples contenedores Docker (uno por cliente)

- **Características:**
  - Cada uno es completamente independiente
  - Tienen su propia base de datos
  - Pueden ser iniciados, detenidos o eliminados
  - No se comunican entre sí

3.3 Relación entre Ambos Mundos



---

4. COMPONENTES DEL SISTEMA

4.1 Componente 1: Portal Web Público (Frontend)

¿Qué es?
La "cara" del negocio. Es lo primero que ven los potenciales clientes.

¿Qué hace?

- Mostrar información sobre el servicio
- Listar los planes disponibles (Básico, Pro, Enterprise)
- Permitir registro de nuevos usuarios
- Procesar compras

¿Con qué se construye?

- **Herramienta:** Odoo Website Builder
- **Por qué:** Ya viene incluido en Odoo Community (gratis), tiene editor visual drag-and-drop, no necesitas programar HTML/CSS

¿Cómo funciona?

1. Usuario entra a www.tuempresa.com
2. Ve una página bonita con información
3. Hace clic en "Ver Planes"
4. Selecciona un plan (ej: "Plan Básico - $10/mes")
5. Hace clic en "Comprar Ahora"

Elementos visuales necesarios:

- Página de inicio (Home)
- Página de precios/planes
- Página "Cómo funciona"
- Botón de registro/login
- Carrito de compras

4.2 Componente 2: Sistema de eCommerce

¿Qué es?
El motor de ventas. Gestiona el carrito, checkout y procesamiento de pagos.

¿Qué hace?

- Agregar productos al carrito
- Calcular totales
- Procesar información de pago
- Confirmar órdenes de venta

¿Con qué se construye?

- **Herramienta:** Odoo eCommerce (módulo `website_sale`)
- **Extensión:** Odoo Sales (módulo `sale`)

- **Por qué:** Sistema completo de ventas ya probado, integración nativa con pasarelas de pago

¿Cómo funciona?

1. Cliente selecciona "Plan Básico"
2. Se agrega al carrito como un producto
3. Cliente hace checkout
4. Ingresa datos de pago (tarjeta)
5. El pago se procesa vía Stripe/PayPal
6. Si es exitoso, se crea una "Orden de Venta"

Configuración necesaria:

- Crear productos (Plan Básico, Plan Pro, etc.)
- Configurar precios
- Activar pasarela de pago (Stripe o PayPal)
- Configurar impuestos si aplica

4.3 Componente 3: Procesamiento de Pagos

¿Qué es?
El puente entre el banco del cliente y nuestro sistema.

¿Qué hace?

- Recibe información de tarjeta (de forma segura)
- Procesa el cargo
- Notifica a Odoo si fue exitoso o no

¿Con qué se construye?

- **Herramienta:** Stripe o PayPal (servicios externos)
- **Integración:** Payment Provider de Odoo
- **Por qué:** No podemos procesar tarjetas directamente (requerimientos legales PCI-DSS), estos servicios lo hacen por nosotros

¿Cómo funciona?

1. Cliente ingresa datos de tarjeta
2. Los datos van **directamente** a Stripe (no pasan por nuestro servidor)
3. Stripe cobra el dinero
4. Stripe envía un "webhook" (notificación) a nuestro Odoo
5. Odoo marca la orden como "Pagada"

Concepto clave: Webhook

- Es como un "timbre" que Stripe toca en nuestra puerta
- Nos dice: "Hey, el pago de la orden #123 fue exitoso"
- Nuestro sistema escucha ese timbre y actúa

4.4 Componente 4: Motor de Automatización (El Cerebro)

¿Qué es?
El código personalizado que hace la "magia". Es lo que TÚ vas a programar.

¿Qué hace?

- Detecta cuando se confirma una venta
- Crea automáticamente el contenedor Docker del cliente
- Configura la base de datos
- Envía las credenciales por email
- Monitorea fechas de expiración
- Suspende/reactiva instancias

¿Con qué se construye?

- **Lenguaje:** Python
- **Framework:** Odoo ORM
- **Librerías:** subprocess (para ejecutar comandos Docker)
- **Por qué:** Python es el lenguaje nativo de Odoo, el ORM facilita trabajar con la base de datos

¿Cómo funciona? (Flujo completo)

TRIGGER: Pago confirmado
↓
PASO 1: Detectar el evento
→ Odoo ejecuta el método action_confirm() de sale.order
→ Nuestro código "hereda" ese método
→ Agregamos nuestra lógica
↓
PASO 2: Validar que es una suscripción
→ ¿El producto vendido es "Plan Odoo"?
→ Si NO → continuar normal
→ Si SÍ → ejecutar PASO 3
↓
PASO 3: Preparar datos para la instancia
→ Obtener nombre del cliente
→ Generar nombre único para contenedor: "odoo_cliente_123"
→ Calcular siguiente puerto disponible: 8071
→ Generar contraseña temporal aleatoria
→ Calcular fecha de expiración: hoy + 30 días
↓
PASO 4: Registrar en base de datos de control
→ Crear un registro en tabla "saas.instance"
→ Guardar: cliente, puerto, contenedor, fecha, estado="creando"
↓
PASO 5: Ejecutar comando Docker
→ Construir comando: "docker run -d --name odoo_cliente_123..."

→ Ejecutar vía subprocess.run()
→ Esperar confirmación
↓
PASO 6: Actualizar estado
→ Cambiar estado a "activo"
→ Guardar ID del contenedor
↓
PASO 7: Notificar al cliente
→ Enviar email con:
- URL de acceso
- Usuario (admin)
- Contraseña temporal
↓
FIN: Cliente recibe email y puede empezar a usar su Odoo

4.5 Componente 5: Gestión de Contenedores Docker

¿Qué es?
La tecnología que permite crear múltiples "Odoos mini" en un solo servidor.

¿Qué hace?

- Crea entornos aislados (contenedores)
- Cada contenedor es como una computadora virtual ligera
- Puede iniciar, detener, reiniciar contenedores
- Asigna recursos (CPU, RAM) a cada uno

¿Con qué se construye?

- **Herramienta:** Docker Engine
- **Orquestación:** Docker Compose (para el maestro), comandos directos (para clientes)
- **Por qué:** Es el estándar de la industria, ligero, rápido, portable

Conceptos clave:

Imagen vs Contenedor:

- **Imagen:** Es la "plantilla" (como un .ISO o instalador)
- **Contenedor:** Es la "instancia ejecutándose" (como un programa abierto)
- Analogía: La imagen es el molde de galletas, el contenedor es cada galleta

Comando para crear instancia de cliente:

```
docker run -d
  --name odoo_cliente_juan
  --network odoo_network
  -p 8071:8069
  -e HOST=db_juan
```

```
odoo:17
```

Desglose del comando:

- `docker run`: Crear y ejecutar un contenedor
- `-d`: Modo background (no bloquea la terminal)
- `--name`: Nombre único del contenedor
- `--network`: Red para que se comunique con su base de datos
- `-p 8071:8069`: "El puerto 8071 del servidor apunta al puerto 8069 del contenedor"
- `-e HOST=db_juan`: Variable de entorno (dónde está su base de datos)
- `odoo:17`: Imagen base a usar

4.6 Componente 6: Sistema de Bases de Datos

¿Qué es?
Donde se almacena TODA la información.

¿Qué hace?

- Almacena datos del Odoo Maestro
- Almacena datos de cada cliente (en bases de datos separadas)
- Procesa consultas (SELECT, INSERT, UPDATE)

¿Con qué se construye?

- **Herramienta:** PostgreSQL
- **Versión:** 15 o superior
- **Por qué:** Es el único motor de BD que soporta Odoo, es robusto, gratis, open-source

Estructura de bases de datos:

```
SERVIDOR POSTGRESQL
│
├── Base de Datos: odoo_maestro
│   ├── Tabla: res_partner (clientes que compran)
│   ├── Tabla: sale_order (órdenes de venta)
│   ├── Tabla: saas_instance (control de instancias)
│   └── Tabla: product_product (planes de suscripción)
│
├── Base de Datos: db_cliente_juan
│   ├── Tabla: res_partner (clientes de Juan)
│   ├── Tabla: sale_order (ventas de Juan)
│   ├── Tabla: account_move (facturas de Juan)
│   └── ... (todas las tablas de Odoo estándar)
│
├── Base de Datos: db_cliente_maria
│   └── ... (datos de María, aislados)
│
```

└── Base de Datos: db_cliente_pedro
    └── ... (datos de Pedro, aislados)

Aislamiento:

- Juan NO puede ver datos de María
- Cada base de datos es completamente independiente
- Si María borra todo, solo afecta su BD

4.7 Componente 7: Portal del Cliente

¿Qué es?
La sección privada donde el cliente gestiona su suscripción.

¿Qué hace?

- Mostrar estado de la suscripción (Activo/Suspendido)
- Botón para acceder a su Odoo
- Ver fecha de próximo cobro
- Actualizar método de pago
- Ver historial de facturas

¿Con qué se construye?

- **Herramienta:** Odoo Portal (módulo `portal`)
- **Customización:** Templates XML personalizados
- **Por qué:** Odoo ya tiene un sistema de portal para clientes, solo lo extendemos

¿Cómo funciona?

1. Cliente hace login en www.tuempresa.com
2. Ve un menú "Mi Cuenta"
3. Dentro ve:
    - Mis Instancias Odoo
    - Mis Facturas
    - Mi Información
4. En "Mis Instancias" ve:
    - Nombre de su instancia
    - Estado (🟢 Activo / 🔴 Suspendido)
    - Fecha de expiración
    - Botón "Acceder a mi Odoo" (si está activo)

4.8 Componente 8: Sistema de Automatización de Ciclo de Vida (Cron Jobs)

¿Qué es?
Un "robot" que trabaja en segundo plano ejecutando tareas periódicamente.

¿Qué hace?

- Cada noche a las 00:00 revisa todas las instancias
- Identifica cuáles expiraron
- Suspende las que no pagaron
- Envía recordatorios de pago

¿Con qué se construye?

- **Herramienta:** Odoo Scheduled Actions
- **Lenguaje:** Python
- **Por qué:** Odoo tiene un sistema de cron integrado, no necesitamos herramientas externas

Lógica del Cron:

```
EJECUCIÓN: Todos los días a las 00:00

PASO 1: Obtener fecha actual
   → hoy = 2026-02-05
   ↓
PASO 2: Buscar instancias problemáticas
   → Consulta SQL:
     SELECT * FROM saas_instance
     WHERE estado = 'activo'
     AND fecha_expiracion < '2026-02-05'
   ↓
PASO 3: Por cada instancia encontrada:
   ↓
   3.1: Obtener nombre de contenedor
        → ejemplo: "odoo_cliente_juan"
   ↓
   3.2: Ejecutar comando de suspensión
        → docker stop odoo_cliente_juan
        → docker stop odoo_cliente_juan_db
   ↓
   3.3: Actualizar registro en BD
        → estado = 'suspendido'
        → fecha_suspension = hoy
   ↓
   3.4: Enviar email al cliente
        → "Tu suscripción ha expirado"
        → "Paga para reactivar"
        → Link al checkout
   ↓
PASO 4: Registrar en log
   → "2026-02-05 00:00: Suspendidas 3 instancias"
   ↓
FIN
```

4.9 Componente 9: Sistema de Notificaciones (Email)

¿Qué es?
El sistema que envía correos automáticos a los clientes.

¿Qué hace?

- Enviar credenciales cuando la instancia está lista
- Recordatorios de pago antes de expiración
- Notificación de suspensión
- Confirmación de reactivación

¿Con qué se construye?

- **Herramienta:** Odoo Mail System
- **SMTP:** Gmail, SendGrid, o Mailgun
- **Por qué:** Odoo tiene sistema de emails con plantillas, solo configuramos el servidor SMTP

Plantillas de email necesarias:

1. **Email de Bienvenida:**

Asunto: ¡Tu Odoo está listo!

Hola [NOMBRE],

Tu instancia de Odoo ha sido creada exitosamente.

Accede aquí: [URL]
Usuario: admin
Contraseña: [CONTRASEÑA_TEMPORAL]

Te recomendamos cambiar tu contraseña al iniciar sesión.

¡Bienvenido!

2. **Email de Recordatorio (7 días antes):**

Asunto: Tu suscripción vence en 7 días

Hola [NOMBRE],

Tu suscripción expira el [FECHA].

Para mantener tu servicio activo, asegúrate de que tu
método de pago esté actualizado.

[BOTÓN: Actualizar Pago]

3. **Email de Suspensión:**

Asunto: Tu servicio ha sido suspendido

Hola [NOMBRE],

Tu suscripción expiró el [FECHA] y no hemos recibido el pago.

Tu instancia ha sido suspendida temporalmente.
Tus datos están seguros y serán restaurados al renovar.

[BOTÓN: Renovar Ahora]

---

5. FLUJOS DE PROCESO

5.1 Flujo Principal: Compra de Suscripción

```
┌─────────────────────────────────────────────────────────────┐
│ ACTOR: Cliente Potencial                                    │
└─────────────────────────────────────────────────────────────┘

PASO 1: Descubrimiento
   → Cliente busca en Google "Odoo barato"
   → Encuentra www.tuempresa.com
   → Entra al sitio
   ↓
PASO 2: Exploración
   → Lee la página de inicio
   → Ve los beneficios del servicio
   → Hace clic en "Ver Planes"
   ↓
PASO 3: Selección de Plan
   → Compara planes (Básico vs Pro vs Enterprise)
   → Decide: "Plan Básico - $10/mes"
   → Hace clic en "Comprar Ahora"
   ↓
PASO 4: Registro/Login
   ┌─ ¿Ya tiene cuenta?
   │  ├─ SÍ → Hacer login
   │  └─ NO → Llenar formulario de registro
   │          - Nombre
   │          - Email
   │          - Empresa
   │          - Contraseña
```

└─ Se crea usuario en Odoo Maestro
↓
PASO 5: Checkout
→ Se muestra resumen de compra
→ Se muestra total: $10.00
→ Selecciona método de pago
↓
PASO 6: Procesamiento de Pago
→ Ingresa datos de tarjeta en formulario seguro
→ Los datos van a Stripe
→ Stripe procesa el cargo
┌─ ¿Pago exitoso?
│ ├─ NO → Mostrar error, reintentar
│ └─ SÍ → Continuar PASO 7
↓
PASO 7: Confirmación de Orden
→ Odoo crea Orden de Venta
→ Estado cambia a "Confirmado"
→ Se muestra mensaje: "¡Pago recibido!"
↓
PASO 8: TRIGGER AUTOMÁTICO (Backend)
→ Nuestro módulo detecta la orden confirmada
→ Valida que es producto "Plan Odoo"
→ Inicia proceso de creación de instancia
↓
PASO 9: Creación de Instancia (invisible para cliente)
9.1: Generar datos únicos
- Nombre contenedor: odoo_cliente_123
- Puerto: 8071
- Base de datos: db_cliente_123
- Contraseña: abc123XYZ
9.2: Ejecutar comando Docker
- docker run... (crear PostgreSQL)
- docker run... (crear Odoo)
9.3: Esperar que contenedores inicien (10-15 seg)
9.4: Configurar base de datos inicial
9.5: Actualizar registro de control
↓
PASO 10: Notificación al Cliente
→ Sistema envía email automático
→ Email contiene:
- URL de acceso
- Credenciales
- Instrucciones básicas
↓
PASO 11: Cliente Recibe Email
→ Abre el email (puede tardar 1-2 minutos)
→ Lee las credenciales

→ Hace clic en el link
↓
PASO 12: Primer Acceso
→ Se abre nueva pestaña
→ URL: http://tuempresa.com:8071
→ Ve pantalla de login de Odoo
→ Ingresa: admin / abc123XYZ
↓
PASO 13: Configuración Inicial
→ Odoo le pregunta idioma, zona horaria
→ Selecciona módulos a instalar
→ Empieza a usar su sistema
↓
FIN: Cliente tiene su Odoo funcionando

TIEMPO TOTAL: 5-10 minutos desde el pago

5.2 Flujo Secundario: Renovación de Suscripción

┌───┐
│ ESCENARIO: Cliente con suscripción por vencer │
└───┘

PASO 1: Recordatorio Automático (7 días antes)
→ Cron ejecuta a las 00:00
→ Busca instancias que expiran en 7 días
→ Envía email recordatorio
→ Email contiene link para renovar
↓
PASO 2: Cliente Recibe Recordatorio
┌─ ¿Qué hace el cliente?
│ ├─ Ignora el email → Ir a FLUJO DE SUSPENSIÓN
│ └─ Hace clic en "Renovar" → Continuar PASO 3
↓
PASO 3: Proceso de Pago
→ Cliente entra al checkout
→ Sistema detecta que ya existe instancia
→ No crea nueva, solo procesa pago
↓
PASO 4: Confirmación de Pago
→ Pago procesado exitosamente
→ Trigger automático detecta la renovación
↓
PASO 5: Actualización de Fecha
→ Sistema busca registro de instancia
→ Actualiza: fecha_expiracion = hoy + 30 días

→ Actualiza: ultimo_pago = hoy
→ Estado permanece: activo
↓
PASO 6: Notificación
→ Email: "Tu suscripción ha sido renovada"
→ Email: "Próximo cobro: [FECHA]"
↓
FIN: Suscripción extendida por 30 días más

5.3 Flujo de Suspensión

```
┌──────────────────────────────────────────────────────────────────────┐
│ ESCENARIO: Cliente no renovó a tiempo                                │
└──────────────────────────────────────────────────────────────────────┘

PASO 1: Llegada de Fecha de Expiración
   → Cron ejecuta a las 00:00
   → Fecha actual: 2026-03-05
   → Fecha expiración de Juan: 2026-03-04
   → Sistema detecta: expiracion < hoy
   ↓
PASO 2: Validación
   → ¿Estado actual? = activo
   → ¿Hay pago pendiente procesándose? = no
   → Proceder a suspensión
   ↓
PASO 3: Ejecución de Comandos Docker
   3.1: docker stop odoo_cliente_juan
   3.2: docker stop odoo_cliente_juan_db
   → Contenedores se detienen pero NO se borran
   → Los datos permanecen intactos en disco
   ↓
PASO 4: Actualización de Base de Datos
   → UPDATE saas_instance
     SET estado = 'suspendido',
         fecha_suspension = '2026-03-05'
     WHERE id = [ID_JUAN]
   ↓
PASO 5: Notificación
   → Enviar email de suspensión
   → Incluir link directo para renovar
   ↓
PASO 6: Experiencia del Cliente
   → Juan intenta acceder a http://tuempresa.com:8071
   → Ve error: "No se puede conectar"
   → O página personalizada: "Servicio Suspendido"
```

↓
ESTADO: Instancia suspendida, esperando pago

5.4 Flujo de Reactivación

```
┌──────────────────────────────────────────────────────────────┐
│ ESCENARIO: Cliente paga después de suspensión                │
└──────────────────────────────────────────────────────────────┘

PASO 1: Cliente Decide Pagar
   → Recibe email de suspensión
   → Hace clic en "Renovar Ahora"
   → O entra a su portal y ve botón "Reactivar"
   ↓
PASO 2: Proceso de Pago
   → Ingresa a checkout
   → Paga la suscripción
   → Pago se confirma
   ↓
PASO 3: Trigger de Reactivación
   → Sistema detecta pago
   → Busca instancia del cliente
   → Encuentra estado = 'suspendido'
   ↓
PASO 4: Ejecución de Comandos Docker
   4.1: docker start odoo_cliente_juan_db
        → Esperar 3 segundos
   4.2: docker start odoo_cliente_juan
        → Esperar 10 segundos
   ↓
PASO 5: Actualización de Registro
   → UPDATE saas_instance
     SET estado = 'activo',
         fecha_expiracion = hoy + 30,
         ultimo_pago = hoy
     WHERE id = [ID_JUAN]
   ↓
PASO 6: Notificación
   → Email: "¡Tu servicio ha sido reactivado!"
   → Email: "Ya puedes acceder a tu Odoo"
   ↓
PASO 7: Cliente Accede
   → Entra a http://tuempresa.com:8071
   → Hace login con sus mismas credenciales
   → Ve TODOS sus datos tal como los dejó
   → Continúa trabajando normalmente
```

↓
FIN: Servicio restaurado, datos intactos

---

6. STACK TECNOLÓGICO

6.1 Capa de Infraestructura

Componente	Tecnología	Versión	Justificación
Sistema Operativo	Ubuntu Server	22.04 LTS	• Estándar de la industria • Soporte hasta 2027 • Compatibilidad total con Docker • Amplia documentación
Contenedorización	Docker Engine	24.0+	• Aislamiento de instancias • Gestión eficiente de recursos • Portabilidad • Escalabilidad horizontal
Orquestación Básica	Docker Compose	2.20+	• Definición declarativa de servicios • Gestión de redes • Control de volúmenes • Fácil replicación

¿Por qué Ubuntu y no Windows?

- Docker funciona nativamente en Linux
- Windows Server tiene costos de licencia
- Ubuntu es más ligero (menos RAM/CPU)
- Comandos estándar en servidores cloud

¿Por qué Docker y no máquinas virtuales?

- VMs: Cada una necesita SO completo (pesado)
- Docker: Comparten el kernel (ligero)
- Ejemplo:
    - 10 VMs = 10 GB RAM mínimo
    - 10 Contenedores = 2-3 GB RAM total

6.2 Capa de Aplicación

Componente	Tecnología	Versión	Justificación

ERP/CMS	Odoo Community	17.0	• Gratis y open-source • eCommerce integrado • Sistema de portal incluido • Framework de desarrollo robusto
Base de Datos	PostgreSQL	15+	• Único compatible con Odoo • ACID compliant • Excelente rendimiento • Gratis
Lenguaje Backend	Python	3.10+	• Lenguaje nativo de Odoo • Amplia librería estándar • Fácil integración con Docker
Framework Web	Odoo Web Framework	Incluido	• Ya integrado en Odoo • No necesita React/Vue/Angular • Editor visual incluido

¿Por qué Odoo Community y no Enterprise?

- Community es gratis
- Tiene todas las funcionalidades base necesarias
- eCommerce está incluido
- Podemos programar lo que falta

¿Por qué NO usar Odoo.sh (SaaS oficial)?

- Cuesta $24/usuario/mes
- No permite controlar la infraestructura
- No aprenderíamos sobre DevOps
- Nuestro proyecto enseña más

6.3 Capa de Desarrollo

Componente	Tecnología	Propósito
IDE/Editor	VS Code	Desarrollo de módulos
Control de Versiones	Git + GitHub	Historial y colaboración
SSH Client	OpenSSH	Acceso al servidor
Gestión de Dependencias	pip (Python)	Librerías adicionales

6.4 Capa de Servicios Externos

Servicio	Proveedor	Costo	Propósito

Pasarela de Pago	Stripe	2.9% + $0.30 por transacción	Procesar pagos con tarjeta
Email	Gmail SMTP (dev) SendGrid (prod)	Gratis hasta 100/día $15/mes (40k emails)	Envío de notificaciones
Dominio	Namecheap	$10-15/año	www.tuempresa.com
Servidor (VPS)	DigitalOcean	$6-12/mes	Hosting del sistema
SSL/TLS	Let's Encrypt	Gratis	Certificado HTTPS

6.5 Capa de Monitoreo (Opcional - Fase 2)

Componente	Tecnología	Propósito
Logs	Journalctl (sistema) Odoo logs (aplicación)	Debugging y auditoría
Uptime	UptimeRobot	Alertas si el servicio cae
Backups	Script cron + rsync	Respaldo diario de bases de datos

---

7. PRODUCT BACKLOG

7.1 Estructura del Backlog

Cada ítem tiene:

- **ID:** Identificador único
- **Tipo:** Epic / User Story / Task / Spike (investigación)
- **Prioridad:** Critical / High / Medium / Low
- **Estimación:** Puntos de historia (1, 2, 3, 5, 8, 13)
- **Dependencias:** De qué otros ítems depende

7.2 Épicas del Proyecto

ÉPICA 1: Infraestructura Base
→ Configurar servidor y entorno de desarrollo

ÉPICA 2: Portal Web y eCommerce
→ Crear la tienda online funcional

ÉPICA 3: Sistema de Provisión Automática
→ Automatizar creación de instancias

ÉPICA 4: Gestión de Ciclo de Vida
→ Automatizar suspensión/reactivación

ÉPICA 5: Portal del Cliente
→ Interfaz para gestión de suscripción

ÉPICA 6: Despliegue a Producción
→ Migrar a servidor real en la nube

7.3 Backlog Detallado

ÉPICA 1: Infraestructura Base

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E1-001	**[SPIKE]** Investigar Docker y contenedores	Critical	3	1
E1-002	Instalar Ubuntu Server en VirtualBox	Critical	2	1
E1-003	Configurar red bridged para acceso desde host	High	2	1
E1-004	Instalar Docker Engine en Ubuntu	Critical	2	1
E1-005	Instalar Docker Compose	Critical	1	1
E1-006	Crear estructura de proyecto	High	1	1
E1-007	Configurar docker-compose.yml para Odoo Maestro	Critical	3	1
E1-008	Levantar Odoo Maestro + PostgreSQL	Critical	2	1
E1-009	Verificar acceso desde navegador Windows	Critical	1	1
E1-010	Configurar volumen para addons personalizados	High	2	1

ÉPICA 2: Portal Web y eCommerce

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint

E2-001	Como visitante, quiero ver una landing page atractiva que explique el servicio	High	5	2
E2-002	Como visitante, quiero ver los planes disponibles con sus precios	Critical	3	2
E2-003	Instalar módulo Website en Odoo	Critical	1	2
E2-004	Instalar módulo eCommerce en Odoo	Critical	1	2
E2-005	Diseñar página de inicio con Website Builder	High	5	2
E2-006	Crear página "Cómo Funciona"	Medium	3	2
E2-007	Crear página "Precios"	High	3	2
E2-008	Configurar productos: Plan Básico, Plan Pro	Critical	3	2
E2-009	Configurar precios y descripciones	High	2	2
E2-010	Personalizar tema visual (colores, logo)	Medium	3	2
E2-011	Como cliente, quiero poder registrarme en el sitio	Critical	2	2
E2-012	Como cliente, quiero poder hacer login	Critical	1	2
E2-013	Configurar formulario de registro	High	2	2
E2-014	Probar flujo completo de registro	High	1	2

ÉPICA 3: Sistema de Provisión Automática

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E3-001	**[SPIKE]** Investigar Odoo ORM y herencia de modelos	Critical	5	3

E3-002	**[SPIKE]** Investigar subprocess y comandos Docker desde Python	Critical	5	3
E3-003	Crear estructura del módulo saas_docker_manager	Critical	2	3
E3-004	Configurar **manifest**.py con dependencias	Critical	1	3
E3-005	Crear modelo saas.instance para control de instancias	Critical	5	3
E3-006	Definir campos del modelo (nombre, puerto, estado, etc.)	Critical	3	3
E3-007	Crear vistas XML para gestión de instancias	High	5	3
E3-008	Implementar método para calcular siguiente puerto disponible	High	3	4
E3-009	Heredar modelo sale.order para detectar ventas	Critical	5	4
E3-010	Sobrescribir método action_confirm()	Critical	3	4
E3-011	Implementar validación de producto (es suscripción Odoo?)	High	2	4
E3-012	Crear función para generar contraseña aleatoria	Medium	2	4
E3-013	Implementar creación de registro en saas.instance	Critical	3	4
E3-014	Implementar comando Docker para crear PostgreSQL del cliente	Critical	8	5
E3-015	Implementar comando Docker para crear Odoo del cliente	Critical	8	5
E3-016	Implementar gestión de red Docker	High	5	5
E3-017	Implementar manejo de errores en creación	High	5	5
E3-018	Probar creación manual de instancia	Critical	3	5

E3-019	Probar creación automática tras venta	Critical	5	5

ÉPICA 4: Gestión de Ciclo de Vida

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E4-001	**[SPIKE]** Investigar Cron Jobs en Odoo	Critical	3	6
E4-002	Como sistema, quiero verificar diariamente las suscripciones expiradas	Critical	5	6
E4-003	Crear Scheduled Action para verificación diaria	Critical	3	6
E4-004	Implementar método _cron_check_expired_subscriptions()	Critical	5	6
E4-005	Implementar lógica de búsqueda de instancias expiradas	High	3	6
E4-006	Implementar función action_stop_container()	Critical	5	6
E4-007	Implementar comando docker stop	Critical	3	6
E4-008	Actualizar estado en BD a 'suspendido'	High	2	6
E4-009	Implementar función action_start_container()	Critical	5	7
E4-010	Implementar comando docker start	Critical	3	7
E4-011	Actualizar estado en BD a 'activo'	High	2	7
E4-012	Implementar trigger de reactivación al pagar	Critical	5	7
E4-013	Detectar pago de renovación	High	3	7
E4-014	Actualizar fecha de expiración	High	2	7

E4-015	Probar ciclo completo: expiración → suspensión → pago → reactivación	Critical	5	7

ÉPICA 5: Portal del Cliente

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E5-001	Como cliente, quiero ver mis instancias en "Mi Cuenta"	High	5	8
E5-002	Como cliente, quiero acceder a mi Odoo con un botón	Critical	3	8
E5-003	Como cliente, quiero ver el estado de mi suscripción	High	3	8
E5-004	Crear template XML para portal	High	5	8
E5-005	Extender portal_my_home	High	3	8
E5-006	Crear controlador para /my/instances	High	5	8
E5-007	Diseñar tarjetas de instancia	Medium	3	8
E5-008	Implementar botón "Acceder a mi Odoo"	High	2	8
E5-009	Mostrar fecha de expiración	High	2	8
E5-010	Mostrar estado visual (activo=verde, suspendido=rojo)	Medium	2	8
E5-011	Implementar mensaje si no hay instancias	Low	1	8

ÉPICA 6: Sistema de Notificaciones

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E6-001	Como cliente, quiero recibir mis credenciales por email al crear instancia	Critical	5	9
E6-002	Como cliente, quiero recibir recordatorio antes de expiración	High	3	9

E6-003	Como cliente, quiero recibir notificación de suspensión	High	3	9
E6-004	Configurar servidor SMTP en Odoo	Critical	2	9
E6-005	Crear plantilla de email de bienvenida	High	3	9
E6-006	Crear plantilla de email de recordatorio	High	3	9
E6-007	Crear plantilla de email de suspensión	High	3	9
E6-008	Crear plantilla de email de reactivación	Medium	2	9
E6-009	Implementar envío automático tras crear instancia	Critical	3	9
E6-010	Implementar envío en cron de recordatorio	High	3	9
E6-011	Probar recepción de todos los emails	High	2	9

ÉPICA 7: Integración de Pagos

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E7-001	**[SPIKE]** Investigar integración de Stripe con Odoo	Critical	5	10
E7-002	Como cliente, quiero pagar con tarjeta de forma segura	Critical	8	10
E7-003	Crear cuenta de Stripe (modo test)	Critical	1	10
E7-004	Instalar Payment Provider de Stripe en Odoo	Critical	2	10
E7-005	Configurar claves API de Stripe	Critical	2	10
E7-006	Configurar Webhook de Stripe	Critical	5	10

E7-007	Probar pago de prueba	Critical	3	10
E7-008	Verificar que webhook activa creación de instancia	Critical	3	10
E7-009	Documentar proceso de configuración	Medium	2	10

ÉPICA 8: Despliegue a Producción

ID	Historia de Usuario / Tarea	Prioridad	Estimación	Sprint
E8-001	**[SPIKE]** Investigar proveedores VPS	High	3	11
E8-002	Contratar VPS (DigitalOcean/Linode)	Critical	1	11
E8-003	Configurar Ubuntu en VPS	Critical	3	11
E8-004	Configurar firewall (UFW)	High	2	11
E8-005	Instalar Docker en VPS	Critical	2	11
E8-006	Transferir código con Git	High	2	11
E8-007	Levantar servicios en VPS	Critical	3	11
E8-008	Comprar dominio	High	1	12
E8-009	Configurar DNS	High	2	12
E8-010	Instalar Nginx	High	3	12
E8-011	Configurar Nginx como reverse proxy	High	5	12
E8-012	Instalar certificado SSL (Let's Encrypt)	High	3	12
E8-013	Configurar Stripe en modo producción	Critical	2	12
E8-014	Configurar email SMTP producción	High	2	12
E8-015	Pruebas end-to-end en producción	Critical	5	12

---

8. PLANIFICACIÓN DE SPRINTS

8.1 Metodología Scrum Adaptada

Duración de Sprint: 1 semana
Equipo: 2 desarrolladores
Ceremonias:

- Daily Standup: 15 min diarios (async vía chat si es necesario)
- Sprint Planning: Lunes 30 min
- Sprint Review: Viernes 30 min
- Sprint Retrospective: Viernes 20 min

Definition of Done:

- Código escrito y probado
- Documentación actualizada
- Sin errores críticos
- Merge a rama main
- Demo funcional

8.2 Sprints Detallados

SPRINT 1: Fundamentos (Semana 1)

Objetivo: Tener entorno de desarrollo funcionando

Backlog Items:

- E1-001 a E1-010 (todos los ítems de infraestructura)

Entregables:

- Ubuntu con Docker funcionando
- Odoo Maestro accesible desde navegador
- Carpeta de addons configurada

Criterios de Aceptación:

- ✅ Puedo entrar a http://IP_UBUNTU:8069 desde Windows
- ✅ Puedo crear una base de datos
- ✅ Puedo activar modo desarrollador
- ✅ Puedo ver la carpeta de addons vacía

SPRINT 2: Portal Web (Semana 2)

Objetivo: Tener tienda online funcional

Backlog Items:

- E2-001 a E2-014

Entregables:

- Website con página de inicio
- Página de precios
- eCommerce configurado
- Productos creados

Criterios de Aceptación:

- ✅ Un visitante puede ver los planes
- ✅ Puede registrarse
- ✅ Puede agregar un plan al carrito
- ✅ Puede llegar hasta el checkout (sin pagar aún)

SPRINT 3: Fundamentos del Módulo (Semana 3)

Objetivo: Crear módulo base y modelo de instancias

Backlog Items:

- E3-001 a E3-007

Entregables:

- Módulo saas_docker_manager instalable
- Modelo saas.instance creado
- Vista para ver instancias

Criterios de Aceptación:

- ✅ El módulo aparece en Apps
- ✅ Puedo instalarlo sin errores
- ✅ Puedo crear un registro de instancia manualmente
- ✅ Puedo ver la lista de instancias en el menú

SPRINT 4: Captura de Ventas (Semana 4)

Objetivo: Detectar cuando se vende una suscripción

Backlog Items:

- E3-008 a E3-013

Entregables:

- Herencia de sale.order
- Validación de producto

- Creación de registro automático

Criterios de Aceptación:

- ✅ Cuando confirmo una venta manual, se crea un registro en saas.instance
- ✅ El puerto es único
- ✅ La contraseña es aleatoria
- ✅ La fecha de expiración es +30 días

SPRINT 5: Creación de Instancias (Semana 5)

Objetivo: Crear contenedores Docker automáticamente

Backlog Items:

- E3-014 a E3-019

Entregables:

- Función que ejecuta docker run
- Gestión de red Docker
- Creación completa funcional

Criterios de Aceptación:

- ✅ Hago una venta de prueba
- ✅ Automáticamente se crean 2 contenedores (PostgreSQL + Odoo)
- ✅ Puedo acceder a http://IP:PUERTO_ASIGNADO
- ✅ Veo pantalla de Odoo del cliente
- ✅ Los datos están aislados del maestro

SPRINT 6: Suspensión Automática (Semana 6)

Objetivo: Implementar corte de servicio

Backlog Items:

- E4-001 a E4-008

Entregables:

- Cron Job configurado
- Función de suspensión
- Actualización de estado

Criterios de Aceptación:

- ✅ Cambio manualmente una fecha de expiración al pasado
- ✅ Ejecuto el cron manualmente
- ✅ La instancia se suspende
- ✅ El contenedor está stopped
- ✅ No puedo acceder al Odoo del cliente

SPRINT 7: Reactivación (Semana 7)

Objetivo: Permitir reactivación al pagar

Backlog Items:

- E4-009 a E4-015

Entregables:

- Función de reactivación
- Trigger de renovación
- Ciclo completo funcional

Criterios de Aceptación:

- ✅ Instancia está suspendida
- ✅ Hago una "renovación" (venta del mismo producto)
- ✅ La instancia se reactiva automáticamente
- ✅ Puedo acceder de nuevo
- ✅ Los datos están intactos

SPRINT 8: Portal del Cliente (Semana 8)

Objetivo: Interfaz para que el cliente vea su instancia

Backlog Items:

- E5-001 a E5-011

Entregables:

- Sección "Mis Instancias" en portal
- Botón de acceso
- Información de estado

Criterios de Aceptación:

- ✅ Hago login como cliente
- ✅ Veo menú "Mi Cuenta"
- ✅ Dentro veo "Mis Instancias"

- ✅ Veo mi instancia con estado
- ✅ Hay un botón que me lleva a mi Odoo

SPRINT 9: Notificaciones (Semana 9)

Objetivo: Emails automáticos funcionando

Backlog Items:

- E6-001 a E6-011

Entregables:

- SMTP configurado
- Plantillas de email
- Envíos automáticos

Criterios de Aceptación:

- ✅ Al crear instancia, recibo email con credenciales
- ✅ 7 días antes de expirar, recibo recordatorio
- ✅ Al suspenderse, recibo notificación
- ✅ Al reactivar, recibo confirmación

SPRINT 10: Pagos Reales (Semana 10)

Objetivo: Integrar Stripe funcionando

Backlog Items:

- E7-001 a E7-009

Entregables:

- Stripe configurado
- Webhook funcionando
- Pagos de prueba exitosos

Criterios de Aceptación:

- ✅ Hago una compra con tarjeta de prueba
- ✅ El pago se procesa
- ✅ Se crea la instancia automáticamente
- ✅ Recibo el email

SPRINT 11: Infraestructura Cloud (Semana 11)

Objetivo: Sistema funcionando en servidor real

Backlog Items:

- E8-001 a E8-007

Entregables:

- VPS contratado y configurado
- Código desplegado
- Servicios corriendo

Criterios de Aceptación:

- ✅ Puedo SSH al servidor
- ✅ Docker está corriendo
- ✅ Odoo Maestro está accesible por IP pública
- ✅ Puedo hacer una compra de prueba

SPRINT 12: Producción Final (Semana 12)

Objetivo: Sistema público con dominio y HTTPS

Backlog Items:

- E8-008 a E8-015

Entregables:

- Dominio configurado
- HTTPS funcionando
- Stripe en modo producción
- Sistema completamente funcional

Criterios de Aceptación:

- ✅ Entro a www.tuempresa.com (sin HTTP, sin puerto)
- ✅ Veo el candado verde (HTTPS)
- ✅ Hago una compra real
- ✅ Mi instancia se crea
- ✅ Todo funciona end-to-end

---

9. ESTRATEGIA DE IMPLEMENTACIÓN

9.1 Fases de Desarrollo

FASE 1: Prototipo Local (Sprints 1-5)

- Trabajo en VirtualBox
- Foco en lógica del negocio
- Pruebas con datos ficticios
- No gastar en cloud aún

FASE 2: Funcionalidades Completas (Sprints 6-10)

- Agregar automatizaciones
- Integrar pagos (modo test)
- Pulir experiencia de usuario
- Aún en local

FASE 3: Despliegue (Sprints 11-12)

- Migrar a VPS
- Configurar producción
- Pruebas finales
- Go-live

9.2 Estrategia de Pruebas

Pruebas Unitarias:

- Cada función crítica debe tener prueba
- Especialmente: cálculo de puertos, generación de contraseñas

Pruebas de Integración:

- Flujo completo de compra
- Flujo de suspensión/reactivación

Pruebas de Usuario:

- Una persona externa intenta comprar
- Debe poder hacerlo sin ayuda

9.3 Estrategia de Git

Ramas:

```
main (producción)
  ├── develop (integración)
     ├── feature/ecommerce
     ├── feature/provisioning
     ├── feature/lifecycle
     └── feature/portal
```

Commits:

- Mensajes claros: "Implementa creación de instancia #E3-014"
- Commits pequeños y frecuentes

Pull Requests:

- Todo merge a develop requiere review
- Demo antes de merge a main

---

10. PLAN DE DESPLIEGUE

10.1 Ambientes

Ambiente	Ubicación	Propósito	Datos
Desarrollo	VirtualBox Local	Programar y probar	Ficticios
Staging	VPS (subdominio test.)	Pruebas pre-producción	Ficticios + algunos reales
Producción	VPS (dominio principal)	Sistema real de clientes	Reales

10.2 Checklist Pre-Producción

Seguridad:

- Firewall configurado (solo puertos 22, 80, 443 abiertos)
- Contraseñas seguras en BD
- SSH solo con clave, no password
- Odoo con usuario admin fuerte
- Variables de entorno para secrets

Rendimiento:

- Al menos 2GB RAM en VPS
- Docker con límites de recursos por contenedor
- PostgreSQL con configuración optimizada

Backups:

- Script de backup diario de bases de datos
- Almacenamiento externo (S3 o similar)
- Prueba de restauración exitosa

Monitoreo:

- UptimeRobot configurado
- Logs centralizados
- Alertas de email si el sistema cae

10.3 Plan de Rollback

Si algo falla en producción:

1. Mantener versión anterior en rama `main-backup`
2. Script de rollback: `git checkout main-backup && docker-compose up -d`
3. Notificar clientes si es necesario
4. Investigar el problema en ambiente staging
5. Re-deploy cuando esté solucionado

---

11. GESTIÓN DE RIESGOS

11.1 Matriz de Riesgos

Riesgo	Probabilidad	Impacto	Mitigación
Contenedores no se crean correctamente	Alta	Crítico	• Logs detallados • Retry automático • Notificación al admin
Conflicto de puertos	Media	Alto	• Algoritmo de asignación robusto • Validación antes de crear
Cliente no recibe email	Media	Alto	• Usar servicio confiable (SendGrid) • Mostrar credenciales también en portal
Fallo en pasarela de pago	Media	Crítico	• Manejo de errores graceful • Retry del webhook
Servidor se queda sin recursos	Alta	Crítico	• Límites por contenedor • Monitoreo de RAM/CPU • Auto-scaling (fase 2)
Pérdida de datos por fallo de disco	Baja	Crítico	• Backups diarios • Almacenamiento redundante
Ataque de seguridad	Media	Crítico	• Firewall • Actualizaciones de seguridad

			• Certificado SSL
Cliente malintencionado satura recursos	Media	Alto	• Límites de CPU/RAM • Rate limiting • Monitoreo de uso anormal

11.2 Plan de Contingencia

Si el servidor cae:

1. Recibir alerta de UptimeRobot
2. SSH al servidor
3. Verificar logs: `docker logs odoo_maestro`
4. Reiniciar servicios: `docker-compose restart`
5. Si persiste: restaurar desde backup

Si un contenedor de cliente falla:

1. Detectar en cron de monitoreo
2. Intentar restart: `docker start [container]`
3. Si falla: crear nuevo contenedor y migrar BD
4. Notificar al cliente del incidente

---

12. MÉTRICAS DE ÉXITO

12.1 KPIs Técnicos

Métrica	Objetivo	Cómo Medirlo
Tiempo de provisión	< 30 segundos	Log del timestamp inicio vs fin
Uptime	> 99.5%	UptimeRobot
Tasa de error en creación	< 1%	Instancias creadas vs errores
Tiempo de respuesta web	< 2 segundos	Google PageSpeed

12.2 KPIs de Negocio

Métrica	Objetivo	Cómo Medirlo
Conversión visita → compra	> 2%	Analytics

Tasa de renovación	> 80%	Renovaciones / Total clientes
Churn (abandono) mensual	< 5%	Clientes perdidos / Total
Ingreso recurrente mensual (MRR)	Crecient e	Total de suscripciones activas × precio

12.3 Criterios de Éxito del Proyecto

MVP Exitoso si:

- ✅ Un usuario externo puede comprar sin ayuda
- ✅ Recibe su instancia en < 1 minuto
- ✅ Puede usar Odoo normalmente
- ✅ La suspensión funciona automáticamente
- ✅ La reactivación funciona al pagar
- ✅ El sistema corre 7 días sin intervención manual

CONCLUSIÓN

Este documento es la **hoja de ruta completa** para construir el portal SaaS de Odoo. No contiene código, pero explica **qué hacer, por qué hacerlo, y en qué orden**.

Próximos Pasos Inmediatos:

1. **Estudiar este documento completo** (2-3 días)
2. **Investigar conceptos clave** (Docker, Odoo ORM, herencia) (1 semana)
3. **Iniciar Sprint 1** (configurar entorno)
4. **Daily standups** con tu compañero Marco
5. **Adaptacion de plan** según lo que aprendan

Filosofía de Trabajo:

- **Iterativo:** No todo saldrá bien a la primera, y está bien
- **Incremental:** Cada sprint agrega valor
- **Comunicación:** Hablar diario con Marco sobre bloqueos
- **Documentación:** Cada decisión importante, documentarla

Recuerda:

> "El mejor código es el que funciona. El segundo mejor es el que se entiende."

No busques la perfección, busca que funcione. Luego puedes mejorarlo.

Documento preparado para: Pasantía de Desarrollo de Software
Fecha: Febrero 2026
Versión: 1.0 - Plan Estratégico