# UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

## AGRICULTURE COMPANY
## Software Architecture Document (SAD)

**PROPIETARIO(S) DEL CONTENIDO: Diego Martínez Lora, Luis Carlos Pacheco y Nicolás Molina Díaz**

| NÚMERO DE DOCUMENTO: | VERSIÓN/REVISIÓN: | LANZAMIENTO/FECHAREVISIÓN: |
|---|---|---|
| • 1 | • 1.0 | • 26/11/2024 |
| • | • | • |
| • | • | • |
| • | • | • |
| • | • | • |
| • | • | • |

# Tabla de Contenidos

**1 Tabla de Contenidos**

1. **Documentation Roadmap ..... 1**
   - **1.1. Propósito de la Revisión .....1**
   - **1.2. Objetivo y Propósito del SAD ..... 2**
   - **1.3. ¿Qué es la arquitectura de software? .....2**
   - **1.4. Cómo está organizado el SAD ..... 3**
2. **Representación de los Stakeholders ..... 5**
   - **2.1. Stakeholders y puntos de relevancia ..... 5**
   - **2.2. Stakeholders y preocupaciones abordadas .....6**
3. **Trasfondo de la Arquitectura .....13**
   - **3.1. Trasfondo del problema ..... 13**
     - **3.1.1. Vista general del sistema ..... 14**
     - **3.1.2. Metas y contexto ..... 14**
     - **3.1.3. Requisitos clave ..... 15**
   - **3.2. Trasfondo de la solución .....15**
     - **3.2.1. Enfoques arquitectónicos ..... 16**
     - **3.2.2. Resultados de análisis ..... 16**
     - **3.2.3. Cobertura de requerimientos ..... 17**
     - **5.2.4. Resumen de cambios en la versión actual ..... 18**
   - **5.3. Consideración de reutilización en línea de productos .....19**
4. **Vistas ..... 21**
   - **4.1. Vista de modelado de datos ..... 21**
     - **4.1.1. Catálogo de elementos ..... 22**
     - **4.1.2. Relaciones clave ..... 22**
   - **4.2. Vista de interacción del sistema .....23**
     - **4.2.1. Componentes y conectores ..... 21**
     - **4.2.2. Mecanismos de variabilidad ..... 21**
   - **4.3. Vista de asignación de hardware .....21**
5. **Definiciones de puntos de vista ..... 6**
   - **3.1. Elementos, relaciones, propiedades y restric-ciones.....7**

**3.2. Lenguajes para modelar ..... 9**
**3.3. Técnicas de evaluación y criterios de consistencia/completitud .... 7**
**6. Materiales de referencia**
**7. Directorio ..... 6**
**3.1. Elementos, relaciones, propiedades y restric-ciones.....7**
**3.2. Lenguajes para modelar ..... 9**
**3.3. Técnicas de evaluación y criterios de consistencia/completitud .... 7**
**8. Figuras y tablas ..... 21**
**4.1. Vista de modelado de datos ..... 21**
**4.1.1. Catálogo de elementos ..... 22**
**4.1.2. Relaciones clave ..... 22**
**4.2. Vista de interacción del sistema .....23**
**4.2.1. Componentes y conectores ..... 21**
**4.2.2. Mecanismos de variabilidad ..... 21**
**4.3. Vista de asignación de hardware .....21**
**3. Trasfondo de la Arquitectura .....13**
**5.1. Trasfondo del problema ..... 13**
**5.1.1. Vista general del sistema ..... 14**
**5.1.2. Metas y contexto ..... 14**
**5.1.3. Requisitos clave ..... 15**
**5.2. Trasfondo de la solución .....15**
**5.2.1. Enfoques arquitectónicos ..... 16**
**5.2.2. Resultados de análisis ..... 16**
**5.2.3. Cobertura de requerimientos ..... 17**
**5.2.4. Resumen de cambios en la versión actual ..... 18**
**5.3. Consideración de reutilización en línea de productos .....19**

**9. Relaciones entre vistas ..... 23**
**6.1. Relaciones generales entre vistas .....23**
**6.2. Relaciones específicas entre vistas ..... 24**
**10. Glosario y acrónimos ..... 25**
**7.1. Glosario ..... 25**
**7.2. Lista de acrónimos ..... 26**

**11. Figuras y tablas ..... 27**
**8.1. Lista de figuras ..... 27**
**Figura 1: Diagrama Entidad-Relación sobre el proyecto ..... 31**
**Figura 2: Diagrama de Proceso de Negocio ..... 32**
**8.2. Lista de tablas ..... 27**
**Tabla 1: Representación de stakeholders ..... 5**
**Tabla 2: Stakeholders y puntos de relevancia ..... 6**
**Tabla 3: Vistas representadas en el SAD ..... 23**
**Tabla 4: Relaciones específicas entre vistas ..... 24**
**Tabla 5: Glosario ..... 29**
**Tabla 6: Lista de acrónimos ..... 30**

# Lista de Figuras

Figura 1: Diagrama Entidad-Relación sobre el proyecto ……..31

Figura 2: Diagrama de Proceso de Negocio ……..32

Mapa de la documentación

## 1.1 Manejo del Documento y Configuración del Control en Formación

- Número de Revisión: 1.0
- Propósito de la Revisión: Proveer un documento inicial que describa la arquitectura de software para el sistema Agriculture Company.
- Rango de la revision: página 1 -

## 1.2 Objetivo y Propósito del SAD

Este documento SAD especifica la arquitectura de software para < **la arquitectura de software del sistema de la Compañía Agrícola, diseñada para optimizar los procesos agrícolas como la gestión de cultivos, el inventario y la distribución.** >. Toda la información sobre la arquitectura de software se puede encontrar en este documento, aunque mucha información se incorpora por referencia a otros documentos.

### 1.2.1 ¿Qué es la arquitectura de software?

La arquitectura de software para el sistema de la Compañía Agrícola define la estructura y las relaciones entre sus componentes.

Incluye:

- **Elementos de software:** Módulos como Gestión de Cultivos, Inventario, Distribución y Ventas.
- **Propiedades visibles externamente:** Los servicios que brindan estos elementos, incluidas las características de rendimiento, la tolerancia a fallas y el uso de recursos.
- **Relaciones:** Las interacciones entre módulos (por ejemplo, actualización del inventario en función de los datos de cosecha del módulo Gestión de Cultivos).

Esta arquitectura garantiza la **modularidad**, la **escalabilidad** y la **flexibilidad**, y admite tanto las funcionalidades actuales como las expansiones futuras.

## 1.2.2 Elementos y relaciones

La arquitectura prioriza las relaciones entre los componentes sobre sus detalles de implementación interna. Las consideraciones clave incluyen:

- **Interfaces:** Las interfaces públicas definen cómo interactúan los módulos entre sí. Por ejemplo, la API entre los módulos de Gestión de Cultivos e Inventario facilita el intercambio de datos sobre el estado de los cultivos y los rendimientos esperados.
- **Comportamiento:** El comportamiento de cada módulo está documentado para garantizar interacciones predecibles, como la forma en que el módulo de Distribución solicita datos de inventario para la planificación de rutas.
- **Abstracción:** Los detalles de implementación interna se omiten a menos que influyan en las interacciones entre los componentes.

## 1.2.3 Estructuras múltiples

La arquitectura del sistema está compuesta por múltiples estructuras para abordar diferentes perspectivas:

### 1.2.3.1 Estructura del módulo

- A cada módulo (Gestión de Cultivos, Inventario) se le asignan responsabilidades claras para garantizar la separación funcional.
- Los módulos están diseñados para su reutilización en otros sistemas de gestión agrícola.

### 1.2.3.2 Estructura de componentes y conectores

- Captura interacciones en tiempo de ejecución, como el flujo de datos entre la base de datos y los módulos.
- Por ejemplo, el módulo de Gestión de Cultivos envía datos sobre los cronogramas de siembra al módulo de Inventario en tiempo real.

### 1.2.3.3 Estructura de asignación

- Asigna módulos y componentes a recursos físicos, como contenedores Docker y bases de datos basadas en la nube.
- Por ejemplo, el módulo Inventario está en contenedores para permitir la escalabilidad durante las temporadas pico de cosecha.

Estas estructuras permiten que el equipo aborde problemas funcionales, de rendimiento y de implementación de forma independiente, manteniendo al mismo tiempo la coherencia general.

### 1.2.4 Comportamiento

La arquitectura del sistema define cómo interactúan y se comportan sus componentes en diversas condiciones:

- **Procesamiento de datos en tiempo real:** La arquitectura admite actualizaciones en tiempo real, como ajustes inmediatos a los niveles de inventario en función de los datos de cosecha de cultivos.
- **Manejo de fallas:** Por ejemplo, si falla una conexión a la base de datos, el sistema proporciona mecanismos de reintento para garantizar la coherencia de los datos.
- **Escalabilidad:** Durante períodos de alta demanda, como las temporadas de cosecha, el sistema se escala horizontalmente mediante la implementación de contenedores adicionales para servicios críticos.

## 1.3 Como el SAD esta organizado

El siguiente SAD está organizado de la siguiente forma:

- **Sección 1:** Proporciona una visión general del documento, su propósito y audiencia.
  - **Sección 2:** Explica el trasfondo del sistema, objetivos, restricciones y el razonamiento detrás de la arquitectura.
  - **Sección 3:** Especifica las vistas arquitectónicas, incluidas las perspectivas de módulos, componentes y conectores, y asignaciones.
  - **Sección 4:** Discute las relaciones y consistencias entre las vistas arquitectónicas.
  - **Sección 5:** Lista las referencias utilizadas para desarrollar este SAD.
  - **Sección 6:** Incluye el glosario, acrónimos y un índice de los elementos arquitectónicos.

## 1.4 Representacion de los stakeholders

| | |
|---|---|
| Usuarios finales | Usabilidad, rapidez de respuesta y disponibilidad del sistema. |

| | |
|---|---|
| Administradores | Seguridad, escalabilidad y monitoreo en tiempo real. |
| Desarrolladores | Mantenibilidad, modularidad y claridad en las dependencias. |
| Gerentes de proyecto | Costo, tiempo de desarrollo y cumplimiento de objetivos comerciales. |

Tabla 1: Representación de stakeholders.

## 1.5 Definiciones de puntos de vista

| **Stakeholder** | **Viewpoint(s) that apply to that class of stakeholder's concerns** |
|---|---|
| Usuarios finales | Vista de componentes y conectores |
| Administradores | Vista de descomposicion modular |
| Desarrolladores | Vista de Descomposición Modular, Vista de Asignación. |
| DevOps | Vista de asignacion |

Tabla 2: Stakeholders y puntos de relevancia.

### 1.5.1 Definiciones de puntos de vista

#### 1.5.1.1 Stakeholders y Preocupaciones Abordadas

Los stakeholders y sus preocupaciones principales para este punto de vista incluyen:

- **Gerentes de proyecto:** Necesidad de definir asignaciones de trabajo, formar equipos y planificar cronogramas y presupuestos.
- **Especialistas en COTS:** Identificar componentes reutilizables para integrarlos al sistema.
- **Testers e integradores:** Definir unidades claras para pruebas e integración.

- **Especialistas en gestión de configuración:** Mantener versiones actualizadas y consistentes de los elementos.
- **Ingenieros de compilación:** Garantizar un sistema funcional basado en los módulos definidos.
- **Mantenedores:** Facilitar la actualización y modificación de los elementos.
- **Implementadores:** Crear elementos que cumplan con especificaciones y estándares.
- **Arquitectos de software:** Diseñar módulos suficientemente claros y autónomos para soportar cambios futuros.
- **Clientes:** Garantizar que los cambios futuros puedan implementarse de manera económica, confinando los efectos a un número limitado de módulos.

#### 1.5.1.2 Elementos, Relaciones, Propiedades y Restricciones

- **Elementos:** Los módulos incluyen Cultivo, Inventario, Ventas y Distribución, cada uno con funciones claramente definidas.
- **Relaciones:** Las relaciones entre los módulos se definen mediante interfaces, asegurando comunicación estructurada.
- **Propiedades:**
  - Nombre del módulo.
  - Funcionalidad asignada.
  - Interfaces entre módulos.
- **Restricciones:**
  - Los módulos deben ser independientes.
  - Las dependencias circulares no están permitidas.

#### 1.5.1.3 Lenguaje(s) para Modelar

- Diagramas UML, utilizando subsistemas o clases para representar elementos.
- Representaciones textuales utilizando formato jerárquico para definir relaciones entre módulos.

#### 1.5.1.4 Técnicas de Evaluación y Criterios de Consistencia/Completitud

- **Criterios de consistencia:**
  - No puede haber más de un padre por elemento.
  - Cada módulo debe cubrir funcionalidades únicas, sin superposición.
  - Toda la funcionalidad del sistema debe estar mapeada en los módulos.
  - Cada pieza de código fuente debe asociarse a un módulo definido.
- **Técnicas de evaluación:**
  - Métodos basados en escenarios como ATAM para validar la modularidad frente a cambios.
  - Mapeo disciplinado a requisitos para asegurar cobertura.

#### 1.5.1.5 Fuente del Punto de Vista

El estilo de descomposición modular está basado en [Clements 2002, Sección 2.1], que proporciona directrices para estructurar sistemas en módulos jerárquicos y autónomos.

## 1.6 Como se documenta una vista

#### 1.6.1.1 Descripción de la Vista

Esta vista describe la arquitectura del sistema dividiéndola en módulos jerárquicos y autónomos. Cada módulo tiene responsabilidades específicas que aseguran la modularidad y la facilidad de mantenimiento. Los módulos principales representan funciones clave del sistema **Agriculture Company**, como la gestión de cultivos, inventario, distribución y ventas.

#### 1.6.1.2 Presentación Principal

Los elementos en esta vista están representados mediante un diagrama UML de clases y componentes. El diagrama muestra los módulos como nodos independientes, conectados a través de interfaces bien definidas que facilitan la comunicación y la integración entre ellos.

**Lenguaje de representación:** UML (diagramas de clases y componentes).

### 1.6.1.3 Catálogo de Elementos

##### 1.6.1.3.1 Elementos

- **Cultivo:** Módulo encargado de la planificación y monitoreo de cultivos.
- **Inventario:** Módulo que gestiona el almacenamiento y la disponibilidad de productos.
- **Distribución:** Responsable de la optimización de rutas y planificación de entregas.
- **Ventas:** Gestión de transacciones con puntos de venta.

#### 1.6.1.3.2 Relaciones

- "Cultivo" se conecta con "Inventario" para actualizar datos de productos disponibles.
- "Inventario" interactúa con "Distribución" para planificar entregas según disponibilidad.
- "Distribución" se sincroniza con "Ventas" para garantizar el cumplimiento de pedidos.

#### 1.6.1.3.3 Interfaces

- **Interfaces públicas:** Cada módulo expone sus funcionalidades principales mediante APIs REST.
- **Interfaces internas:** Comunicación segura y encriptada entre módulos.

#### 1.6.1.3.4 Comportamiento

Cada módulo opera de manera autónoma, permitiendo actualizaciones y consultas sin interrumpir otros procesos.

#### 1.6.1.3.5 Restricciones

- Las dependencias circulares no están permitidas.
- Cada módulo debe ser lo suficientemente desacoplado para facilitar su mantenimiento.

### 1.6.1.4 Mecanismos de Variabilidad

Esta vista considera las siguientes variabilidades:

- **Adaptación modular:** Posibilidad de añadir nuevos módulos, como un sistema de predicción con IA, sin alterar los módulos existentes.
- **Configuración dinámica:** Ajustes en tiempo de ejecución mediante parámetros personalizados.

#### 1.6.1.5 Antecedentes de la Arquitectura

La modularidad fue seleccionada como principio clave para garantizar escalabilidad, facilidad de integración y mantenimiento. Este diseño facilita futuras actualizaciones y la incorporación de nuevas tecnologías como inteligencia artificial para análisis predictivo.

## 1.6.2 Vista de Componentes y Conectores

### 1.6.2.1 Descripción de la Vista

Esta vista detalla la interacción en tiempo de ejecución entre los componentes del sistema y los conectores que facilitan la comunicación. Los componentes principales representan unidades ejecutables del sistema, mientras que los conectores especifican las vías de comunicación.

### 1.6.2.2 Presentación Principal

Un diagrama UML de componentes muestra los elementos principales del sistema:

- API REST para gestionar solicitudes.
- Base de datos PostgreSQL para almacenamiento de datos.
- Interfaces de usuario para acceso y operación.

### 1.6.2.3 Catálogo de Elementos

#### 1.6.2.3.1 Elementos

- **API REST:** Componente principal para interactuar con el sistema.
- **Base de datos PostgreSQL:** Almacén central de información.
- **Frontend:** Interfaz gráfica de usuario.

#### 1.6.2.3.2 Relaciones

- API REST se conecta con la base de datos para operaciones CRUD.
- El frontend consume datos proporcionados por la API REST.

#### 1.6.2.3.3 Interfaces

- **APIs públicas:** Puntos finales expuestos por la API REST.
- **Conexiones internas:** Comunicación entre API y base de datos mediante controladores.

#### 1.6.2.3.4 Comportamiento

El flujo comienza en la interfaz de usuario, que envía solicitudes a la API REST, que a su vez consulta o actualiza la base de datos según sea necesario.

#### 1.6.2.3.5 Restricciones

- Uso obligatorio de HTTPS para todas las comunicaciones externas.
- Todas las transacciones deben estar registradas para auditoría.

### 1.6.2.4 Mecanismos de Variabilidad

- **Escalabilidad horizontal:** Agregar instancias adicionales de componentes para manejar alta demanda.
- **Replicación de base de datos:** Mejora en la disponibilidad y redundancia.

## 1.6.2.5 Antecedentes de la Arquitectura

La arquitectura basada en componentes y conectores fue seleccionada para garantizar un comportamiento óptimo en tiempo de ejecución, alta disponibilidad y compatibilidad con sistemas externos.

## 1.7 Relaciones con otros SAD

No aplica.

## 1.8 Proceso de Subida de éste SAD

#### 1.8.1.1 Reporte de Discrepancias, Errores, Inconsistencias u Omisiones

Los lectores que identifiquen cualquier discrepancia, error, inconsistencia u omisión en este Documento de Arquitectura de Software (SAD) deberán seguir el siguiente procedimiento para reportarlo:

1. **Identificación del problema:**
    a. Describa detalladamente el error encontrado.
    b. Incluya la sección, página o figura específica donde se detectó el problema.
2. **Formato de reporte:**
    a. Complete el formulario de reporte de errores disponible en el repositorio del proyecto o solicítelo al equipo responsable.
    b. El formulario incluye los siguientes campos obligatorios:
        i. Nombre del reportante.
        ii. Fecha de reporte.
        iii. Descripción del problema.
        iv. Impacto identificado.
        v. Sugerencia de corrección.
3. **Envío del reporte:**
    a. Envíe el formulario al correo electrónico: **arquitectura@agriculturecompany.com**.

# 2 Trasfondo de la Arquitectura

## 2.1 Trasfondo del Problema

### 2.1.1 Vista General del Sistema

El sistema **Agriculture Company** está diseñado para optimizar procesos agrícolas clave a través de un conjunto de módulos interconectados. Estos módulos abarcan:

1. *Gestión de Cultivos*: Supervisión del estado de los cultivos y administración de recursos agrícolas esenciales como agua, fertilizantes y pesticidas.

2. *Gestión de Inventarios*: Registro y actualización en tiempo real de los niveles de stock de insumos y productos.

3. *Gestión de Distribución y Ventas*: Coordinación logística y control de ventas, asegurando que los productos lleguen frescos al mercado.

El sistema utiliza tecnologías modernas como bases de datos relacionales (PostgreSQL), contenedores Docker para garantizar portabilidad y consistencia, y protocolos de comunicación segura como HTTPS.

### 2.1.2 Metas y Contexto

Los objetivos clave del sistema son:

1. *Automatizar la Gestión Agrícola*: Reducir el tiempo dedicado a tareas manuales mediante herramientas digitales para la planificación, monitoreo y distribución.

2. *Mejorar la Productividad*: Garantizar que los agricultores puedan tomar decisiones informadas basadas en datos en tiempo real.

3. *Asegurar la Escalabilidad*: Diseñar una arquitectura capaz de manejar un aumento en los volúmenes de datos y usuarios durante temporadas de cosecha intensivas.

4. *Integrar Tecnologías Emergentes*: Proveer una base sólida que permita incorporar funcionalidades avanzadas como inteligencia artificial (IA) y análisis predictivo en el futuro.

### 2.1.3 Requisitos Clave

Los requisitos principales que influyeron en el diseño de esta arquitectura incluyen:

- *Procesamiento en Tiempo Real*: Garantizar que los datos se procesen rápidamente para evitar retrasos en decisiones críticas, como el riego o la distribución.
- *Tiempos de Respuesta Rápidos*: Ofrecer consultas y actualizaciones de inventario casi instantáneas, especialmente durante temporadas de alta demanda.
- *Seguridad de Datos*: Proteger la información sensible del sistema mediante autenticación de usuarios, cifrado de datos (HTTPS) y auditorías.
- *Interoperabilidad*: Facilitar la integración con sistemas agrícolas externos y plataformas en la nube.
- *Mantenimiento y Evolución*: Proveer un diseño modular que permita actualizar o reemplazar componentes sin interrumpir el sistema.

## 2.2 Trasfondo de la Solución

El diseño del sistema se basa en una **arquitectura modular**, lo que facilita el mantenimiento, la escalabilidad y la integración con nuevos módulos o tecnologías. Los principales enfoques adoptados son:

1. *División Modular*: El sistema está dividido en módulos independientes pero interconectados (Cultivo, Inventario, Ventas y Distribución), asegurando que cada uno tenga responsabilidades claramente definidas.
2. *Uso de Contenedores Docker*: Esto permite que la aplicación sea portable y funcione de manera consistente en diferentes entornos (desarrollo, pruebas y producción).

3. *Futura Integración de IA*: Aunque no está implementada actualmente, la arquitectura está preparada para incorporar análisis predictivo y recomendaciones basadas en datos climáticos y de suelo.

4. *Protocolos de Comunicación Segura (HTTPS)*: Todas las interacciones del sistema utilizan cifrado para proteger la privacidad y la integridad de los datos.

### 2.2.1 Enfoques Arquitectónicos

El uso de arquitecturas orientadas a datos y microservicios, ofrecen una major adaptabilidad para cada caso de uso del Sistema, emplenado así, una arquitectura modular mediant el uso de los siguientes enfoques:

1. *División Modular*: El sistema está dividido en módulos independientes pero interconectados (Cultivo, Inventario, Ventas y Distribución), asegurando que cada uno tenga responsabilidades claramente definidas.

2. *Uso de Contenedores Docker*: Esto permite que la aplicación sea portable y funcione de manera consistente en diferentes entornos (desarrollo, pruebas y producción).

3. *Futura Integración de IA*: Aunque no está implementada actualmente, la arquitectura está preparada para incorporar análisis predictivo y recomendaciones basadas en datos climáticos y de suelo.

4. *Protocolos de Comunicación Segura (HTTPS)*: Todas las interacciones del sistema utilizan cifrado para proteger la privacidad y la integridad de los datos.

### 2.2.2 Resultados de Análisis

Los resultados del análisis que satisfacen los requisitos del sistema:

1. **Escalabilidad y Rendimiento**:

    - Se ha diseñado la base de datos para soportar **grandes densidades de datos**, dividiendo grandes tablas (como Cosecha y Ventas) para su uso independiente. Esto mejora significativamente la capacidad de respuesta y permite un procesamiento eficiente durante períodos de alta actividad en el sistema.

- Los **índices en columnas clave** (ID_Cultivo, Fecha_cosecha, ID_Punto_Venta) aceleran las consultas y optimizan el uso de recursos del sistema.

2. **Disponibilidad de Datos**:

   - Se utiliza replicación de bases de datos para asegurar la **alta disponibilidad** de la información, incluso en caso de fallos del sistema o picos de carga.

3. **Seguridad y Consistencia**:

   - La arquitectura garantiza la integridad referencial entre las entidades a través de claves foráneas, asegurando que los datos relacionados (como Encargo y Punto de Venta) sean válidos.

   - Validaciones específicas, como restricciones que aseguran que Cantidad_cosecha y Capacidad_Carga sean valores positivos, reducen errores en la operación.

## 2.2.3 Cobertura de Requerimientos

### 2.2.3.1 Gestión de Cultivos:

**Objetivo:** Dar soporte a la planificación de cultivos disponibles, asegurando que el usuario pueda realizar una gestión segura y efectiva de las áreas de cultivo y optimizar su uso (ver figura 2 para entendimiento del proceso, p. 32).

**Funcionalidades:**

*Planificación y Seguridad de Datos*: Asegurar que el usuario pueda verificar fácilmente qué cultivos y silos están disponibles, evitando superposiciones o mal uso del terreno. Implementación de IA para

**Recomendaciones:** Aunque está prevista para fases futuras, se deberá estructurar el sistema para que pueda integrar un modelo de IA basado en patrones climáticos, suelo y datos de enfermedades para recomendar las mejores ubicaciones de cultivos y ciclos de producción.

**Interacciones con Otros Módulos:**

*Con el Inventario*: Integrar la información de cosechas planificadas y en crecimiento, lo que permite un control completo de recursos.

*Con el Sistema de Clima y Suelo*: Para mejorar la calidad de las recomendaciones de IA, este módulo podrá conectarse en fases futuras a datos meteorológicos y análisis de suelo.

### 2.2.3.2 Gestión de Inventario

**Objetivo:** Proporcionar un registro detallado de las cosechas, incluyendo ubicación, fecha de cultivo, estado de crecimiento y cantidad de recursos disponibles en cada silo y almacén.

**Funcionalidades:**

*Registro Detallado de Cosechas*: Mantener un registro actualizado de la cantidad de cada cosecha y su ubicación específica, apoyando la planificación de distribución.

*Estado de Crecimiento*: Permitir a los usuarios rastrear el progreso de cada cultivo, desde su plantación hasta la cosecha, para tener información precisa sobre disponibilidad futura.

**Interacciones con Otros Módulos:**

*Con Gestión de Cultivos*: Sincronizar los datos para asegurar que la planificación de nuevos cultivos tome en cuenta la disponibilidad de recursos y el estado de crecimiento actual de cada cultivo.

*Con Distribución y Ventas*: Facilitar un acceso rápido a los niveles de inventario, permitiendo ver la disponibilidad inmediata para ventas y planificación de distribución.

#### 2.2.3.3 Gestión de Distribución y Ventas

**Objetivo:** Coordinar la distribución y ventas con la disponibilidad de inventario y cultivos, minimizando tiempos de espera y evitando cuellos de botella en la entrega.

**Funcionalidades:**

*Optimización de Distribución*: Organizar rutas y fechas de entrega según el inventario disponible y los pedidos.

**Interacciones con Otros Módulos:**

*Con Inventario*: Permitir reservas automáticas de inventario una vez que se confirma una venta, evitando sobreventas y coordinando la disponibilidad.

*Con Gestión de Cultivos*: Sincronizar el calendario de cosechas con el de distribución para garantizar una oferta continua de productos frescos.

### 2.2.4 Resumen de Cambios en la Versión Actual

Los cambios y decisiones incorporados en la arquitectura reflejan las siguientes prioridades:

1. **Preparación para Crecimiento Futuro**:
    - El diseño modular permite la adición de nuevos módulos o funcionalidades, como por ejemplo:
        - **Integración con sistemas de clima y suelo** para proporcionar recomendaciones inteligentes basadas en datos ambientales.
2. **Estrategias de Escalabilidad**:
    - **Optimización del sistema de bases de datos** mediante buena estructuración de la base de datos.
3. **Mejora Continua de Seguridad y Fiabilidad**:
    - Registro detallado de auditoría para todas las transacciones críticas, fortaleciendo la trazabilidad del sistema.
4. **Simplificación del Mantenimiento**:
    - La arquitectura facilita la actualización o reemplazo de componentes específicos sin afectar la operación global del sistema, gracias a su diseño desacoplado.

## 2.3 Consideración de Reutilización en Línea de Productos

La arquitectura del sistema Agriculture Company está diseñada para ser modular y reutilizable, lo que facilita su adaptación a otros contextos dentro y fuera del sector agrícola:

1. **Reutilización de Módulos Existentes**:
    - **Gestión de Inventarios**: Este módulo puede ser reutilizado en industrias como el almacenamiento logístico o la manufactura, donde se requiere un control eficiente de insumos y productos terminados.

- **Gestión de Distribución y Ventas**: Es adaptable para optimizar cadenas de suministro en otros sectores, como la distribución minorista o el comercio electrónico.

2. **Posibilidades de Variación**:

    - La arquitectura admite modificaciones o ampliaciones específicas para satisfacer necesidades regionales o del cliente, como la integración con nuevas bases de datos o sistemas de terceros.

3. **Base para Nuevas Aplicaciones**:

    - El enfoque modular y la infraestructura de contenedores Docker permiten que el sistema sea base para aplicaciones en sectores como la ganadería (gestión de animales y recursos) o la pesca (gestión de capturas y distribución).

# 3 Vistas

Las vistas arquitectónicas se dividen en tres categorías principales:

1. Vista de Módulos (Module View):
   Describe cómo el sistema está dividido en módulos que representan unidades de implementación. Responde preguntas como:

   - ¿Qué responsabilidades funcionales tiene cada módulo?
   - ¿Qué otros módulos están relacionados por generalización o especialización?
   - ¿Qué módulos colaboran entre sí y cómo?

2. Vista de Componentes y Conectores (Component-and-Connector View):
   Representa cómo los componentes del sistema interactúan durante el tiempo de ejecución. Responde preguntas como:

   - ¿Cuáles son los principales componentes ejecutables y cómo interactúan?
   - ¿Qué partes del sistema están replicadas o corren en paralelo?
   - ¿Cómo fluye la información a través del sistema?

3. Vista de Asignación (Allocation View):
   Muestra la relación entre los elementos de software y los entornos externos donde el software es desarrollado y ejecutado. Responde preguntas como:

   - ¿Dónde se ejecuta cada componente de software (hardware, servidores)?
   - ¿Cómo están asignados los elementos de software a los equipos de desarrollo?

Las vistas presentadas en este SAD son las siguientes:

| Nombre de la vista | Tipo de vista que la define | Tipos de elementos y relaciones mostrados | | ¿Es una vista de módulos? | ¿Es una vista de componentes y conectores? | ¿Es una vista de asignación? |
|---|---|---|---|---|---|---|
| Vista del modelado de datos | Vista de módulo | | | Sí | No | No |
| Vista de interacción del sistema | Vista de componente y conector | Componentes y comunicación entre ellos | | No | Sí | No |
| Vista de Asignación de Hardware | Vista de locación | Hardware | Software | No | No | Sí |

Tabla 3: Vistas representadas en el SAD.

# 3.1 Vista del Modelado de Datos

## 3.1.1 Descripción de la Vista

Esta vista presenta el diseño de la base de datos del sistema, destacando las entidades principales y las relaciones entre ellas. Es fundamental para garantizar que los datos sean consistentes, escalables y fáciles de gestionar.

### 3.1.2 Vista Previa de los Paquetes

La vista está organizada en los siguientes paquetes:

**3.1.3 Entidades principales:** Cultivo, Cosecha, Silo, Punto de Venta, Ventas, Encargo, Vehículo (ver figura 1, p. 31).

**3.1.4 Relaciones clave:**

3.1.5 Cultivo ↔ Cosecha (1:N).

3.1.6 Cosecha ↔ Silo (1:N).

3.1.7 Punto de Venta ↔ Ventas (1:N).

### 3.1.8 Antecedentes de la Arquitectura

El modelo de datos está diseñado para soportar la gestión integral de cultivos, inventarios y ventas, con un enfoque en la integridad referencial entre entidades.

### 3.1.9 Mecanismos de Variabilidad

La base de datos admite variabilidad mediante:

- *Particionado horizontal*: División de datos por temporadas o ubicaciones geográficas.
- *Índices optimizados*: Mejora de consultas en campos clave como fechas y IDs.

### 3.1.10 Vista de Paquetes

#### 3.1.5.1 Paquete de Entidades

1. **Presentación Principal:**
   La base de datos incluye tablas como Cultivo, Cosecha, Silo, Punto de Venta, Ventas, Encargo y Vehículo.

2. **Catálogo de Elementos:**

   - ***Cultivo*:** ID único, tipo, área cultivada, fechas de siembra/cosecha, estado y necesidades de tratamiento.

   - ***Cosecha*:** ID único, cantidad cosechada, área, y relación con Cultivo.

   - ***Silo*:** Capacidad, contenido y relación con Cosecha.

3. **Diagrama de Contexto:**
   Representa las relaciones entre las tablas y cómo interactúan para actualizar inventarios y gestionar cosechas.

4. **Mecanismos de Variabilidad:**
   Soporte para esquemas adaptables según el tamaño de la operación agrícola.

## 3.2 Vista de Interacción del Sistema

### 3.2.1 Descripción de la Vista

Esta vista se centra en la interacción entre componentes del sistema en tiempo de ejecución, utilizando APIs REST y protocolos HTTP/HTTPS.

### 3.2.2 Vista de Paquetes

Incluye:

- *Conectores*: Comunicación segura entre módulos (p. ej., Cultivo ↔ Inventario).

- *Componentes*: Módulos de Cultivo, Inventario, Distribución y Ventas.

### 3.2.2 Antecedentes de la Arquitectura

El diseño asegura un flujo continuo de datos, habilitando decisiones en tiempo real y sincronización entre módulos.

### 3.2.4 Mecanismos de Variabilidad

- *Escalabilidad horizontal*: Adición de servidores para manejar mayor carga.
- *Replicación*: Mejora de disponibilidad mediante redundancia.

### 3.2.3 Vista de Paquetes

Incluyen diagramas de componentes y conectores que muestran interacciones críticas como las siguientes:

- Pantalla de gestión de cultivos ↔ API de Inventario.
- Pantalla de ventas ↔ API de distribución.

## 3.3 Vista de Asignación de Hardware

### 3.3.1 Descripción de la Vista

- Describe cómo los módulos de software se asignan a servidores y dispositivos.

### 3.3.2 Paquetes de la Vista

- *Infraestructura*: Servidores para bases de datos y APIs.
- *Dispositivos cliente*: PCs, laptops, móviles.

### 3.3.3 Mecanismos de Variabilidad

3.3.4 *Distribución geográfica*: Balanceo de carga entre regiones.

3.3.5 *Compatibilidad*: Asegurar funcionamiento en múltiples plataformas (Windows, Linux, Android, iOS).

# 4 Relaciones entre vistas

**Vista de Descomposición Modular y Vista de Componentes y Conectores:**

- Los módulos definidos en la Vista de Descomposición Modular (como Cultivo, Inventario, Distribución y Ventas) se corresponden directamente con los componentes definidos en la Vista de Componentes y Conectores.
- Cada módulo de la descomposición tiene un equivalente en los componentes que interactúan en tiempo de ejecución.

**Vista de Componentes y Conectores y Vista de Asignación:**

- Los componentes de la Vista de Componentes y Conectores, como la API REST y la base de datos PostgreSQL, están asignados a contenedores específicos y servidores físicos definidos en la Vista de Asignación.
- Esta relación asegura que los elementos arquitectónicos sean desplegables y escalables.

**Vista de Descomposición Modular y Vista de Asignación:**

- Los módulos definidos en la Vista de Descomposición Modular se asignan a las capas y entornos de despliegue detallados en la Vista de Asignación.
- Esto garantiza que las dependencias entre módulos sean consideradas en el diseño de la infraestructura.

#### 4.1.1.1 Relaciones Específicas Entre Vistas

| Elemento | Vista de Origen | Vista Relacionada | Relación |
|---|---|---|---|
| Módulo de Cultivo | Descomposición Modular | Componentes y Conectores | El módulo se implementa como un componente ejecutable que interactúa con la base de datos. |
| API REST | Componentes y Conectores | Asignación | La API REST se asigna a un contenedor Docker en el servidor de aplicaciones. |
| Base de datos PostgreSQL | Componentes y Conectores | Asignación | La base de datos está asignada a un nodo específico de infraestructura. |

| | | | |
|---|---|---|---|
| Relaciones entre módulos | Descomposición Modular | Componentes y Conectores | Las relaciones entre módulos se reflejan como conectores entre componentes. |

Tabla 4: Relaciones específicas entre vistas.

#### 4.1.1.2 Consistencia Entre Vistas

Se ha realizado una validación para garantizar que las vistas sean consistentes entre sí:

Cada módulo en la Vista de Descomposición Modular tiene un equivalente en la Vista de Componentes y Conectores.
Las asignaciones descritas en la Vista de Asignación coinciden con las dependencias y relaciones definidas en las otras vistas.

# 5 Materiales de Referencia

| | |
|---|---|
| IEEE 1471 | ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000. |

# 6 Directorio

## 6.1 Glosario

| Término | Definición |
|---|---|
| software architecture | La estructura o estructuras del sistema **Agriculture Company**, que comprenden los elementos de software (módulos como Cultivo, Inventario, Distribución, Ventas), las propiedades visibles externamente de esos elementos y las relaciones entre ellos. |
| view | Una representación de un sistema completo desde la perspectiva de un conjunto relacionado de preocupaciones [IEEE 1471]. En este proyecto, una vista puede representar los módulos del sistema, los componentes en tiempo de ejecución o las asignaciones a la infraestructura física. |
| view packet | La unidad más pequeña de documentación arquitectónica que podría ser útil para un stakeholder. En este proyecto, un paquete de vistas incluye la descripción de módulos individuales o la interacción entre componentes específicos. |
| viewpoint | Una especificación de las convenciones para construir y utilizar una vista. En el sistema **Agriculture Company**, los puntos de vista definen cómo modelar los módulos (como Cultivo, Inventario), los componentes y sus interacciones en tiempo de ejecución, y las asignaciones a entornos físicos. |
| Inventario | Módulo que administra la disponibilidad y |

| | |
|---|---|
| | almacenamiento de productos agrícolas en silos y almacenes, asegurando actualizaciones en tiempo real para apoyar las decisiones de distribución y ventas. |
| Docker | Plataforma de contenedorización empleada para desplegar los componentes del sistema de manera eficiente y reproducible, permitiendo que los microservicios se ejecuten en cualquier entorno. |

Tabla 5: Glosario.

## 6.2 Lista de acronimos

| | |
|---|---|
| API | Application Programming Interface; Application Program Interface; Application Programmer Interface |
| HTTP/HTTPS | (Protocolo de Transferencia de Hipertexto/Protocolo de Transferencia de Hipertexto Seguro) |
| IEEE | Institute of Electrical and Electronics Engineers |
| UML | Unified Modeling Language |

Tabla 6: Lista de acrónimos.

# 7 Figuras y tablas

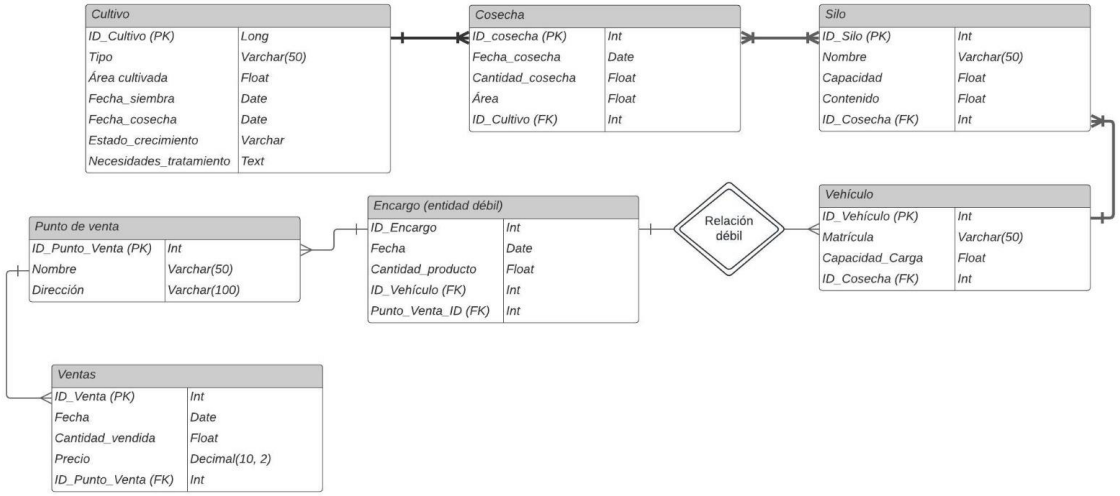


Figura 1: Diagrama Entidad-Relación sobre el proyecto.

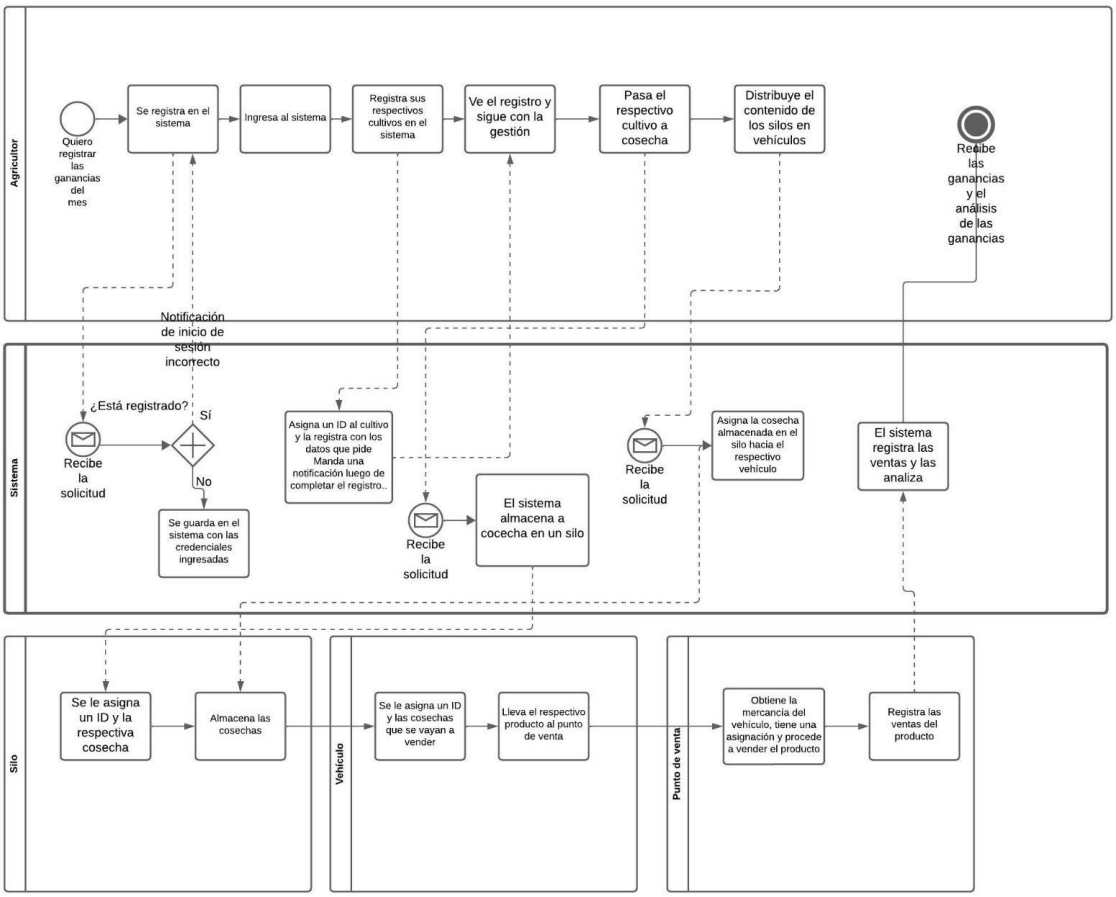


Figura 2: Diagrama de Procesos de Negocio.