Fastrack: Soluciones de *IT* para la gestión de producción por sistema fasón

Alumnos:

- Cacace, Camila
- Luna, Lautaro
- Presa, Martiniano
- Reale, Valentina

Director:

- Hinojal, Hernán

Co-Director:

- Genín, Fernando

Proyecto final para optar el grado de Ingeniero en Informática

Mar del Plata, 10 de julio de 2025

Autorización Repositorio Institucional -RINFI

Se presenta conjuntamente con la versión final del Trabajo Final

Repositorio Institucional RINFI, Facultad de Ingeniería, UNMDP

En calidad de TITULARES de los derechos de autor de la obra que se detalla a continuación, y sin infringir según mi conocimiento derechos de terceros, por la presente informo a la Facultad de Ingeniería de la UNMDP mi decisión de concederle en forma gratuita, no exclusiva y por tiempo ilimitado la autorización para:

1) Publicar el texto del trabajo más abajo indicado, exclusivamente en medio digital, en el sitio web de la Facultad y/o Universidad, por Internet, a título de divulgación gratuita de la producción científica generada por la Facultad, a partir de la fecha especificada.

2) Permitir a la Biblioteca que, sin producir cambios en el contenido, establezca los formatos de publicación en la web para su más adecuada visualización y la realización de copias digitales y migraciones de formato necesarias para la seguridad, resguardo y preservación a largo plazo de la presente obra:

Autor 1: Lautaro Luna Documento: 42455403 Teléfono: 2262677900 E-mail: lauluna817@gmail.com	 Firma 1
Autor 2: Valentina Reale Documento: 43508069 Teléfono: 2234988084 E-mail: valenreale123@gmail.com	 Firma 2
Autor 3: Camila Belén Cacace Documento: 43581411 Teléfono: 2235782616 E-mail: camilacacace128@gmail.com	 Firma 3
Autor 4: Martiniano Presa Documento: 42630024 Teléfono: 2235939497 E-mail: martiniano.presaaa@gmail.com	 Firma 4

Director/a: Hernán Hinojal Documento: 26056421 Leg.	Firma Director/a

2. Título obtenido:

Ingeniero/a en Informática

3. Identificación/Título de la Obra:

Fastrack: Soluciones de IT para la gestión de producción por sistema fasón

4. AUTORIZO la publicación bajo con la licencia Creative Commons BY-NC-ND Atribución-NoComercial-Sin Obra Derivada.

5. **Nota de Embargo:** Para aquellas obras que NO pueden ser de acceso a texto completo por razones de acuerdos previos con empresas o instituciones; por razones de índole comercial u otras razones; se procederá según lo establecido en Art. 6 de la Ley 26899 de Repositorios digitales institucionales de acceso abierto:

> **ARTICULO 6º —** En caso que las producciones científico-tecnológicas y los datos primarios estuvieran protegidos por derechos de propiedad industrial y/o acuerdos previos con terceros, los autores deberán proporcionar y autorizar el acceso público a los metadatos de dichas obras intelectuales y/o datos primarios, comprometiéndose a proporcionar acceso a los documentos y datos primarios completos a partir del vencimiento del plazo de protección de los derechos de propiedad industrial o de la extinción de los acuerdos previos antes referidos.
>
> Asimismo, podrá excluirse la difusión de aquellos datos primarios o resultados preliminares y/o definitivos de una investigación no publicada ni patentada que deban mantenerse en confidencialidad, requiriéndose a tal fin la debida justificación institucional de los motivos que impidan su difusión. Será potestad de la institución responsable en acuerdo con el investigador o equipo de investigación, establecer la pertinencia del momento en que dicha información deberá darse a conocer. A los efectos de la presente ley se entenderá como "metadato" a toda aquella información descriptiva sobre el contexto, calidad, condición o características de un recurso, dato u objeto, que tiene la finalidad de facilitar su búsqueda, recuperación, autentificación, evaluación, preservación y/o interoperabilidad.

En razón de lo expuesto, si el Trabajo se encuentra comprendido en el caso de que su producción esté protegida por derechos de Propiedad Industrial y/o acuerdos previos con terceros que implique la confidencialidad de los mismos, el/la directora/a debe indicar a continuación motivos y fecha de finalización del embargo:

NO SE AUTORIZA la publicación antes de la fecha __/__/_____ por los siguientes motivos:

……

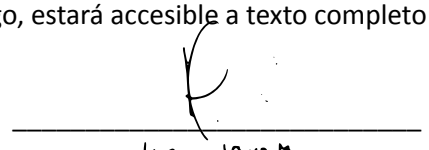
……

……

……

……

Cumplido el plazo del embargo, estará accesible a texto completo según contempla la normativa vigente.



Director/a del TF

Índice

Agradecimientos

A nuestras familias, por su constante apoyo a lo largo de toda nuestra carrera.

A la Universidad Nacional de Mar del Plata y sus docentes, por brindarnos la formación académica que nos permitió desarrollar este trabajo. En especial a nuestros profesores Hernán Hinojal y Fernando Genín, por su dedicación y orientación como director y co-director de este proyecto.

A nuestros compañeros y futuras colegas, con quienes compartimos todos estos años, y con quienes esperamos seguir construyendo un camino profesional.

Y a nosotros mismos, por la dedicación, el esfuerzo y la perseverancia que nos permitieron alcanzar este objetivo.

1. Resumen

Este proyecto integrador se centró en el desarrollo de un software de control de producción industrial, orientado a satisfacer las necesidades específicas del sector de producción por fasón, asegurando la trazabilidad a lo largo de toda la cadena de suministro.

El término "fasón" hace referencia a un modelo de manufactura por encargo, en el cual una empresa subcontrata a otra para llevar a cabo la producción bajo especificaciones predefinidas. En este esquema, la empresa contratante proporciona los insumos o materias primas, mientras que la empresa contratada se encarga del proceso productivo.

Para la implementación del proyecto, se diseñó un sistema capaz de responder a los requerimientos generales del sector de producción por fasón, además de desarrollar módulos específicos adaptados a las empresas demandantes: ViaVeg, perteneciente a la industria alimenticia, y Coppens, dedicada a la industria pesada.

La ejecución de esta iniciativa contó con financiamiento parcial del FITBA (Fondo de Innovación Tecnológica de la Provincia de Buenos Aires), lo que permitió acelerar la digitalización de ViaVeg, optimizar sus procesos internos y facilitar su adopción de nuevas tecnologías.

Como objetivo final, se estableció la creación de un software genérico con potencial de comercialización en el mercado. La solución desarrollada tiene como meta impulsar la digitalización en diversas empresas y simplificar la adopción de herramientas tecnológicas dentro del modelo de producción fasón, ofreciendo una plataforma escalable y adaptable a distintas operativas industriales.

2. Introducción

La transformación digital ha generado un cambio significativo en los procesos industriales, permitiendo una gestión más eficiente, segura y precisa de la producción. En este contexto, la incorporación de soluciones tecnológicas para la trazabilidad de productos se ha convertido en una necesidad clave para mejorar el control de calidad, cumplir con normativas vigentes y aumentar la competitividad empresarial.

Este sistema no solo busca garantizar la transparencia y seguridad de los datos a lo largo de toda la cadena de suministro, sino también proporcionar herramientas analíticas avanzadas para la toma de decisiones.

Uno de los principales desafíos en este ámbito es la centralización de la información, ya que actualmente muchas empresas operan con sistemas fragmentados que dificultan la integración y el acceso eficiente a los datos de producción. La plataforma desarrollada aborda este problema mediante una arquitectura monolítica modular, que permite una organización clara de las funcionalidades del sistema.

Asimismo, se ha diseñado una interfaz de usuario intuitiva que facilita la gestión de información en tiempo real por parte de los distintos actores involucrados en el proceso productivo. La herramienta genera reportes automáticos y alertas, lo que contribuye a optimizar la eficiencia operativa. Tiene el potencial de implementarse en diversas industrias y adaptarse a distintos modelos de producción, impulsando la digitalización y la adopción de tecnologías innovadoras en el sector industrial.

3. Análisis del problema

3.1. Dominio del problema

El fasón es un acuerdo comercial mediante el cual una empresa encarga a otra la fabricación de productos según especificaciones predefinidas, proporcionando los insumos necesarios. Este modelo de externalización de la producción presenta desafíos clave en cuanto a la trazabilidad y la gestión de la cadena de suministro.

El sistema propuesto tiene como objetivo centralizar y optimizar el control sobre cada etapa del proceso productivo, brindando a las empresas una visión clara y en tiempo real de la cadena de suministros. De esta forma, se asegura la trazabilidad de los insumos y la calidad del producto final, mejorando la coordinación entre los actores involucrados. Además, permite reducir fallos en la calidad, al facilitar la identificación y corrección temprana de posibles inconvenientes durante el proceso.

3.2. Problema a resolver

En la industria manufacturera, muchas empresas optan por externalizar sus procesos de producción a terceros mediante acuerdos de fasón. Esta modalidad presenta riesgos significativos:

- **Interrupciones en la cadena de suministro**:
  La ausencia de un sistema adecuado de rastreo provoca retrasos y errores en la producción.
- **Compromiso de la calidad**:
  La falta de seguimiento dificulta la detección temprana de fallos, afectando la calidad del producto final.
- **Aumento de costos**:
  La ineficiencia en la gestión operativa incrementa los costos generales del proceso.

Sin procesos adecuados y con un uso limitado de herramientas tecnológicas, aumentan los riesgos de interrupciones y se compromete la seguridad y calidad de los productos.

La falta de trazabilidad también dificulta la identificación de problemas durante el proceso productivo, lo que afecta tanto a los productores como a los consumidores, quienes valoran conocer el origen y las condiciones de producción de lo que consumen.

Estas deficiencias pueden tener consecuencias a largo plazo: si no se abordan, las empresas arriesgan su sostenibilidad operativa, se exponen a sanciones legales y pierden competitividad en un mercado global cada vez más exigente.

4. Proyecto

4.1. Objetivo General

El propósito de este proyecto es desarrollar un software genérico que integre funcionalidades comunes al modelo de producción por fasón, estructurado de forma modular con el fin de

facilitar su adaptación a las diversas necesidades de las industrias y de las empresas que lo utilicen.
La finalidad del sistema es proporcionar herramientas para la gestión y el análisis de datos, con el propósito de lograr una toma de decisiones más informada, permitiendo a las organizaciones identificar oportunidades de mejora, optimizar el uso de recursos y garantizar la continuidad operativa de manera más eficiente.

4.2. Requerimientos

El proceso de elicitación fue fundamental para captar no solo las expectativas explícitas del cliente, sino también aquellas implícitas relacionadas con sus procesos internos y flujos operativos. Durante las entrevistas realizadas con los referentes de Vía Veg y Coppens, se pudieron destacar varios puntos importantes sobre lo que se esperaba del sistema.

Ambas empresas expresaron la necesidad de poder registrar y controlar la producción desde la recepción de insumos hasta la entrega del producto final. Esto implica que el sistema debe permitir cargar de forma clara qué insumos se usaron, en qué cantidad, a qué lote pertenecen, y cómo avanza a lo largo del tiempo.

Una de las necesidades más destacadas durante las reuniones fue la importancia de llevar un control preciso del stock, con especial énfasis en la trazabilidad por lote. Cada insumo cuenta con un número de lote que debe registrarse tanto al momento del ingreso como al del egreso de mercadería. Esto permite realizar un seguimiento de calidad a lo largo del tiempo. De esta forma, si se recibe mercadería en mal estado esa situación puede quedar registrada.

El seguimiento de costos también surgió como una preocupación central. Es necesario conocer el valor de cada compra y contar con un historial de precios de las materias primas. Actualmente, las compras se realizan mayormente por WhatsApp, lo cual genera una falta de registro formal. Por eso, es fundamental implementar un sistema en el que se pueda registrar una orden de compra y compararla con la mercadería recibida.

Durante el relevamiento, se hizo hincapié en que, dentro del conjunto de insumos, existen distintas clasificaciones: algunos son consumidos directamente, mientras que otros se combinan para generar insumos intermedios.

La elaboración de cada producto debe estar claramente definida, especificando los insumos que requiere, el depósito del cual se extraen y el número de lote al que pertenecen. Además, el sistema debería permitir registrar cualquier desviación entre la cantidad planificada y la realmente producida.

Otro punto planteado durante la elicitación fue la necesidad de acceder al sistema desde diferentes ubicaciones y dispositivos. Dado que varias etapas del proceso productivo y logístico se llevan a cabo en distintos lugares físicos —como depósitos, plantas de producción o incluso en entornos de trabajo remoto—, se consideró esencial que el sistema sea accesible tanto desde computadoras como desde dispositivos móviles, permitiendo así un trabajo más flexible.

Si bien hubo requerimientos comunes entre ambas empresas, también surgieron necesidades particulares según la modalidad de trabajo y el enfoque de cada una.

En el caso de VíaVeg, una de las prioridades fue controlar la producción por batch, es decir, por tandas. Asimismo, se planteó llevar un registro de la merma, entendida como la diferencia entre la cantidad de materia prima que entra al proceso y la que efectivamente forma parte del producto final.

En el caso de Coppens, el foco estuvo puesto en el control de calidad al momento de terminar la producción. Se solicitó que el sistema permita cargar los resultados de las pruebas de control.

Es importante mencionar que estos requerimientos fueron analizados con detalle y formalizados en la sección “Anexo”.

4.3. Alcance inicial

El proyecto se enfocará en el desarrollo de un sistema integral para la trazabilidad de productos e insumos en industrias que operan bajo la modalidad de producción por fasón. Su objetivo principal es optimizar la planificación, ejecución y seguimiento de los procesos productivos, así como el control del flujo de insumos a lo largo de toda la cadena de valor.

El alcance del sistema comprenderá funcionalidades específicas para la gestión de producción, tales como la administración de órdenes de producción, la asignación de recursos y el

seguimiento del avance de los lotes en cada etapa del proceso. Además, incorporará herramientas para la gestión logística de insumos, incluyendo el registro de ingresos y egresos de bienes en depósitos, la trazabilidad entre proveedores, y la gestión de stock disponible para producción.

Quedarán excluidos del alcance del proyecto los aspectos tributarios, fiscales, contables y de nómina. Del mismo modo, no se abordarán en profundidad las particularidades de cada proceso productivo, lo que permitirá implementar la trazabilidad de insumos a lo largo de la cadena de suministro, sin depender del tipo de industria. Esta abstracción favorece una trazabilidad robusta, manteniendo al mismo tiempo la flexibilidad necesaria para futuras adaptaciones.

4.4. Entregables del proyecto

Se detalla a continuación los entregables de este proyecto:

- **Especificación de Requerimientos:**
  - **Requerimientos funcionales comunes (*core*)**: Funcionalidades que estarán disponibles en todas las versiones del software.
  - **Requerimientos específicos por industria**: Adaptaciones o extensiones necesarias para casos particulares.
  - **Requerimientos no funcionales.**
- **Documentación:**
  - **Documentación de la API:** descrito con *Swagger*
  - **Diagrama de arquitectura del sistema**
  - **Diagrama de contexto**
  - **Diagrama de entidad-relación**
  - **Diagrama de componentes**
  - **Diagramas de secuencia**
- **Código fuente del sistema:**
  - **Backend**
  - **Frontend**
  - **Dockerfiles**
  - **Esquema SQL**

4.5. Aporte del proyecto

4.5.1 Beneficiarios

El desarrollo del sistema beneficia principalmente a industrias que operan bajo el modelo de producción por fasón. Si bien el proyecto fue originalmente impulsado para la empresa ViaVeg, su enfoque evolucionó hacia una solución más genérica, lo que posibilita su implementación en la empresa Coppens y su futura aplicación en otras organizaciones del sector.

Entre los beneficiarios directos se encuentran las áreas operativas de las empresas, que podrán gestionar de forma más eficiente las órdenes de producción, el control de stock de insumos y productos, la relación con proveedores y la coordinación de tareas con terceros.

De este modo, la herramienta no se limita exclusivamente a las empresas destinatarias originales, sino que se proyecta como una solución adaptable para otras organizaciones que trabajen con esquemas similares.

Pensando más allá del contexto inmediato, el proyecto también puede aportar valor a consultores, desarrolladores de soluciones tecnológicas, quienes podrían encontrar en esta herramienta una base para nuevos proyectos, estudios de caso o propuestas de mejora continua.

4.5.2. Roles del sistema

El sistema contempla distintos roles de usuario, cada uno con responsabilidades específicas que responden a las distintas áreas involucradas en el proceso productivo. De esta forma, se lograría organizar y controlar las operaciones, estableciendo permisos y funcionalidades diferenciadas según el perfil. A continuación se detallan los roles existentes:

Administrador del sistema: Es responsable de la configuración general, la administración de usuarios y la supervisión del correcto funcionamiento del sistema.

Responsable de producción: Gestiona las órdenes de producción. Asigna la producción a terceros y supervisa su avance. Controla el estado general de las operaciones productivas.

Operario de planta: Recibe las órdenes de producción. Registra los avances diarios, los consumos de insumos y los tiempos de fabricación. Entrega los productos terminados y reporta incidencias durante el proceso productivo.

Responsable de compras: Gestiona las órdenes de compra y remitos. Coordina la adquisición de insumos con los proveedores. Relaciona las compras con los proveedores correspondientes y organiza la entrega de insumos, incluso para producción tercerizada.

Responsable de stock: Controla la disponibilidad y movimientos de stock, tanto de insumos como de productos terminados. Verifica la recepción de insumos desde los proveedores y de productos elaborados desde planta o terceros. Gestiona transferencias entre depósitos.

Usuario básico: Tiene acceso limitado al sistema. Puede visualizar insumos, productos y stock pero no tiene permisos para realizar modificaciones.

Usuario maestro: Tiene acceso completo a todas las funcionalidades del sistema, con excepción de la gestión de usuarios.

4.5.3. Análisis FODA

En esta sección se presenta un análisis FODA, una herramienta estratégica que permite identificar las fortalezas, oportunidades, debilidades y amenazas de un proyecto. El objetivo es comprender tanto los aspectos internos que pueden potenciar o limitar el desarrollo (fortalezas y debilidades), como los factores externos del entorno que pueden influir positiva o negativamente (oportunidades y amenazas).

Fortalezas

- El equipo de desarrollo cuenta con una experiencia previa de colaboración en proyectos, lo que favorece una dinámica de trabajo consolidada.
- Se tiene contacto directo con empresas con trayectoria en el sector, lo que permite un desarrollo adaptado a las necesidades reales.

Oportunidades

- No se identifican soluciones tecnológicas que contemplen de manera específica la modalidad de producción por fasón, caracterizada por la tercerización de procesos productivos y la utilización de múltiples depósitos.
- Crecimiento sostenido del mercado y la demanda por herramientas digitales que optimicen la producción y fortalezcan la trazabilidad en diversas industrias

Debilidades

- El equipo de desarrollo presenta una limitada experiencia en la creación de software orientado específicamente a la industria, lo cual puede generar una curva cara de aprendizaje inicial.
- Falta de disponibilidad horaria del grupo de desarrollo.
- Desconocimiento parcial de algunas de las tecnologías a utilizar en el proyecto.

Amenazas

- Posible resistencia por parte de los operarios de empresas tradicionales al uso de nuevas herramientas tecnológicas.
- Posibilidad de que ingresen competidores con mayores recursos y decidan especializarse en el mismo nicho de mercado.
- Dificultad para identificar suficientes funcionalidades comunes entre empresas del sector, lo que pondría en cuestión la viabilidad de desarrollar una solución genérica.
- Cambios en las regulaciones que puedan requerir modificaciones significativas en la aplicación.
- Rápido avance tecnológico que podría dejar obsoletas las herramientas desarrolladas, obligando a futuras actualizaciones.

A partir de las debilidades y amenazas identificadas en este análisis, se desarrolló un análisis de riesgos específico, con el fin de anticipar los eventos que podrían materializarse.

4.5.4. Análisis de Riesgos

En esta sección se identifican y evalúan los principales riesgos que podrían afectar el desarrollo y la implementación del sistema. Cada riesgo se describe junto con sus posibles

consecuencias, y se le asigna un peso en función de su probabilidad de ocurrencia e impacto de acuerdo al criterio del equipo. El objetivo de este análisis es anticiparse a los obstáculos potenciales, cuantificar su relevancia y preparar estrategias que minimicen su efecto negativo sobre el proyecto.

La escala utilizada fue la siguiente:

Probabilidad (P):
1 – Poco probable
2 – Probable
3 – Muy probable

Impacto (Imp):
1 – Poco impacto
2 – Impacto significativo
3 – Gran impacto

La ponderación del riesgo se obtiene multiplicando la probabilidad por el impacto (P × Imp). Si el valor resultante es mayor a 6, se elabora un plan de contingencia específico para ese riesgo.

ID	Riesgo	Descripción	Consecuencia	P	Imp	Peso
R01	Obsolescencia tecnológica	Necesidad de migrar a nuevas tecnologías, lo que puede generar interrupciones en el servicio.	Interrupción del servicio y aumento de costos de actualización.	2	2	4
R02	Resistencia de empresas tradicionales	Las empresas con procesos convencionales pueden resistirse a adoptar el producto.	Dificultad en la adopción y posible pérdida de clientes.	2	3	6
R03	Entrada de competidores más grandes	Nuevas empresas con mayores recursos pueden ingresar al mercado.	Aumento de la presión competitiva y reducción de la cuota de mercado.	2	3	6

R04	Limitaciones del software por su diseño genérico	El software puede no adaptarse completamente a las necesidades específicas de los clientes.	Insatisfacción del usuario y pérdida de competitividad.	2	2	4
R05	Falta de flexibilidad del producto	El software puede no ser lo suficientemente adaptable para distintos clientes.	Pérdida de oportunidades de negocio y menor competitividad.	2	3	6
R06	Uso inadecuado del software	Errores en el ingreso de datos o en la configuración del sistema.	Inconsistencias en los resultados y toma de decisiones erróneas.	2	3	6
R07	Retrasos en el desarrollo del proyecto	Problemas en la planificación y ejecución del proyecto.	Aplazamiento en la entrega y costos adicionales.	2	3	6
R08	Retrasos en la implementación (deploy)	Problemas técnicos o logísticos durante el despliegue del sistema.	Interrupciones en el servicio y descontento de los clientes.	2	3	6

4.5.5. Planes de contingencia

Los planes de contingencia son estrategias pensadas para anticiparse a los riesgos que podrían afectar negativamente el desarrollo o funcionamiento del proyecto. Su objetivo es reducir la probabilidad de ocurrencia de dichos riesgos o, en caso de que se materialicen, mitigar su impacto.

Del análisis de riesgos surgió la necesidad de elaborar planes de contingencia para aquellos con un peso total igual o superior a seis. Se identificaron seis casos con este nivel de criticidad, por lo que se debieron desarrollar los siguientes planes:

- **R02**: Desarrollar estrategias de marketing y educación para destacar los beneficios del producto y abordar las preocupaciones de las empresas tradicionales.

- **R03**: Mantener un monitoreo constante del mercado y buscar continuamente formas de diferenciación y mejora del producto.
- **R05**: Investigar y priorizar los rubros más importantes para los clientes y planificar actualizaciones futuras para ampliar la cobertura del producto.
- **R06**: Implementar programas de capacitación para los usuarios, incluyendo documentación detallada y soporte técnico.
- **R07**: Considerar un *overhead* en las tareas más laboriosas por cualquier inconveniente o imprevisto que pueda surgir durante su desarrollo.
- **R08**: Planificar el despliegue con margen de tiempo suficiente para manejar imprevistos. Se contará con un equipo de soporte disponible durante la implementación para resolver problemas técnicos de manera inmediata y garantizar una transición sin interrupciones.

Si bien los riesgos fueron identificados y los planes de contingencia correspondientes fueron establecidos durante el desarrollo del proyecto, las acciones específicas, tanto preventivas como reactivas, no fueron detalladas en su momento. Estas fueron elaboradas de forma retrospectiva y se encuentran desarrolladas en la sección "Anexo", con el fin de documentar cómo se podrían haber abordado o cómo se actuó efectivamente en caso de haberse materializado el riesgo.

4.5.6. Análisis de mercado

Al desarrollar un proyecto de trazabilidad en la producción por fasón, es fundamental analizar el mercado y las soluciones existentes para capitalizar fortalezas y evitar los errores cometidos por otros competidores. A continuación, se presentan tres enfoques de soluciones de trazabilidad:

1. MapexTrace

Características principales:

- **Captura de datos**: Permite la recolección manual y automática de datos de trazabilidad de los lotes de entrada, utilizando tecnologías como códigos de barras y RFID.

- **Etiquetado automatizado**: Facilita la impresión de etiquetas con información detallada y en tiempo real de la trazabilidad de salida.
- **Historial de lotes**: Genera registros históricos de los lotes de materias primas utilizados en cada lote de salida.

2. NUT-AT Alimentación y Trazabilidad

Características principales:

- **Gestión integral**: Controla y registra cada etapa del proceso productivo, desde la recepción de materias primas hasta la distribución del producto final.
- **Cumplimiento normativo**: Ayuda a las empresas a cumplir con las regulaciones específicas del sector alimentario, garantizando la seguridad y calidad de los productos.
- **Control de calidad**: Incluye módulos para la gestión de controles de calidad en diferentes fases de la producción, permitiendo la detección temprana de posibles incidencias.

3. Módulos de trazabilidad en ERP con funcionalidades limitadas

Características principales:

- **Integración básica**: Ofrecen una integración elemental de la trazabilidad dentro de los procesos generales del ERP, permitiendo el seguimiento de lotes y productos.
- **Funcionalidades estándar**: Proporcionan herramientas básicas para el registro y seguimiento de productos, pero pueden carecer de funcionalidades avanzadas como la captura automática de datos o la gestión detallada de historiales de lotes.
- **Adaptabilidad limitada**: Al no estar especializados en la trazabilidad, estos módulos pueden no adaptarse completamente a las necesidades específicas de ciertos sectores industriales, limitando su eficacia en entornos que requieren un control exhaustivo.

Al analizar el mercado en el que competirá Fastrack, se identifican competidores indirectos con funcionalidades avanzadas en trazabilidad de producción, como MapexTrace y NUT-AT. No obstante, ninguna de estas soluciones se especializa en la trazabilidad de procesos tercerizados, lo que revela un nicho desatendido. Esta situación representa una oportunidad

para desarrollar una propuesta enfocada específicamente en las particularidades de la producción por fasón.

4.6. Estimación inicial

4.6.1. Planificación

Una vez definidos los requerimientos del sistema, se procedió a realizar una estimación inicial de las tareas, de acuerdo a su prioridad. Además, se consideró un *overhead* para las tareas más grandes.

El plan de trabajo fue estructurado en función de las prioridades de cada empresa. Las tareas comenzaron con las correspondientes a ViaVeg, ya que Coppens se incorporó al proyecto en una etapa posterior. Cabe destacar que la estimación de tiempos se basó en el conocimiento previo y la experiencia individual de cada integrante, pese a que el equipo no contaba con antecedentes en proyectos de esta magnitud.

La estructura de la estimación inicial comprendía las siguientes etapas:

1. Análisis ViaVeg
2. Diseño ViaVeg
3. Desarrollo ViaVeg
4. Validación ViaVeg
5. Testing ViaVeg
6. Deploy ViaVeg
7. Análisis Coppens
8. Diseño Coppens
9. Desarrollo Coppens
10. Validación Coppens
11. Capacitación ViaVeg
12. Testing Coppens
13. Capacitación Coppens
14. Deploy Coppens
15. Investigación y documentación del sistema genérico

Para organizar el cronograma, se elaboró un diagrama de Gantt que permitía visualizar la distribución temporal de las tareas y sus dependencias. La planificación inicial y la construcción del Gantt se basaron en la primera definición de requerimientos realizada a partir de las dos reuniones iniciales con el cliente. Posteriormente, los tiempos estimados para cada tarea fueron ajustados con mayor precisión a partir del *feedback* recibido en etapas sucesivas, lo que permitió refinar el alcance y optimizar la asignación de tareas dentro del cronograma.

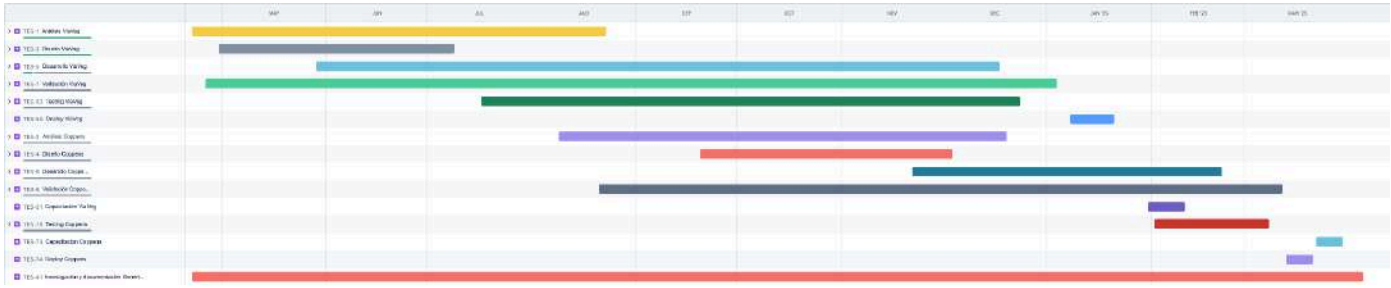


Fig 1. Diagrama de Gantt inicial

4.7. Metodologías

4.7.1. Metodología de trabajo

Para abordar el desarrollo de este proyecto de ingeniería se adoptó una metodología de trabajo basada en la organización por etapas, la colaboración entre integrantes y la constante revisión de los avances. El equipo se dividió en dos grupos principales: uno encargado del desarrollo del backend y otro del frontend. A pesar de esta división técnica, las decisiones estratégicas —como la planificación general, el diseño funcional, los plazos de entrega y la priorización de requerimientos— se tomaron de forma consensuada.

El trabajo se estructuró en base a objetivos parciales definidos para cada fase del proyecto, permitiendo así evaluar resultados intermedios y redirigir el enfoque si era necesario. La interacción con el entorno real del sistema a implementar —incluyendo entrevistas con actores clave del dominio— permitió contextualizar las decisiones y enriquecer el análisis.

Debido a las diferencias horarias y compromisos académicos de los integrantes, se optó por una modalidad flexible de organización, combinando encuentros presenciales, reuniones

virtuales semanales y comunicación asincrónica. Además, se documentaron las decisiones y los aprendizajes en cada etapa, fortaleciendo la trazabilidad del proceso y la integración de aportes individuales en un único producto.

4.7.2. Metodología de desarrollo

Al iniciar el proyecto se hizo evidente que los requerimientos no estaban sumamente definidos por parte del cliente. Por lo tanto, se optó por utilizar una metodología ágil, ya que si bien no es bueno que ocurran cambios significativos, este tipo de metodologías permiten minimizar el impacto. Su implementación representó un desafío inicial para el equipo, ya que no se contaba con una gran experiencia en este enfoque de trabajo. Sin embargo, a medida que se avanzó en el proceso, se logró adaptar y aprovechar sus ventajas.

Se eligió Scrum como la metodología a utilizar debido a que al ser un proyecto financiado por el FITBA, se requería de la pronta implementación del sistema, y esta metodología permite obtener un MVP (Producto Mínimo Viable) que incorporaría funcionalidades de forma iterativa e incremental.

En cuanto a los *sprints*, cada uno tenía una duración de dos semanas e incluía una planificación, una *weekly* y demos. Al inicio del proyecto se realizaban reuniones diarias, pero debido a la variedad de horarios dentro del equipo y la dificultad de coincidir en un horario fijo, se optó por reemplazarlas por reuniones semanales. Además, las dinámicas del grupo fueron mejorando y se reforzaron aquellos aspectos en los que se identificaban oportunidades de mejora, como la comunicación entre equipos, la claridad en la definición de tareas y la optimización de los tiempos de desarrollo.

4.7.3. Herramientas utilizadas

Durante el desarrollo del proyecto, se utilizaron diversas herramientas que facilitaron la organización del equipo, la gestión de tareas y la validación de los requerimientos. Cada una de ellas desempeñó un rol clave en la eficiencia y colaboración dentro del grupo:

- **Git**: Se utilizó para el control de versiones del código, permitiéndonos trabajar en paralelo sin riesgo de sobrescribir cambios y asegurando un historial claro de

modificaciones. Esto facilitó la integración de nuevas funcionalidades de manera segura.

- **Jira**: Fue la herramienta principal para la gestión de tareas y seguimiento del proyecto. Permitió organizar el trabajo en *sprints*, asignar responsabilidades, establecer prioridades y visualizar el estado de cada tarea mediante tableros Kanban. Gracias a Jira, se pudo mantener un flujo de trabajo eficiente y asegurando el cumplimiento de los plazos establecidos.

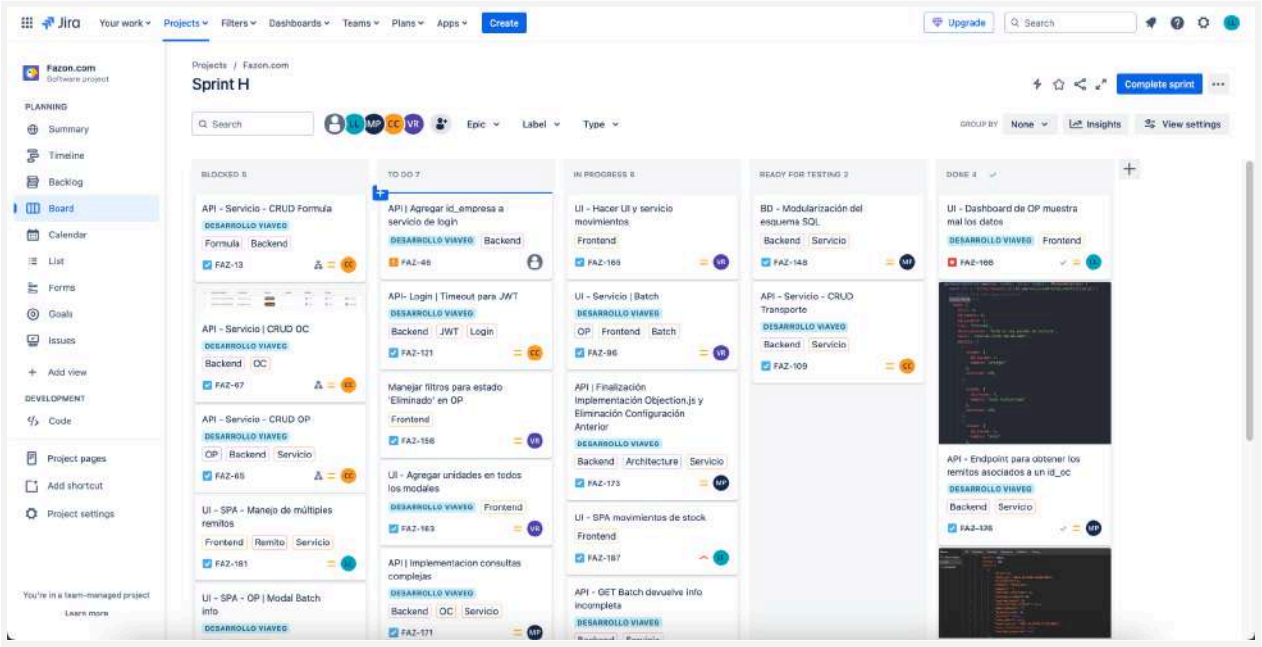


Fig 2. Tablero Kanban de Jira utilizado en el proyecto

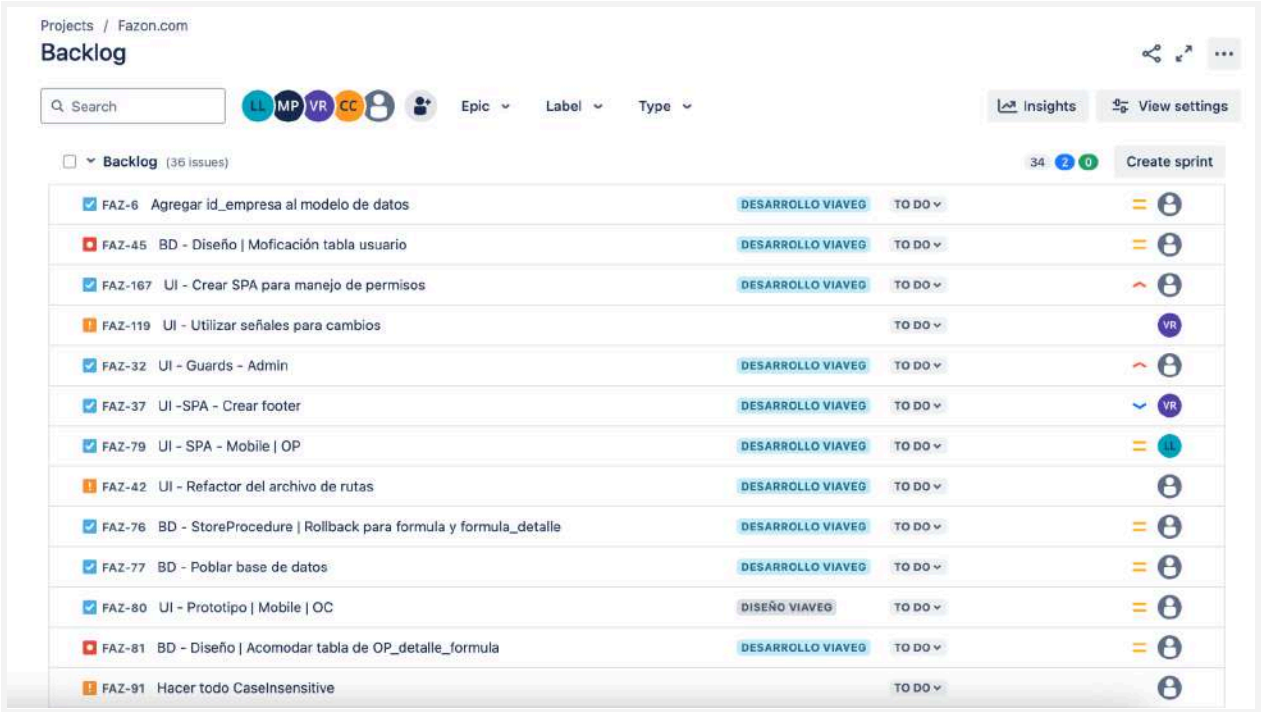


Fig 3. Backlog de Jira utilizado en el proyecto

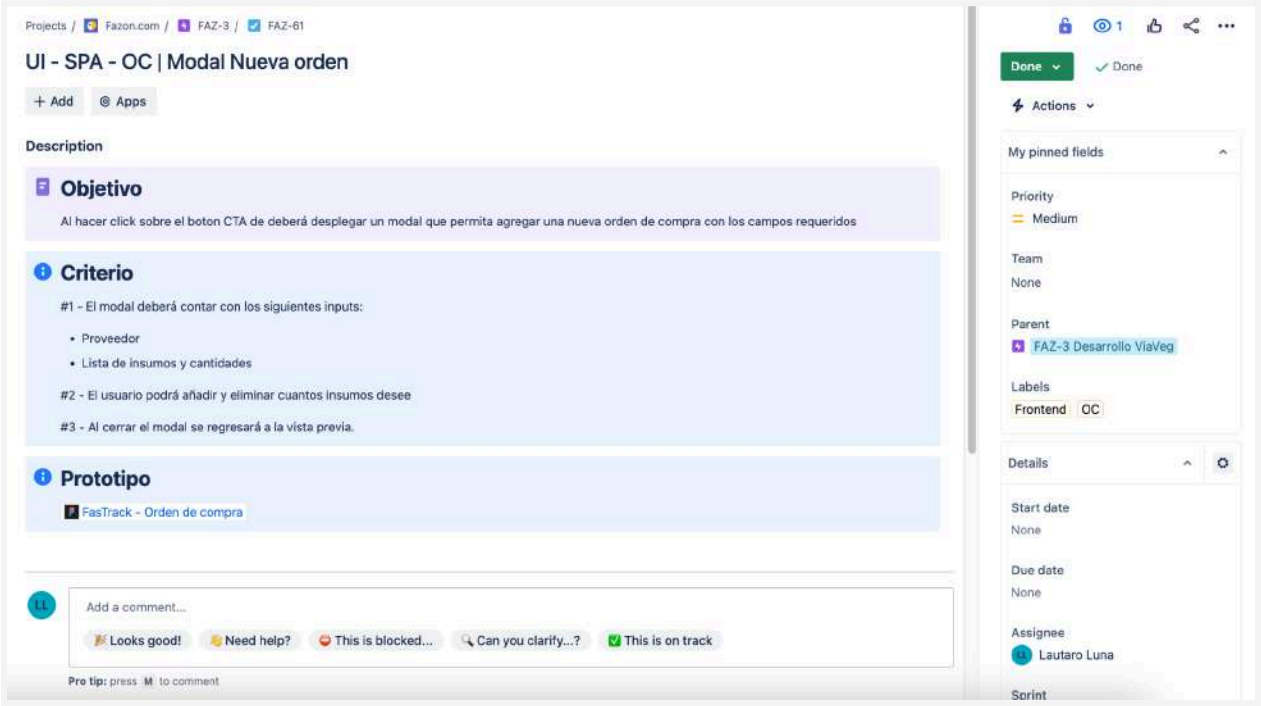


Fig 4. Tarjeta de Jira utilizado en el proyecto

El ciclo de vida de una tarjeta consistía en:

1. Creación de la tarea en el *backlog*
2. Refinación de la tarea
3. Asignación a un *sprint* en la columna *To Do*
4. Una vez iniciada se mueve a la columna *In Progress*
5. Una vez finalizada, se mueve a *Ready for Testing* para ser testeada
6. Si el test se aprueba, se marca como *Done*, de lo contrario vuelve a la columna *To Do* con su determinado *feedback*

- **Notion**: Se utilizó como una plataforma de colaboración para organizar la información del proyecto. Su flexibilidad permitió centralizar las notas del equipo, los requisitos del proyecto, la planificación de tareas y la gestión de recursos, todo en un solo lugar. Esta herramienta ayudó a mantener el enfoque y a tener una visión global del progreso del proyecto.
- **Postman**: Fue una herramienta esencial para la prueba y validación de las APIs desarrolladas. Permitió verificar el correcto funcionamiento de los servicios, identificar errores y garantizar que la comunicación entre los distintos componentes del sistema fuera efectiva.
- ***Swagger**:* Se utilizó para documentar y visualizar la API del sistema de manera clara y estructurada. Gracias a esta herramienta, se generó una documentación interactiva que facilitó la comprensión y prueba de los endpoints por parte de los desarrolladores y testers. Su capacidad de generar documentación en tiempo real aseguró que los equipos contaran siempre con información actualizada sobre la API, reduciendo posibles inconsistencias y mejorando la comunicación entre los desarrolladores *backend* y *frontend*.
- ***Draw.io* y Excalidraw**: Se utilizaron para la creación de diagramas y esquemas visuales, lo que facilitó la representación gráfica de la arquitectura del sistema, los flujos de datos y la estructura de la base de datos. Gracias a su interfaz intuitiva, fue posible diseñar modelos de manera colaborativa, permitiendo que todo el equipo tuviera una visión clara de la estructura del proyecto. Estas herramientas resultaron especialmente útiles para documentar procesos y facilitar la toma de decisiones técnicas.

4.7.4. Comunicación

Durante el desarrollo del proyecto integrador de ingeniería, la comunicación dentro del equipo y con los distintos actores involucrados fue fundamental para la coordinación de tareas, la resolución de problemas y la toma de decisiones. Se utilizaron diferentes medios y estrategias para mantener un flujo de información constante y eficiente. A nivel interno, se realizaron reuniones entre los miembros del equipo de manera presencial y virtual, utilizando plataformas como Discord o Google Meet, dependiendo de la disponibilidad de los integrantes.

Con el fin de recibir orientación y *feedback* sobre los avances del trabajo, se realizaron reuniones con el director del proyecto, así como con los representantes de ambas empresas. Por el lado de ViaVeg, se coordinaron los requerimientos y se validaron dichos avances. En el caso de Coppens, los encuentros se centraron en comprender sus procesos productivos, identificar diferencias clave respecto a ViaVeg y validar la aplicabilidad general de la solución desarrollada.

El uso de herramientas digitales fue un aspecto clave en nuestra comunicación. Gracias a Discord, fue posible coordinar tareas diariamente, mientras que Google Meet permitió reuniones virtuales cuando no era posible reunirnos en persona. Además, se implementó un tablero Kanban para visualizar el progreso del proyecto y hacer seguimiento de las tareas.

4.7.5. Reuniones con el cliente

Durante el desarrollo del proyecto, se llevaron a cabo diversas reuniones con representantes de las empresas ViaVeg y Coppens con el objetivo de recopilar información clave, validar requerimientos y recibir retroalimentación sobre el avance de la aplicación. En todas estas instancias participaron los cuatro integrantes del equipo de desarrollo, junto con el director del proyecto, quien estuvo presente en cada encuentro y cumplió un rol clave como facilitador del proceso de comunicación. Por parte de las empresas, los referentes principales fueron Juan Casareto, dueño de ViaVeg, y Julián Arévalo, coordinador de planificación de Coppens.

La reunión inicial con ViaVeg tuvo como propósito principal la elicitación de requerimientos, dado que en ese momento el equipo contaba únicamente con el contexto general proporcionado por el director del proyecto, quien inicialmente presentó la propuesta. En esta

instancia, se llevó a cabo una lluvia de ideas junto con el representante de la empresa, con el fin de obtener una visión más clara de la problemática y sus procesos internos.

Además de conocer el contexto, también fue necesario entender cómo funcionaban los flujos de trabajo y el circuito operativo de la empresa —es decir, los distintos pasos involucrados desde la recepción de insumos hasta la salida del producto final. Durante este proceso, se detectó que muchos de estos flujos no estaban claramente definidos por parte del cliente, por lo que fue necesario reconstruirlos, formalizarlos y validarlos con los referentes para poder avanzar con el diseño del sistema.

En una instancia posterior, el equipo presentó una versión inicial funcional de la aplicación web, utilizando datos de prueba con el objetivo de obtener *feedback* directo del cliente. Mostrar un prototipo en esta etapa temprana permitió validar si la solución se alineaba con sus expectativas y generar mayor confianza al mostrar avances concretos en el desarrollo. No obstante, surgieron nuevas dudas relacionadas con ciertos flujos operativos de la empresa, que fueron discutidas en detalle durante la reunión. Gracias a estas aclaraciones, el equipo pudo avanzar con mayor precisión en el desarrollo.

En las reuniones siguientes, el equipo continuó presentando avances del sistema, incorporando las modificaciones sugeridas por el cliente. Estas instancias permitieron validar los cambios implementados, resolver nuevas dudas y seguir ajustando la solución de forma iterativa, en base al *feedback* recibido durante el proceso de desarrollo.

Una vez completado el desarrollo, se presentó la versión final del software, con todas las funcionalidades implementadas y refinadas según las observaciones previas. Tras una revisión detallada, el representante de ViaVeg aprobó la solución, dando su conformidad con el producto elaborado.

Además de las instancias con ViaVeg, se llevaron a cabo reuniones con el representante de la empresa Coppens. En total, se realizaron tres reuniones virtuales con el referente técnico.

La primera reunión tuvo un carácter exploratorio. En ella, se presentaron los objetivos generales del proyecto y se indagó acerca de los principales procesos internos de la empresa. El responsable técnico aportó una visión complementaria a la de ViaVeg, destacando diferencias en los flujos de trabajo y en los sistemas de control interno, lo que permitió al equipo

considerar escenarios más diversos durante el diseño del sistema. Por ejemplo, se identificó que, a diferencia de ViaVeg, la producción en Coppens no utilizaba un sistema formal de trazabilidad por batch. Si bien trabajan con lotes, se manejan de forma más flexible, lo cual implicó adaptar ciertos aspectos del modelo de datos y la lógica de la aplicación. A partir de este encuentro, surgieron observaciones valiosas sobre posibles mejoras en la interfaz y en la forma en que se presentan los reportes.

En la segunda reunión ya se había definido que el producto sería de tipo genérico, por lo que el encuentro se centró en comparar los módulos existentes con los procesos propios de Coppens, para identificar cuáles se ajustaban mejor a su forma de trabajo. A partir de ese intercambio, se terminaron de definir tanto el núcleo del sistema como los módulos específicos para cada industria. En esa instancia, el referente técnico manifestó interés en incorporar una sección destinada al registro de evaluaciones de calidad de los productos. Se elicitaron los requerimientos correspondientes y se planificó su incorporación en etapas posteriores del desarrollo.

En esta etapa inicial, los requerimientos funcionales del sistema fueron definidos y ajustados en base al *feedback* del cliente, lo que permitió refinar el enfoque y adaptar mejor la solución a las necesidades concretas de la empresa.

En una reunión posterior, se realizó una demostración completa del sistema, mostrando en detalle las funcionalidades implementadas. El cliente validó formalmente la solución, confirmando que cumplía con los requerimientos acordados y que respondía a sus necesidades operativas. También se presentaron las opciones de despliegue disponibles, quedando a consideración del cliente su evaluación.

Cabe destacar que, además de los canales formales, se mantuvo una comunicación informal con ambos referentes a través del director del proyecto, quien actuó como intermediario. Esto permitió validar en distintos momentos decisiones puntuales del desarrollo y resolver dudas operativas con mayor agilidad. Esta modalidad se adoptó, en parte, debido a las dificultades para coordinar con ellos por cuestiones de disponibilidad horaria.

Asimismo, se realizaron validaciones conceptuales con el propio director, relacionadas con aspectos del dominio del problema, lo que ayudó a clarificar términos, supuestos y prioridades en la interpretación funcional del sistema.

5. Diseño del sistema

5.1. Arquitectura

Sistema core

A continuación se adjunta un diagrama de la arquitectura que utiliza el sistema core:

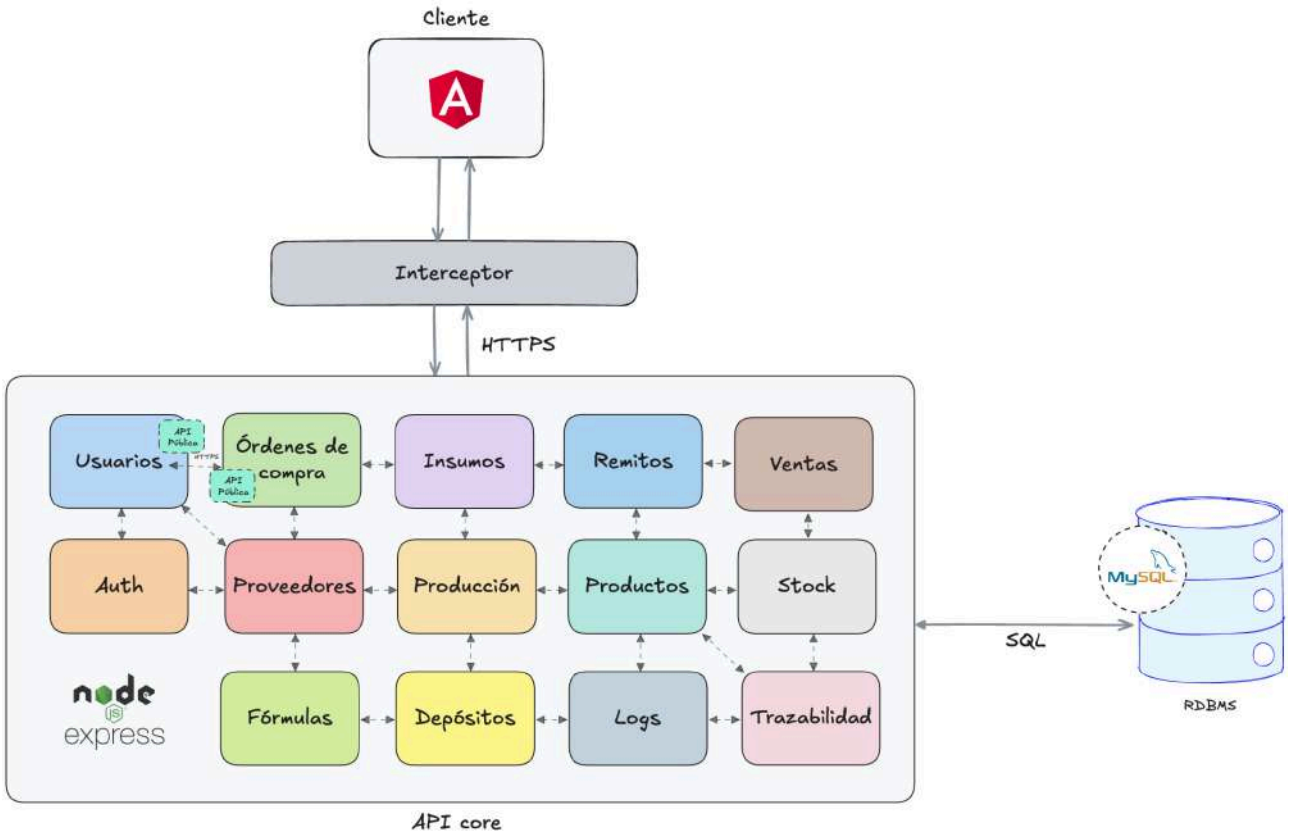


Fig 5. Diagrama de arquitectura

Dado el requerimiento identificado durante la elicitación que planteaba la necesidad de acceder al sistema desde distintas ubicaciones y dispositivos, se definió que el desarrollo se llevaría a cabo como una aplicación web. Esta decisión garantiza el acceso remoto, facilitando así un uso más flexible y de alcance amplio.

Además, el equipo contaba con experiencia previa en el desarrollo de aplicaciones web, lo que representó un factor determinante para optar por esta tecnología, ya que permitió aprovechar conocimientos existentes y reducir la curva de aprendizaje durante las etapas iniciales del proyecto.

Actualmente, el sistema implementa una arquitectura de tipo cliente-servidor, con un cliente web y un backend. Como componente intermedio, se implementó un interceptor utilizando la librería HttpInterceptor de Angular. Este interceptor se encarga del manejo global de errores, el registro de la actividad del usuario —lo que permite una trazabilidad más eficiente—, y además, se utiliza para mostrar un *spinner* o *loader* durante las solicitudes HTTP, mejorando así la experiencia del usuario al indicar que una operación está en curso.

En cuanto al cliente, se eligió una arquitectura de tipo SPA (*Single Page Application*), en la cual todo el contenido HTML, CSS y JavaScript se carga al acceder al sitio. De esta forma, Angular permite que la aplicación funcione íntegramente en una única página: intercepta los cambios en la URL y muestra el componente correspondiente sin necesidad de recargar. Este mecanismo es posible gracias al sistema de ruteo interno del *framework*, gestionado a través del módulo *RouterModule*, que permite definir un conjunto de rutas y sus respectivos componentes.

Al no utilizar renderizado del lado del servidor (SSR), todo el proceso de renderizado ocurre en el cliente. Este enfoque no solo permite una navegación más fluida, sino que también reduce la carga sobre el servidor y favorece una arquitectura más escalable, modular y sencilla de mantener.

Por otro lado, el backend se implementó bajo las normas que establece el patrón arquitectónico Monolítico Modular, con el objetivo de garantizar una estructura con bajo acoplamiento, alta cohesión y alineada a los principios SOLID. Esta elección no solo promueve buenas prácticas de desarrollo, sino que también favorece a atributos de calidad esenciales como la escalabilidad, la mantenibilidad y la interoperabilidad del sistema.

Patrón Monolítico Modular

El patrón arquitectónico monolítico modular se caracteriza por una clara separación de responsabilidades, donde cada módulo tiene una única responsabilidad. Los módulos interactúan entre sí mediante interfaces bien definidas, lo que permite modificar su implementación interna sin impactar al resto del sistema.

Para garantizar un sistema verdaderamente modular, es fundamental mantener los datos aislados, asegurando así la independencia y el bajo acoplamiento entre módulos. En este sentido, la arquitectura modular monolítica establece reglas estrictas para preservar la integridad de los datos: cada módulo solo puede acceder a sus propias tablas, sin compartir objetos ni tablas con otros módulos, y permitiendo únicamente joins entre tablas dentro del mismo módulo. Asimismo, los módulos deben ser autosuficientes y responsables de la gestión de sus propios datos, a los cuales otros módulos sólo pueden acceder a través de su API pública.

En este contexto, se optó inicialmente por una arquitectura modular monolítica en lugar de microservicios, debido a que representa una solución más simple y eficiente tanto en términos operativos como de costos. Este tipo de arquitectura no requiere una infraestructura distribuida compleja: se evita la necesidad de múltiples contenedores, bases de datos distribuidas, *gateways*, *load balancers* o herramientas de orquestación, lo que reduce significativamente los costos asociados al monitoreo, los servidores y las tareas de DevOps. Además, al trabajar con un único repositorio y proceso de ejecución, se facilita el desarrollo y el *debugging*, ya que es posible trazar errores de punta a punta, realizar refactorizaciones sin romper contratos entre servicios, y modificar la lógica de negocio con mayor libertad.

Por otra parte, las invocaciones internas entre módulos eliminan la latencia de red y los fallos intermitentes propios de las arquitecturas distribuidas.

Finalmente, este enfoque permite una evolución controlada del sistema, brindando la posibilidad de migrar a microservicios más adelante si el crecimiento o la escalabilidad lo requieren, pero sin asumir desde el inicio los altos costos y riesgos que implicaría una arquitectura distribuida sin la madurez técnica necesaria, y teniendo en cuenta el contexto económico que atraviesan las empresas actuales.

Gracias a la arquitectura implementada, se logró reutilizar el *core* y extender sus funcionalidades para adaptarlo a las necesidad específicas de cada industria, como se muestra a continuación.

Adaptación ViaVeg

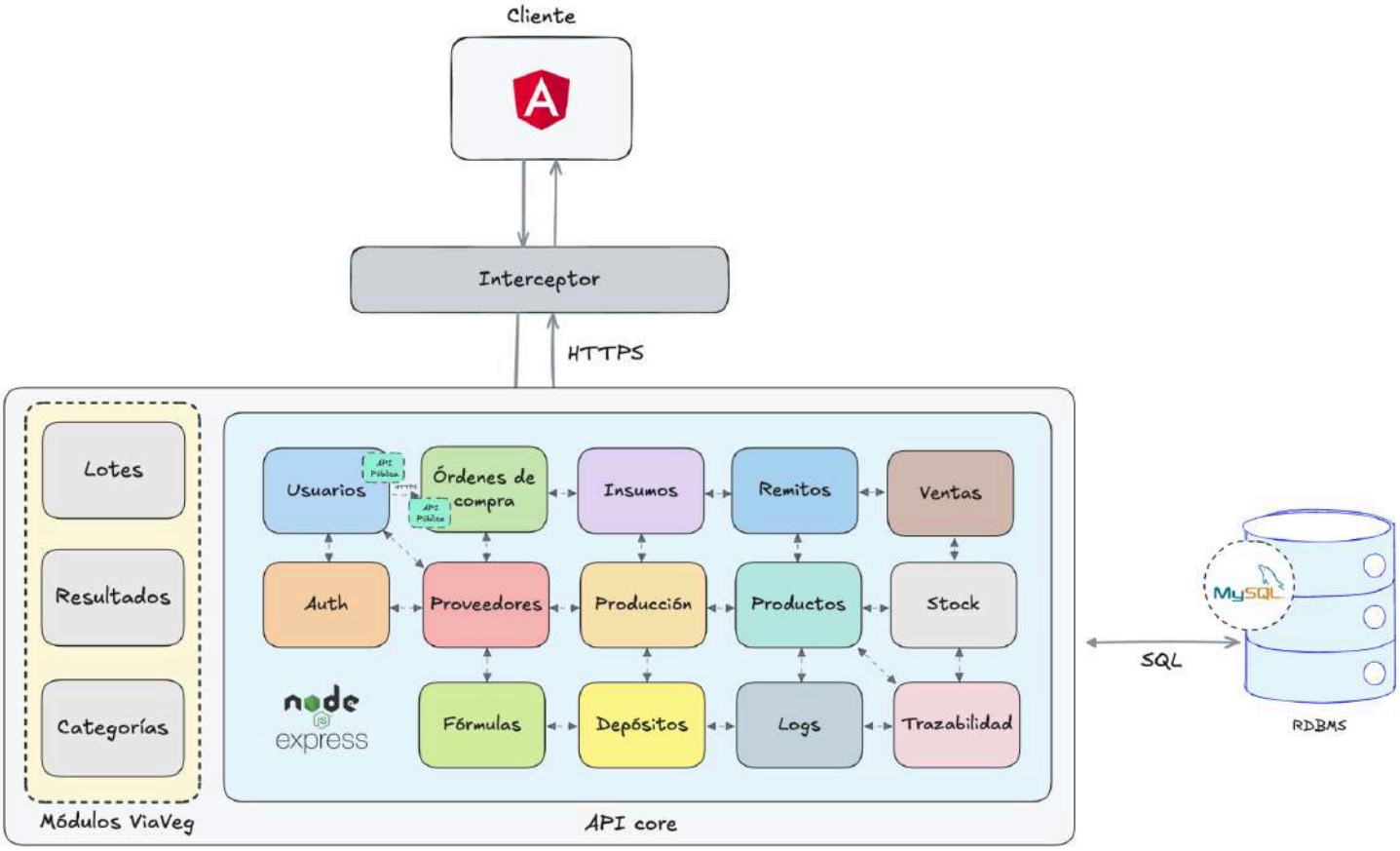


Fig 6. Diagrama de arquitectura para ViaVeg

Para el caso particular de ViaVeg, se añadieron tres módulos nuevos con funcionalidades que responden a los requerimientos propios de la industria. Aquellos son:

- Lotes
- Resultados
- Categorías

En la sección “Módulos específicos para ViaVeg” se puede encontrar una descripción detallada de los mismos.

Adaptación Coppens

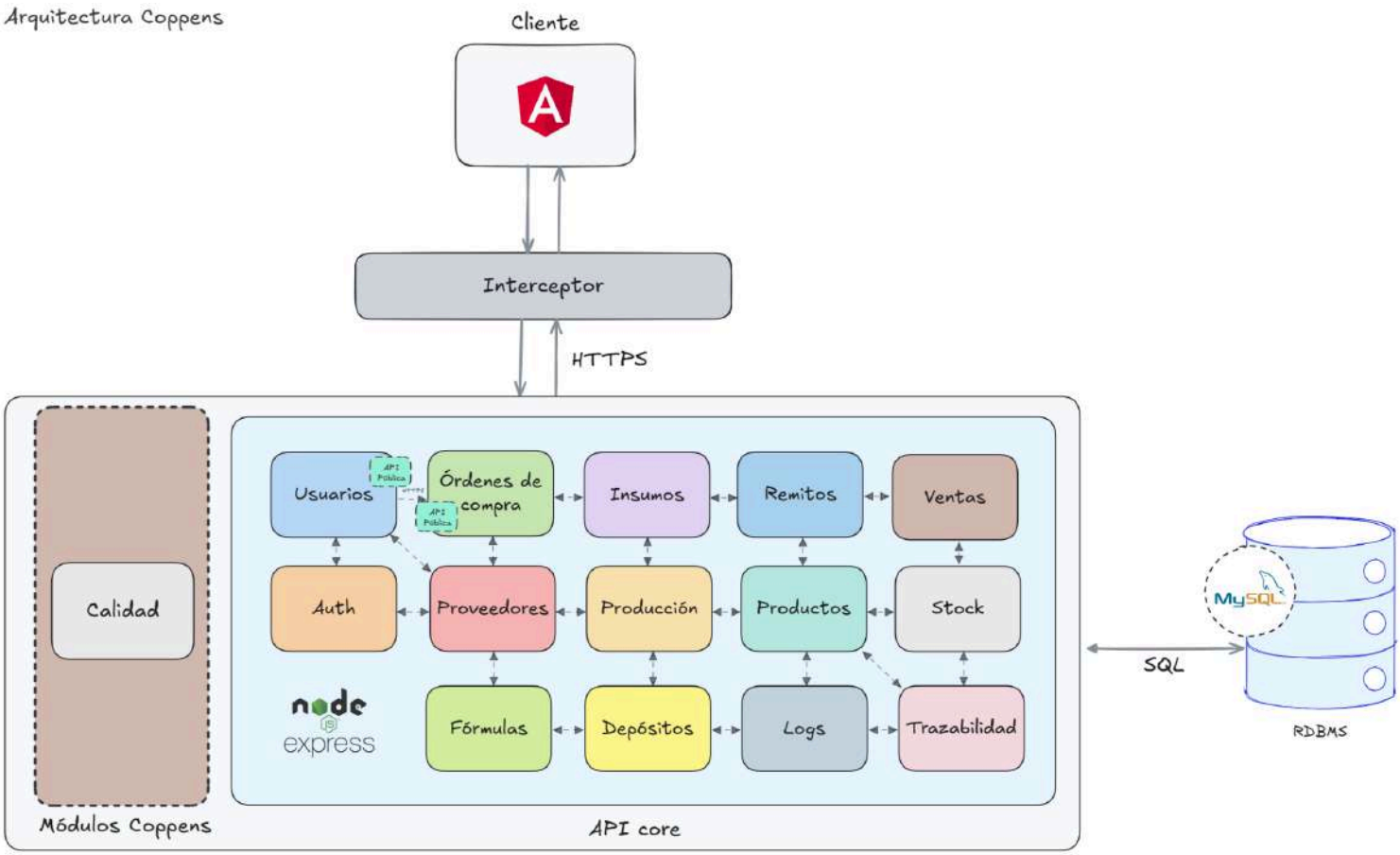


Fig 7. Diagrama de arquitectura para Coppens

Para el caso particular de Coppens, se añadió un módulo de calidad con funcionalidades que responden a los requerimientos propios de la industria.

En la sección “Módulos específicos para Coppens” se puede encontrar una descripción detallada del mismo.

5.2. Frontend

5.2.1. Tecnologías

El *frontend* de la aplicación fue desarrollado utilizando Angular 17, un *framework* de desarrollo web basado en TypeScript que facilita la creación de aplicaciones dinámicas y modulares. Angular permite el uso de componentes reutilizables, inyección de dependencias y una arquitectura basada en módulos, lo que contribuye a la mantenibilidad y escalabilidad del proyecto.

HTML y CSS

Angular utiliza HTML5 y CSS3 para la estructuración y el diseño visual de la aplicación. HTML5 proporciona un marcado semántico que permite una mejor accesibilidad y optimización en los motores de búsqueda. La aplicación está construida sobre una base sólida de etiquetas HTML5, lo que facilita la organización del contenido y mejora la interacción con el usuario. En cuanto al estilo y diseño visual, se usó CSS3 para darle formato a la interfaz.

Integración con APIs

En cuanto a la comunicación con el backend, Angular se encargó de gestionar las peticiones HTTP utilizando su módulo HttpClient, lo que permite consumir servicios RESTful de manera eficiente y segura. Esto se complementó con la implementación de RxJS para manejar las respuestas asincrónicas de las peticiones, utilizando Observables para representar flujos de datos que pueden ser suscritos y manipulados de manera reactiva.

5.2.2. Patrones de Diseño

En el desarrollo del frontend se aplicaron varios patrones de diseño para mejorar la organización del código y facilitar su reutilización. Angular, por su arquitectura basada en módulos y componentes, facilita la implementación de estos patrones de manera eficiente:

- **Component-Based Pattern:** Se utilizaron componentes reutilizables para estructurar la interfaz de usuario, favoreciendo la separación de responsabilidades y la reutilización de código.

- **Observable Pattern and Reactive Programming:** Mediante el uso de RxJS, se implementó un manejo eficiente de eventos y datos asincrónicos, permitiendo una actualización fluida de la interfaz sin necesidad de refrescar la página.
- **Dependency Injection Pattern:** Angular facilita este patrón mediante su sistema de Dependency Injection (DI), que permite a los desarrolladores definir cómo se resuelven las dependencias de los componentes y servicios.
- **MVC** (Model-View-Controller): Aunque Angular no sigue estrictamente el patrón MVC tradicional, adopta una variación conocida como MVVM (Model-View-ViewModel). Los componentes actúan como el ViewModel, enlazando la vista (HTML) con el modelo (datos).
- **Lazy Loading:** Se implementó carga diferida de módulos para mejorar el rendimiento y la velocidad de carga de la aplicación, cargando solo las partes necesarias en cada momento.
- **State Management con Señales:** Se utilizaron señales de Angular para gestionar el estado de la aplicación de manera eficiente y reactiva, asegurando una mejor gestión del ciclo de vida de los datos.
- **Singleton Pattern:** Se implementó en los servicios para garantizar una única instancia en toda la aplicación, optimizando el uso de recursos y la gestión de estado compartido entre múltiples componentes.
- **Factory Pattern:** Utilizado en la creación de servicios y componentes dinámicos, facilitando la encapsulación de la lógica de construcción de objetos complejos.

5.2.3. Información adicional

El CLI de Angular agiliza tareas comunes como la generación de componentes, testeo y despliegue acelerando los procesos y mejorando la productividad.

Su sistema de *change detection*, optimizado con la introducción de señales (signals), permite una gestión más eficiente del estado y de las actualizaciones en la interfaz, reduciendo renderizados innecesarios y mejorando el rendimiento general de la aplicación.

Además, herramientas como las directivas y los pipes ofrecen formas reutilizables de manipular el DOM y transformar datos para su presentación.

5.2.4. Paquete de Terceros Principales

Se integraron diversas librerías y paquetes para optimizar el desarrollo:

- **AG Grid:** Se utilizó para la visualización y manipulación de datos en tablas interactivas, proporcionando funciones avanzadas como filtrado, ordenamiento y edición en línea.
- **Angular Material:** Forma parte del ecosistema de Angular y fue utilizado para la creación de una interfaz de usuario adaptable con componentes predefinidos basados en Material Design.
- **NgxUiLoaderModule:** Utilizado para mostrar un indicador de carga en la aplicación, mejorando la experiencia del usuario.
- **RxJS:** Librería esencial para la programación reactiva, manejo de eventos y flujos de datos asíncronos. Aunque forma parte del ecosistema de Angular, se instala por separado.

5.2.5. Manejo de Errores

Para garantizar la estabilidad de la aplicación y mejorar la experiencia del usuario, se implementaron diversas estrategias de manejo de errores:

- **Interceptores HTTP:** Se utilizaron interceptores para capturar y gestionar errores en las solicitudes HTTP, proporcionando mensajes de error adecuados al usuario.
- **Manejo Global de Errores:** Se implementó un servicio centralizado para capturar errores no manejados y registrar logs de manera estructurada.
- **Mensajes de Notificación:** Mediante Angular Material se implementaron notificaciones visuales para informar al usuario sobre errores o acciones exitosas.
- **Fallbacks y Retries:** Se establecieron estrategias de reintento y valores predeterminados en caso de fallas en la carga de datos.

Estas estrategias garantizaron una aplicación más robusta y resiliente ante posibles fallos en la comunicación con el backend o errores en la lógica de negocio.

5.3. Backend

5.3.1 Tecnologías

Para el desarrollo del *backend* del proyecto, se optó por Express.js, un *framework* para Node.js que permite crear aplicaciones y APIs de manera eficiente. Su flexibilidad y ligereza han permitido estructurar el servidor de acuerdo con las necesidades, sin imponer una arquitectura rígida.

Uno de los principales motivos para elegir Express.js es su capacidad para manejar peticiones HTTP de manera sencilla, facilitando la creación de rutas y la gestión de *middleware* para procesar solicitudes antes de enviarlas a los controladores. Además, su estructura modular permite separar las rutas, controladores y servicios.

Otro aspecto clave es su fácil integración con librerías para conectar con bases de datos como MySQL, lo que permite realizar consultas de manera estructurada y eficiente. Esto contribuye a mejorar el rendimiento y la organización del *backend*.

5.3.2. Patrones de diseño

En el desarrollo del *backend*, se aplicaron varios patrones de diseño para mejorar la organización, mantenibilidad y escalabilidad del código.

- **Singleton Pattern**: Utilizado para la conexión a la base de datos MySQL, aseguró que exista una única instancia compartida en toda la aplicación, lo que optimiza el uso de recursos y evita la sobrecarga de conexiones innecesarias.
- **Repository Pattern:** Posibilita el desacoplamiento de la lógica de acceso a datos de la lógica de negocio, permitiendo una mejor estructura y reutilización del código.
- **Dependency Injection Pattern**: Facilitó la inyección de dependencias en los controladores y servicios, lo que mejora la testeabilidad y flexibilidad del código.

5.3.3. Paquete de terceros principales

Para mejorar la seguridad, la eficiencia y la facilidad de desarrollo de la aplicación, se decidió incorporar varios paquetes de terceros considerados esenciales. Estos paquetes permiten optimizar diversas funcionalidades y garantizar una experiencia robusta tanto para los desarrolladores como para los usuarios finales. A continuación, se detallan los principales paquetes utilizados y su función:

- **bcryptjs**: Para encriptar contraseñas de manera segura y proteger la información sensible de los usuarios.
- **cors**: *Middleware* que permite gestionar el acceso entre diferentes dominios, habilitando el uso de la API desde otros orígenes.
- **express**: *Framework* minimalista y flexible para crear la estructura de la API y manejar las rutas y peticiones HTTP.
- **jsonwebtoken**: Utilizado para la autenticación mediante *tokens* JWT, asegurando que los usuarios sean validados de manera segura.
- **mysql2**: Cliente para interactuar con MySQL de manera eficiente, soportando promesas y consultas asíncronas.
- **objection**: ORM basado en Knex que facilita la manipulación de modelos, relaciones y consultas complejas en la base de datos.
- **nodemon**: Herramienta que monitorea los cambios en el código y reinicia automáticamente el servidor durante el desarrollo para facilitar la productividad.
- **swaggerJSDoc:** Genera la documentación de los *endpoints* a partir de los comentarios en el código.
- **swagger-ui-express:** Paquete que sirve una interfaz visual para interactuar con la documentación Swagger.

5.3.4. Manejo de errores

Para garantizar la estabilidad del sistema y una correcta retroalimentación frente a errores, se implementaron diferentes estrategias para su manejo en el *backend*:

- **Middleware global de manejo de errores**: Con el fin de centralizar el tratamiento de errores en el *backend* y mantener una arquitectura limpia y escalable, se implementó un *middleware* global de Express. Para ello, intercepta todas las excepciones no controladas que se propagan desde las rutas o controladores y se encarga de enviar una respuesta coherente al cliente utilizando el módulo response.js.
- **Mensajes estandarizados**: Se implementó un archivo centralizado (response.js) encargado de devolver mensajes con una estructura uniforme ante fallos del sistema. Esta estrategia facilita la interpretación de errores por parte del *frontend* y mejora la experiencia del usuario al recibir mensajes claros, independientemente del módulo o ruta donde ocurra el error.
- **Propagación controlada de errores:** En cada controlador, los errores se capturan mediante *try*/*catch* y se propagan utilizando la función *next(error)*. Posteriormente, el *middleware* global los intercepta y utiliza el módulo de respuestas para enviar el mensaje correspondiente.

5.4. Base de datos

5.4.1. Tecnologías

Para el proyecto, se optó por utilizar una base de datos SQL debido a su estructura relacional, que garantiza integridad, rendimiento y seguridad. Al ser un sistema relacional, permite estructurar los datos en tablas con relaciones claras y gracias a las restricciones como claves primarias y foráneas, se aseguró la coherencia de los datos y se evitaron redundancias.

El uso de *stored procedures* permitió optimizar el rendimiento al ejecutar código precompilado directamente en la base de datos, reduciendo la carga en la aplicación y minimizando la cantidad de consultas enviadas. Además, encapsular ciertas reglas dentro de la base de datos

ayudó a centralizar parte de la lógica de negocio, facilitando la reutilización del código y mejorando la seguridad al restringir el acceso directo a las tablas.

Por otro lado, las *functions* han sido útiles para realizar cálculos y transformaciones dentro de las consultas SQL sin afectar el rendimiento. Permitieron estructurar mejor la lógica al dividir tareas repetitivas en funciones reutilizables.

La gestión de transacciones ACID y los niveles de aislamiento han garantizado que las operaciones se ejecuten de manera segura y consistente en escenarios de concurrencia. Esto permite manejar múltiples operaciones dentro de una misma transacción, asegurando que todas se completen correctamente o se reviertan en caso de fallo.

SQL es una buena opción a largo plazo porque permitirá escalar el sistema de manera eficiente. Si el proyecto crece y surge la necesidad de manejar más información, herramientas como la partición de datos y el uso de índices *cluster* ayudarán a mantener un buen rendimiento sin comprometer la estabilidad.

5.4.2. Diagrama Entidad - Relación

En la sección “Anexo” se puede encontrar el diagrama de entidad-relación.
Además, se incorporó una copia del diagrama en formato PDF entre los entregables, con el objetivo de facilitar su visualización en mayor detalle.

5.5. Seguridad

5.5.1. Variables de entorno

En el *frontend* se utilizaron variables de entorno para dar una capa extra de protección a la información sensible, como es el caso de los usuarios y contraseñas de la base de datos. Uno de los motivos por el cuál se decidió utilizar esta técnica fue que al excluir las claves del código fuente se logró minimizar el riesgo de exposición de la información.

5.5.2. AuthGuards

Se implementaron AuthGuards para controlar el acceso a rutas antes de que se rendericen los componentes correspondientes. Este mecanismo de seguridad es brindado por Angular y consiste en implementar la interfaz CanActivate para interceptar la navegación y determinar si un usuario puede acceder a una ruta o no basándose en la lógica de autenticación. Esta implementación mejora la seguridad y garantiza que solo los usuarios autorizados puedan acceder a contenido protegido.

Se complementó el uso de AuthGuards con *JSON Web Tokens* (JWT), los cuales serán explicados en la siguiente sección. Para comprobar el acceso a una ruta, se verifica que el usuario tenga un JWT válido y se encuentre en un estado autenticado. En caso contrario, se lo redirige a la página de inicio de sesión.

5.5.3. JSON Web Tokens

En el software se utilizaron JSON Web Token (JWT) como mecanismo de autenticación y autorización. Este es un estándar de autenticación basado en *tokens* que permite la transmisión segura de información entre partes en formato JSON. Es una cadena que está codificada en Base64 que consta de tres partes:

- **Header**: Contiene el tipo de *token* y el algoritmo de firma utilizado, en este caso, HS256.
- **Payload**: Contiene la información de identificación del usuario.
- **Firma**: Es generada a partir de una clave secreta que garantiza que el *token* no ha sido alterado.

Se decidió utilizar este método porque resulta eficiente para manejar la autenticación tanto en aplicaciones web como en APIs. Entre sus principales ventajas se encuentra el hecho de que permite una autenticación sin estado. El *token* contiene toda la información necesaria del usuario, por lo que el servidor no necesita mantener sesiones en memoria ni en una base de datos. Además, al tratarse de un estándar, JWT es compatible con múltiples lenguajes de programación.

En la implementación, el *backend* genera el JWT cuando el usuario inicia sesión, enviándolo al *frontend*. Este token se almacena en el *localStorage* del navegador y se incluye en cada solicitud a rutas protegidas para verificar la identidad del usuario. Además, se incorporó un *endpoint* `/check-token`, que permite verificar su validez en cada solicitud. Este punto de verificación asegura que el JWT no haya sido manipulado ni esté expirado.

5.5.4. Encriptación

Para manejar las contraseñas de manera segura se utilizó Bcryptjs. Es un algoritmo de *hash* diseñado específicamente para proteger contraseñas almacenadas. Su característica principal y la razón por la cual se decidió aplicarlo es que incorpora un proceso de *salting*, que consiste en agregar un valor aleatorio único a la contraseña antes de cifrarla. Esto ayuda a prevenir ataques, ya que incluso si dos usuarios tienen la misma contraseña, los hashes generados serán diferentes debido al valor del *salt*. Además, bcryptjs es un algoritmo lentamente computacional, lo que lo hace resistente a ataques de fuerza bruta, ya que la generación de cada hash es intencionadamente lenta.

Gracias a esta implementación, se garantiza que las contraseñas nunca se almacenan en texto plano, reduciendo así el riesgo de exposición en caso de que la base de datos sea comprometida.

5.5.5. Permisos

En la aplicación, la gestión de accesos y permisos se realiza a través de un sistema de roles y habilidades funcionales. Cada usuario pertenece a un rol específico, y cada rol tiene permisos definidos para ejecutar ciertas acciones dentro del sistema.

Los permisos se organizan en habilidades funcionales, que representan acciones específicas que un usuario puede realizar, como crear órdenes de compra o visualizar el stock. Estas habilidades están estructuradas en un modelo de lectura y escritura, siguiendo esta convención:

write-<recurso>: Permite crear, modificar o eliminar un recurso.

read-<recurso>: Permite visualizar un recurso sin modificarlo.

5.5.5.1 Tabla de permisos

Cada rol tiene acceso a determinadas habilidades funcionales. En la sección “Anexo” se presenta la tabla que corresponde a su asignación.

5.5.5.2 Implementación de permisos

La lógica de control de acceso en el *frontend* se implementó utilizando directivas estructurales de Angular, como por ejemplo *ngIf. Esta estrategia permite mostrar u ocultar componentes de la interfaz de usuario según las habilidades funcionales que recibe el cliente al momento de autenticarse.

Al iniciar sesión, el *backend* devuelve un *token* que contiene, entre otros datos, las habilidades asignadas al usuario. Son almacenadas temporalmente en el *frontend* y utilizadas para determinar qué acciones están habilitadas en la interfaz. Por ejemplo, un botón de "Crear orden de compra" sólo será visible si el usuario cuenta con la habilidad *write-oc*.

Este enfoque permite una interfaz dinámica y segura, en la que cada usuario solo puede acceder a las funcionalidades que le han sido asignadas, evitando accesos indebidos desde el cliente.

Además del control visual, todas las acciones sensibles (creación, edición o eliminación de recursos) también están protegidas en el *backend*, por lo que no es posible realizar operaciones restringidas manipulando directamente el cliente.

5.5.6. Logs

Toda acción realizada por los usuarios es registrada con su correspondiente identificación. Esto garantiza trazabilidad completa, ya que no es posible suplantar la identidad de otro usuario para ejecutar acciones. El sistema mantiene un historial claro y verificable de cada operación ejecutada. De esta manera se refuerza la transparencia y la auditoría del sistema.

Cada log incluye el *endpoint* accedido, el método HTTP utilizado, el identificador del usuario, los parámetros enviados, fecha, y cualquier resultado o error producido durante la ejecución.

En la aplicación cliente, se implementó un interceptor utilizando la librería HttpInterceptor que permite capturar todas las peticiones HTTP realizadas por el usuario. Esta implementación asegura que cada interacción con el sistema quede documentada de forma centralizada.

Del lado del *backend*, cada solicitud recibida también es procesada para registrar las acciones del usuario. Este sistema de registro se implementó de manera transversal utilizando *middlewares*. La información recolectada se almacena en una tabla propia.

5.6. Testing

5.6.1. Frontend

Para las pruebas en el frontend, se recurrió al uso de las devTools de Google Chrome y se llevaron a cabo pruebas de funcionalidad cada vez que se implementaba un nuevo módulo. Además, cada componente está asociado a un archivo ".spec.ts" para ejecutar pruebas unitarias utilizando Karma.js como test runner, en conjunto con Jasmine, que actúa como framework de pruebas para definir y estructurar los distintos casos de prueba. Es decir, Karma se encarga de correr los tests automáticamente en un navegador real, mientras que Jasmine facilita la escritura de los distintos tests.

Se eligió esta combinación no solo porque viene integrada por defecto en proyectos Angular y cuenta con buena documentación, sino también porque el equipo ya contaba con experiencia previa utilizándolas.

5.6.2. Backend

Para garantizar la calidad y robustez del backend desarrollado en Express.js, se implementó una estrategia de pruebas dividida en tres niveles: pruebas unitarias, pruebas de integración y pruebas end-to-end.

Pruebas Unitarias

Estas pruebas tienen como objetivo validar de forma aislada la lógica implementada en los métodos del repositorio, es decir, las funciones que interactúan directamente con la base de datos. En estas pruebas se utilizan mocks para simular el comportamiento del acceso a datos

sin necesidad de una base real. La herramienta utilizada en este caso fue Jest (con mocks de las dependencias, como la conexión a MySQL)

Pruebas de Integración

Su función es verificar que la lógica del controlador funcione correctamente al recibir una petición HTTP, validarla, interactuar con el repositorio y enviar una respuesta adecuada. Estas pruebas se realizan sobre la aplicación Express, sin simular rutas ni lógica. Para realizar estas pruebas Jest y Supertest (para simular peticiones HTTP).

Pruebas End-to-End

Se realizaron pruebas para verificar que todas las partes del backend (rutas, controladores, lógica de negocio y base de datos) funcionaran correctamente en conjunto, simulando el uso real del sistema. Para esto, se utilizó Postman como herramienta principal, ya que todos los integrantes del equipo ya la conocían y su uso resultaba práctico. Postman permitió organizar las pruebas en colecciones por módulo, lo que facilitó la ejecución de distintos casos, incluyendo operaciones como GET, POST, PUT y DELETE, con validaciones tanto para respuestas exitosas como para errores.

Gracias a este enfoque, se aseguró que las distintas partes del sistema (base de datos, lógica de negocio, rutas) funcionen juntas. Esto garantizó que los servicios funcionaran correctamente antes de su implementación en producción.

5.7. Documentación

La documentación cumple un rol clave en el desarrollo de aplicaciones que buscan ser escalables, ya que actúa como una guía esencial para entender cómo está implementado el sistema. Esto facilita tanto su mantenimiento como futuras modificaciones.

En el proyecto, los métodos utilizados en el *frontend* están comentados directamente en el código fuente, proporcionando una referencia rápida y accesible para quienes lo desarrollan o deban intervenirlo más adelante.

En el caso de las APIs, se integró la herramienta SwaggerJS, que —como se mencionó en la sección “Herramientas utilizadas”— es una librería de terceros que permite documentar en detalle el *backend* a través de anotaciones en el código. Esta herramienta genera automáticamente una interfaz gráfica navegable, alojada en el mismo servidor donde corre la API. Se adjuntó una copia de la documentación en los entregables, con el fin de poder acceder sin conexión en el caso de requerirse.

Además, se incluyeron distintos diagramas que ayudan a comprender la arquitectura y el flujo general del sistema, tales como diagramas de componentes, diagramas de secuencia, entre otros. Todos ellos están disponibles en la sección “Anexo”.

5.8. Deploys

Para la implementación del sistema, se utilizó Docker junto con Docker Compose, una herramienta que permite definir y gestionar múltiples contenedores de manera sencilla. Se optó por esta opción, ya que facilita el despliegue de la aplicación, permitiendo ejecutar tanto el backend como la base de datos en contenedores separados en el caso de ser necesario.

Cabe destacar que, si bien el sistema fue validado funcionalmente por los referentes de ambas empresas, ninguna de las adaptaciones del software se encuentra actualmente en producción.

Se ofrecieron distintas alternativas de despliegue, incluyendo la posibilidad de hospedar el sistema en una máquina virtual proporcionada por una empresa en Mar del Plata o realizar el deploy en la nube utilizando AWS (Amazon Web Services). Sin embargo, en el caso de ViaVeg, la empresa decidió postergar el despliegue debido a su situación económica. Por otro lado, Coppens aún no ha comunicado cuál será su infraestructura elegida para el despliegue, por lo que tampoco se ha avanzado en su puesta en marcha.

A futuro, el sistema puede ser desplegado con relativa facilidad gracias a su arquitectura (detallada en la sección “Arquitectura”) y el uso de contenedores que permiten replicar el entorno de desarrollo en distintos servidores sin necesidad de configuraciones complejas, mejorando la portabilidad de la aplicación.

Tiempo estimado de despliegue

Se contará con 2 semanas para la puesta en producción de cada software.

El despliegue del sistema se realizará según la elección del cliente, ya sea en un entorno de nube privada local —como una máquina virtual provista por una empresa marplatense especializada en servicios de infraestructura— o en una plataforma de nube pública, como AWS. En ambos casos, será necesario realizar configuraciones adicionales, que incluyen el aprovisionamiento de recursos, la definición de variables de entorno, reglas de red, políticas de seguridad, mecanismos de persistencia de datos y pruebas de accesibilidad desde el exterior.

5.9. Docker

Docker es una plataforma que permite crear, distribuir y ejecutar aplicaciones de manera aislada dentro de contenedores. Un contenedor es un entorno de ejecución aislado que encapsula una aplicación junto con todas sus dependencias, bibliotecas y configuraciones necesarias para su correcto funcionamiento. A diferencia de las máquinas virtuales, los contenedores comparten el mismo sistema operativo, lo que los hace más eficientes y rápidos.

Se optó por utilizar esta herramienta para evitar conflictos durante el desarrollo por la ejecución del software en distintos entornos, y para facilitar el deploy del sistema.

En el desarrollo del proyecto, Docker se utilizó para garantizar que la aplicación se ejecutará de manera consistente en diferentes entornos, evitando problemas de compatibilidad entre sistemas operativos o versiones de software.

Además, permitió contener los servicios auxiliares esenciales, como la base de datos MySQL, asegurando su correcta configuración y ejecución dentro del mismo entorno de desarrollo y producción. Mediante Docker Compose, se definieron y orquestaron estos servicios, lo que simplificó la administración de los contenedores y su comunicación dentro de una misma red virtual.

La compatibilidad de Docker con herramientas de orquestación como Kubernetes abre la posibilidad de una futura expansión del proyecto. En caso de necesitar mayor escalabilidad, se podría migrar a una infraestructura basada en Kubernetes para distribuir la carga de trabajo,

gestionar múltiples instancias de los servicios y adaptarse a mayores demandas de procesamiento y tráfico de manera eficiente.

Gracias a esta implementación, el desarrollo y despliegue del sistema fueron más ágiles, reduciendo problemas de compatibilidad y mejorando la reproducibilidad del entorno en distintas etapas del proyecto.

5.9.1. Arquitectura de Docker

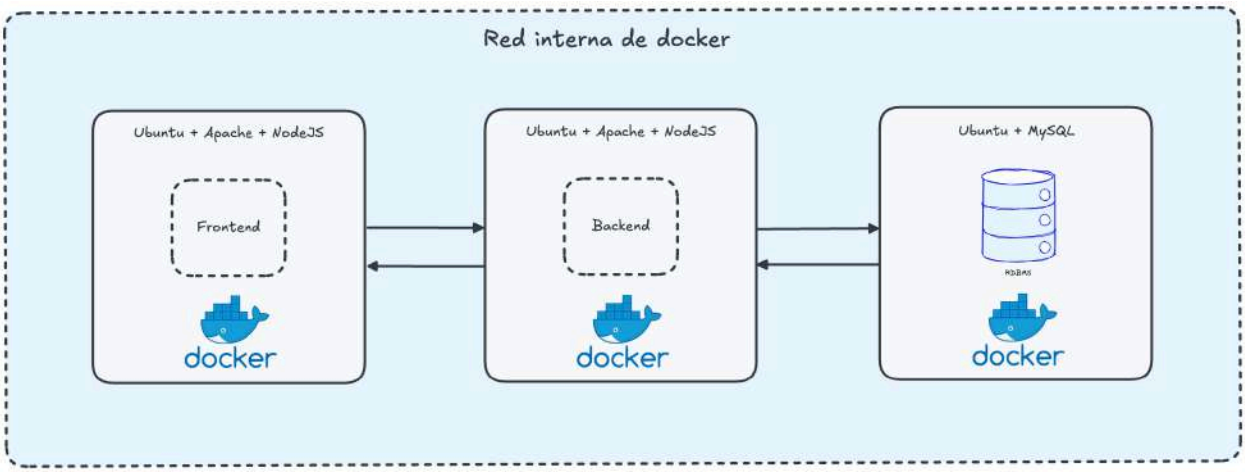


Fig 8. Diagrama de arquitectura de Docker

Como se muestra en la arquitectura, se configuraron contenedores independientes basados en imágenes de Node.js para el frontend y el backend, cada uno con sus respectivas dependencias instaladas. El backend se expuso en el puerto 3000, mientras que el frontend lo hizo en el puerto 4200. A su vez, la base de datos MySQL fue desplegada en un contenedor separado, con credenciales y configuración predefinidas, asegurando la persistencia de los datos incluso tras reinicios del servicio. Esto permitió que el entorno de desarrollo y producción fueran idénticos, reduciendo posibles errores por diferencias en la configuración.

El uso de Docker Compose permitió automatizar la orquestación de los servicios, definiendo las dependencias entre ellos y asegurando que la base de datos estuviera disponible antes de iniciar la aplicación. Además, gracias a los volúmenes, los datos de la base de datos se mantuvieron a través de los reinicios, evitando la pérdida de información.

Los dockerfiles se incluyeron dentro de una carpeta en los entregables del proyecto.

6. Producto

6.1. Producto obtenido

En la sección "Anexo" se pueden encontrar capturas de pantalla de las vistas más representativas del sistema.

6.1.1 Módulos generales (*core*)

ID	Nombre	Alcance
M-01	Autenticación	- Gestión de permisos y roles - Seguridad y control de accesos
M-02	Usuarios	- Registro y gestión de usuarios
M-03	Proveedores	- Registro y gestión de proveedores - Historial de compras y relación comercial - Relación con insumos
M-04	Insumos	- Alta, baja y modificación de insumos - Asociación de insumos a productos
M-05	Productos	- Alta, baja y modificación de producto - Definición de costos asociados
M-06	Órdenes de compra	- Gestión de orden de compra
M-07	Remitos	- Gestión de remitos y recepción de insumos
M-08	Ventas	- Alta, baja y modificación de ventas
M-09	Stock	- Gestión de stock - Stock mínimo - Ubicación en depósitos o almacenes

M-10	Depósitos	- Alta, baja y modificación de depósitos.
M-11	Producción	- Creación y gestión de órdenes de producción. - Asignación de insumos a la producción. - Seguimiento del estado de producción. - Manejo del sistema por parte del trabajador de planta.
M-12	Fórmulas	- Alta, baja y modificación de fórmulas. - Asignación de insumos y cantidades. - Asociación con productos y costos de producción.
M-13	Trazabilidad	- Registro de movimientos de productos e insumos. - Generación de reportes de movimientos.
M-14	Logs	- Registro de logs.

6.1.2 Módulos específicos para ViaVeg

ID	**Nombre**	**Alcance**
M-01	Lotes	- Asignación de códigos para lotes de acuerdo a la lógica interna de la empresa (6 dígitos donde los 2 primeros definen el producto/insumo). - Alta, baja y modificación de lotes.
M-02	Categorías	- Clasificación por categorías (servicio o producto).

M-03	Resultados	- Realizar reportes de :
		  - Producción efectiva
		  - Merma

6.1.3 Módulos específicos para Coppens

ID	Nombre	Alcance
M-01	Calidad	- Control de calidad en recepción del producto

6.2 Trabajos futuros

Despliegue y capacitaciones

Si bien la aplicación ha sido validada funcionalmente con los usuarios finales a través de diversas instancias de prueba y *feedback*, aún no ha sido desplegada en un entorno productivo debido a factores externos. Una vez superados estos obstáculos y concretado el despliegue, se llevará a cabo un plan de capacitación orientado a los distintos perfiles de usuarios involucrados.

La validación previa por parte de los clientes sugiere que los flujos de trabajo de toda la empresa han sido considerados. De todos modos, el equipo consideró que la etapa de capacitación es clave para una correcta puesta en marcha del sistema, por lo que se planificaron sesiones de demostración de los distintos flujos, seguidas de instancias d*e Q&A* orientadas a los diferentes perfiles de usuarios.

Optimización de la Trazabilidad mediante Códigos QR

Actualmente, la trazabilidad se gestiona a través de números de lote internos definidos por cada empresa. Sin embargo, una posible mejora sería la adopción de códigos QR, lo que permitiría un acceso más ágil y preciso a la información de los productos y materias primas en los distintos depósitos. Esta implementación optimizaría la gestión de inventarios, reduciendo tiempos de búsqueda y minimizando errores en la identificación de lotes.

6.2.1 Posibilidad de mejora

Implementación de Facturación

Una posible evolución del sistema podría incluir la gestión de facturación, permitiendo a la empresa centralizar sus procesos administrativos y mejorar la eficiencia operativa. Aunque esta funcionalidad no es prioritaria en la actualidad, su integración en el futuro representaría un valor agregado significativo, optimizando la administración financiera y reduciendo la necesidad de herramientas externas.

Participación de Proveedores y Distribuidores

El diseño basado en una aplicación web ofrece la flexibilidad de integrar a los diferentes actores de la cadena de suministro, incluyendo proveedores y distribuidores. La adopción de esta plataforma por parte de estos actores permitiría una mayor automatización de las tareas de compra y distribución, mejorando la trazabilidad y el control de los procesos logísticos. Su implementación dependerá de la aceptación y adaptación por parte de las empresas involucradas, pero representa una oportunidad clave para fortalecer la eficiencia y transparencia del sistema.

Además, la integración con proveedores podría incorporar un sistema dinámico de consulta de precios, evitando la necesidad de solicitar cotizaciones de forma manual y permitiendo el acceso en tiempo real a valores actualizados de insumos.

Por otro lado, la plataforma podría ofrecer funcionalidades avanzadas para la programación y coordinación de entregas de insumos, asegurando un flujo continuo de abastecimiento y evitando interrupciones en la producción.

Blockchain

Otra posible línea de mejora para este proyecto es la incorporación de tecnología blockchain con el objetivo de sumar transparencia en la cadena de suministro. Al registrar los eventos del circuito productivo en una base de datos descentralizada e inmutable, se facilitaría el acceso público a la trazabilidad del producto, permitiendo a consumidores verificar su procedencia. Esta integración no solo aportaría confianza y valor agregado, sino que ayudaría a las

empresas a cumplir con estándares cada vez más demandados en industrias como la alimenticia. El potencial de aplicar esta tecnología a la trazabilidad se puede observar en empresas como Carrefour, que la implementa para certificar el origen de productos como pollo orgánico, leche, o vinos premium.

8. Memoria del proyecto

8.1. Ejecución

El proyecto se inició a partir de la demanda de un primer cliente, ViaVeg, quien manifestó la necesidad de contar con un software para mejorar la trazabilidad y gestión de su producción por fasón. Posteriormente, surgió un segundo cliente, Coppens, con requerimientos similares. Frente a esta nueva oportunidad, y con la hipótesis de que era posible desarrollar un sistema genérico adaptable a múltiples industrias que producen por fasón, se comenzó la etapa de elicitación y análisis de requerimientos de Coppens con el objetivo de extraer las características comunes entre ambas empresas y utilizarlas como la base de un futuro sistema genérico.

Sin embargo, cuando el desarrollo de la solución para ViaVeg ya estaba relativamente avanzado, la cátedra recomendó cambiar el enfoque del proyecto. A partir del análisis de un producto genérico y las necesidades comunes de la producción tercerizada, se lograría construir una base reutilizable. Esto implicó dejar de priorizar el análisis específico de Coppens.

A su vez, al pausar el desarrollo individual de ambos sistemas se planificó continuarlo posteriormente como módulos integrados al producto genérico, adaptados a las funcionalidades core.

Este cambio implicó rediseñar parte de la arquitectura y funcionalidades para que fueran más flexibles. Si bien hubo etapas de re-trabajo, la desviación en los plazos de entrega para ambos sistemas no fue significativa. No obstante, sí implicó un mayor esfuerzo en términos de horas de trabajo, lo que representó un incremento en los costos del proyecto.

Adaptarse a este nuevo contexto demandó un análisis más profundo y una curva de aprendizaje más pronunciada por parte del equipo, especialmente debido a la incorporación de

una nueva arquitectura. Esto requirió no solo rediseñar partes del sistema, sino también adquirir conocimientos técnicos adicionales de dicha arquitectura, lo que incrementó el tiempo de dedicación para continuar con el desarrollo.

Sin embargo, el impacto de este cambio fue mitigado en parte gracias a que, desde una etapa temprana, el análisis y desarrollo para ViaVeg ya se había abordado con una estructura genérica en mente. Si bien inicialmente no se contemplaban los requerimientos específicos de otra industria como la de Coppens, esa decisión facilitó la adaptación posterior y redujo el esfuerzo que habría implicado comenzar desde cero.

No existieron diferencias significativas entre las necesidades iniciales del primer cliente y las contempladas en el producto genérico. El cambio de enfoque propuesto por la cátedra sí fue necesario y resultó beneficioso ya que generó un producto con mayor potencial de crecimiento y aplicabilidad a diversas industrias, con una visión a largo plazo. Además, al desarrollar una solución con una arquitectura más modular, se mejoró significativamente la mantenibilidad del sistema, facilitando futuras modificaciones, extensiones y adaptaciones a nuevos contextos.

Si bien el sistema se encuentra completamente desarrollado y ha sido validado funcionalmente, no se realizó su despliegue en un entorno productivo como se había planificado. Esta decisión estuvo motivada por razones relacionadas con los clientes, previamente mencionadas en la sección de Deploys. Cabe aclarar que, si bien se realizaron validaciones funcionales durante el proceso de desarrollo, aún resta llevar a cabo una validación funcional exhaustiva en un entorno real de uso, que permita evaluar el comportamiento del sistema bajo condiciones operativas concretas.

Durante el avance del desarrollo, el equipo detectó que la planificación inicial no se ajustaba correctamente al nuevo enfoque del proyecto ni a los cambios constantes en los requerimientos de los clientes. Esta situación motivó una revisión y ajuste en la estrategia de planificación. Por ello, se realizó una nueva estimación de tiempos, que quedó reflejada en un segundo diagrama de Gantt (ver Figura 9).

La re-planificación implicó una redistribución de las etapas del proyecto, que permitiera incorporar cambios sin comprometer la coherencia del sistema. Esto dio lugar a las siguientes etapas definidas en el nuevo cronograma:

1. Análisis ViaVeg
2. Diseño ViaVeg
3. Desarrollo ViaVeg
4. Validación ViaVeg
5. Testing ViaVeg
6. Deploy ViaVeg
7. Capacitación ViaVeg
8. Análisis Coppens
9. Análisis genérico
10. Diseño genérico
11. Desarrollo genérico
12. Desarrollo módulo ViaVeg
13. Desarrollo módulo Coppens

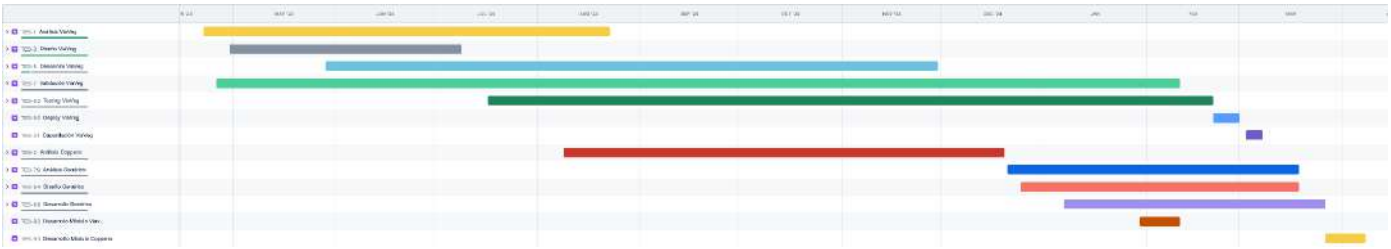


Fig 9. Diagrama de Gantt actualizado

Cabe mencionar que, si bien se planificó realizar una etapa de capacitación y despliegue para el cliente Coppens, esta no fue contemplada explícitamente en el cronograma ni reflejada en el diagrama de Gantt, debido a un error de omisión. Al tratarse de una tarea no presupuestada, se prevé que su ejecución implique un costo adicional no previsto, lo que encarecería el producto y reduciría la utilidad original.

8.1.1 Dificultades enfrentadas

Durante el desarrollo del proyecto, se presentaron diversas problemáticas que requirieron ajustes en la organización y en la implementación técnica:

1. **Distribución de tareas y toma de decisiones**:
   Inicialmente, todos los miembros del equipo cumplián el rol de desarrolladores *FullStack*, lo que generaba desorganización y demoraba el avance del proyecto. Dado que el equipo estaba compuesto por cuatro personas, la toma de decisiones se volvió aún más compleja, ya que cada cambio requería consenso y coordinación entre todos.

   Para optimizar el flujo de trabajo, se decidió dividir al grupo en dos equipos: *frontend* y *backend*. Los roles se asignaron de acuerdo con la experiencia previa y las preferencias de cada integrante. Esta reorganización permitió una mejor distribución de tareas, facilitó la toma de decisiones en cada área y mejoró notablemente la organización general del proyecto, lo que permitió avanzar a un ritmo mucho más ágil.

2. **Adaptación de vistas para operarios de planta**
   Otro desafío fue la necesidad de diseñar ciertas interfaces para dispositivos móviles, ya que serían utilizadas por operarios de planta. Esto implicaba considerar las condiciones del entorno de trabajo, donde la practicidad y rapidez en la interacción con la aplicación eran fundamentales.

   Por ello, se implementó una interfaz más accesible, incluyendo botones grandes, textos claros y una disposición optimizada para su uso en pantallas táctiles, con el fin de garantizar una mejor experiencia de usuario en un entorno de producción real.

3. **Problemas de compatibilidad con bcrypt en Docker**
   Al ejecutar el backend dentro de un contenedor Docker, surgió un error relacionado con la carga de la librería bcrypt: "*Error: bcrypt library failed to load***".**

   Uno de los motivos fue que la librería bcrypt se había instalado en la máquina host con una arquitectura diferente a la del contenedor Docker, lo que generaba incompatibilidades al ejecutar la aplicación. Además, bcrypt requiere compilaciones nativas específicas para cada entorno, lo que dificulta su uso dentro de Docker.

Para evitar problemas de compilación y compatibilidad, se optó por reemplazar bcrypt con bcryptjs, una alternativa completamente implementada en JavaScript que no requiere compilaciones nativas. Este cambio logró una compatibilidad total con Docker junto a una instalación y ejecución sencilla, sin necesidad de configuraciones adicionales. Cabe destacar que al tener la misma lógica que bcrypt, la migración no requirió de cambios significativos de código.

4. **Falta de disponibilidad de tiempo por compromisos laborales**

   Al encontrarse todos los miembros empleados de forma activa, resultó difícil coordinar los tiempos del proyecto con los trabajos personales de cada integrante. Esta situación redujo significativamente el tiempo efectivo de trabajo en distintas etapas, lo que afectó tanto el ritmo de avance como la organización general del equipo.

   Para mitigar este problema, se intentó organizar reuniones de seguimiento más puntuales y aprovechar los fines de semana o feriados para recuperar tiempo de trabajo en equipo. A pesar de las limitaciones, se logró mantener una buena comunicación interna y avanzar con el desarrollo.

5. **Cambio de enfoque**

   El mayor desafío enfrentado a nivel técnico y organizativo fue el cambio de enfoque del proyecto mencionado en la sección “Ejecución”. Esta transformación implicó redefinir la arquitectura, reorganizar prioridades y adaptar lo ya desarrollado.

   Si bien este cambio generó re-trabajo y pausas en el desarrollo original de los sistemas individuales, permitió construir una base mucho más flexible, escalable y mantenible.

6. **Falta de conocimiento sobre el dominio y limitaciones externas**

   Otro aspecto desafiante fue la falta de conocimiento inicial sobre los procesos industriales específicos de cada cliente. Esto obligó a los integrantes a realizar varias iteraciones junto a los referentes técnicos y operativos de cada empresa para comprender en profundidad sus requerimientos y restricciones.

   Además, se presentaron factores externos que condicionaron el ritmo del proyecto. Por ejemplo, ViaVeg atravesó una situación económica difícil que provocó que el proyecto

fuera situado en un segundo plano. Esto impactó directamente en la planificación, ya que en varias ocasiones se debieron reprogramar reuniones o tomar decisiones con información incompleta.

8.1.2. Desviaciones en las tareas

Nº	Tarea	Tiempo Estimado (hs)	Tiempo Ejecutado (hs)	Desviación (hs)
1	Elicitación de requerimientos del sistema	30	40	10
2	Análisis del sistema	170	195	25
3	Diseño de las interfaces del sistema	60	70	10
4	Diseño de la Arquitectura del Sistema	30	40	10
5	Diseño de la base de datos	80	90	10
6	Armado del ambiente de desarrollo	8	6	-2
7	Desarrollo frontend	125	110	-15
8	Desarrollo backend	125	175	50
9	Desarrollo de base de datos	50	40	-10
10	Desarrollo de módulos específicos	145	150	5
11	Testing	50	70	20
12	Documentación del proyecto	60	160	100
13	Despliegue del sistema	0	Por implementar	30(*)
14	Capacitaciones a los usuarios	0	Por implementar	20(*)
	Total	933	1146	263

*Las 50 horas correspondientes al despliegue y las capacitaciones no habían sido contempladas en la planificación inicial, pero fueron incorporadas posteriormente al detectarse su omisión. Si bien estas actividades aún no fueron ejecutadas, se consideran dentro del total de horas previstas, lo que genera una desviación respecto al cronograma original.

Cabe aclarar que, en el siguiente gráfico comparativo entre el tiempo estimado y el tiempo efectivamente ejecutado, estas tareas no se encuentran representadas debido a la omisión mencionada previamente.

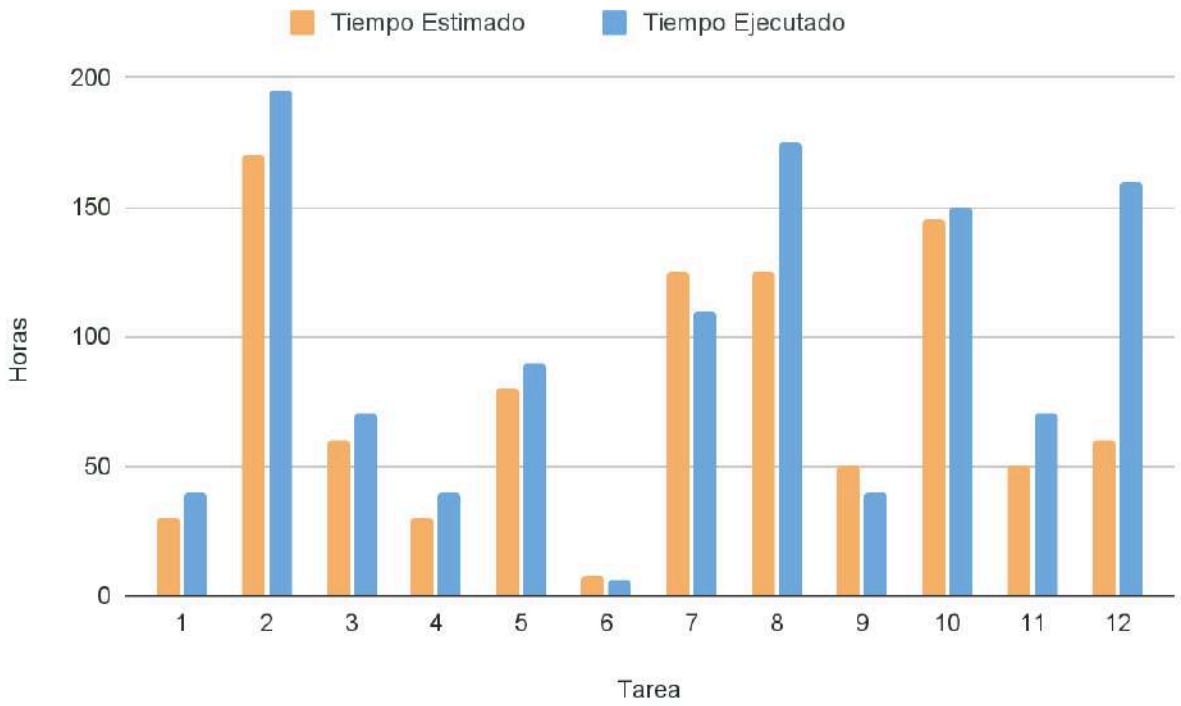


Fig 10. Gráfico de tiempo estimado contra ejecutado

En base al análisis del gráfico de duración de las tareas, se han identificado varios factores que explican la desviación de 263 horas. En primer lugar, la inexperiencia del equipo en la realización de estimaciones para este tipo de proyectos ha sido un factor significativo. El equipo carecía de práctica previa en proyectos reales, lo que limitó su capacidad para estimar con precisión la duración de las etapas. Además, la presencia de numerosas tareas desconocidas generó incertidumbre en cuanto a su dificultad y complejidad.

Por otro lado, las etapas de análisis, diseño y desarrollo se vieron impactadas por los obstáculos detallados en la sección “Dificultades enfrentadas”. Cada modificación implicaba retomar el proceso completo de revisión, lo que extendió considerablemente los tiempos previstos.

El tiempo adicional invertido en la etapa de elicitación de requerimientos del sistema se relaciona con la necesidad de reconstruir y formalizar procesos que no estaban claramente definidos por los clientes. Si bien se habían estimado 30 horas, fue necesario realizar validaciones adicionales con el director del proyecto, para poder entender en profundidad los circuitos operativos y asegurar que los requerimientos recogidos fueran precisos y completos.

En cuanto al diseño de interfaces del sistema, durante la implementación, surgieron cambios en los flujos operativos y nuevas sugerencias por parte de los clientes, lo que obligó al equipo a replantear parcialmente las interfaces. Este rediseño generó una extensión en el tiempo destinado a esta etapa, superando la estimación original.

El tiempo destinado al diseño de la arquitectura fue mayor al inicialmente estimado, principalmente debido a la decisión de adoptar una arquitectura monolítica modular en una etapa tardía. Esta elección implicó un proceso de adaptación que requirió reformular parte de la estructura del sistema y, además, una instancia de capacitación interna, ya que el equipo debió instruirse en los principios y buenas prácticas asociados a este enfoque arquitectónico. Como resultado, esta etapa demandó un esfuerzo adicional tanto en términos de análisis como de implementación.

El diseño de la base de datos fue ajustado en distintas oportunidades debido a modificaciones en los requerimientos y la inclusión de nuevas entidades, lo que generó un trabajo adicional no previsto inicialmente.

La mayor desviación en los tiempos estimados se presentó en el desarrollo del backend. Esto puede explicarse principalmente por la falta de experiencia práctica con tecnologías como Docker; si bien se contaba con conocimientos teóricos, no se había presentado la oportunidad de aplicarlos previamente. Además, la reestructuración de la arquitectura implicó la necesidad de realizar ajustes en el código para adaptarlo al nuevo enfoque del proyecto, lo que también contribuyó a un aumento en el tiempo de ejecución de la tarea.

También se observa que las tareas relacionadas con el desarrollo frontend demandaron menos tiempo del estimado inicialmente. Esta diferencia se debe a que, al momento de su implementación, ya se había alcanzado un grado avanzado en la curva de aprendizaje de la tecnología utilizada, lo que permitió trabajar con mayor eficiencia.

La desviación menor que se presentó en el desarrollo de módulos específicos se debe principalmente al tiempo necesario para terminar de integrar correctamente los módulos particulares de cada cliente con el sistema genérico.

Las pruebas del sistema requirieron más tiempo de lo estimado, ya que fue necesario realizar iteraciones adicionales para validar los cambios incorporados a lo largo del desarrollo. Asimismo, algunos errores detectados durante las pruebas implicaron re-trabajos que no se habían contemplado inicialmente.

Cabe mencionar que en la planificación inicial, no se contempló la elaboración del informe final, sino únicamente la documentación técnica del sistema. Esta omisión contribuyó a subestimar el esfuerzo requerido para esta etapa.

Como consecuencia, la redacción de la documentación también requirió más tiempo del previsto, debido a diversas complejidades. Por un lado, fue un desafío coordinar la escritura entre los cuatro integrantes del equipo, lo que requirió tiempo adicional para unificar criterios y asegurar coherencia en el estilo y el contenido. Por otro lado, la necesidad de revisar y ajustar reiteradamente los textos, en función del avance del desarrollo y de los cambios en el enfoque del proyecto, también implicó un trabajo más extenso. Asimismo, se cuidó especialmente la presentación del informe, lo que demandó sucesivas instancias de revisión y mejora.

Hubo algunos tiempos de inactividad en los cuales no se realizaron tareas debido a que el cliente, por motivos internos, postergaba las reuniones pactadas, retrasando así el desarrollo del sistema. Si bien estas pausas no afectaron los plazos de entrega, sí implicaron un aumento en la cantidad total de horas trabajadas, lo que se reflejó en un mayor costo del proyecto. Esto se compensó con una mayor concentración de esfuerzo en los momentos activos del desarrollo.

A pesar de estos desfasajes, el equipo logró cumplir con los plazos que estaban a su alcance gracias a un esfuerzo sostenido y a una constante capacidad de adaptación. Esto permitió

alcanzar un desarrollo funcional sólido y validado, aunque aún pendiente de despliegue y capacitación.

8.2. Análisis de las etapas

8.2.1. Análisis

Como se mencionó en la sección anterior, el proyecto inició con la idea de crear un sistema específico para una empresa en particular, enfocándose en sus procesos internos y en las necesidades puntuales que surgían de su forma de trabajo. Durante esta primera etapa, se llevaron a cabo reuniones con el referente de la empresa para relevar información, identificar los flujos de trabajo más relevantes, y detectar las principales problemáticas a resolver.

A partir del cambio de enfoque, se inició una nueva etapa de elicitación con el nuevo cliente, pero esta vez centrándose en el análisis del producto genérico. Aunque ambas empresas compartían una metodología de procesos similar, ésta aportaba particularidades funcionales que pusieron en evidencia la necesidad de replantear el alcance de la solución. De esta manera, el proyecto evolucionó de una implementación puntual hacia una versión más flexible y genérica, capaz de adaptarse a distintos tipos de negocios que trabajen bajo un modelo de producción por fasón.

Este nuevo relevamiento permitió comparar y contrastar los requerimientos de ambas empresas, distinguiendo aquellos que eran comunes, de aquellos que eran específicos de cada caso. Esta etapa fue clave para ajustar el diseño del sistema, ya que permitió validar qué funcionalidades podrían ser reutilizables y generalizadas y cuáles requerían un enfoque más parametrizable o configurable según cada empresa.

8.2.2. Diseño

En la etapa inicial del diseño de la arquitectura del *backend*, la principal preocupación fue lograr una modularización adecuada del código. Esto permitió separar las distintas funcionalidades del sistema en unidades bien definidas, y por ende, facilitar la organización del proyecto, la mantenibilidad del código y la colaboración entre distintos miembros del equipo.

A medida que el sistema fue creciendo, se descubrió que se podía aprovechar aún más esta estructura modular integrándola dentro de una arquitectura monolítica modular. Este enfoque permitió mantener todos los módulos dentro de una misma base de código y un mismo proceso de despliegue, lo que simplifica la gestión general del sistema y reduce la complejidad operativa.

La evolución hacia una arquitectura monolítica modular permitió combinar la simplicidad de un sistema monolítico con los beneficios estructurales de la modularización. De esta manera, se logró una base sólida para continuar desarrollando el sistema de forma ordenada y eficiente, que asegure su escalabilidad y capacidad de adaptación frente a futuras necesidades.

8.2.3. Desarrollo

La fase de desarrollo fue, sin dudas, la más extensa e intensa del proyecto. Uno de los principales desafíos técnicos fue la integración con la base de datos y el uso de *stored procedures*, lo que requirió un trabajo cuidadoso para asegurar que la lógica del negocio se aplicara correctamente. Además, se utilizó Docker para facilitar la configuración y despliegue del entorno, lo cual implicó una pronunciada curva de aprendizaje inicial.

A medida que el proyecto fue evolucionando, se hizo necesario refactorizar partes del código para adaptarse a cambios en la arquitectura general y para mejorar la mantenibilidad del software. Estos ajustes implicaron revisar decisiones previas y repensar cómo organizar ciertas funcionalidades.

Desde el aspecto organizativo, la coordinación entre el equipo de *frontend* y el de *backend* no siempre fue sencilla. Si bien se trabajó de manera colaborativa, en varias ocasiones se presentaron dependencias entre partes del sistema que requerían ajustes de alguno de los dos lados, y no siempre se contaba con la disponibilidad para resolverlas rápidamente.

Por momentos, fue un desafío alinear prioridades y mantener una comunicación fluida, sobre todo cuando cada integrante tenía diferentes horarios o compromisos personales y académicos. Sin embargo, con el correr del proyecto se fue puliendo la dinámica de trabajo, aprendiendo a anticipar mejor las necesidades del otro y logrando encontrar una forma de

organizar al equipo que permitiera avanzar de manera constante. Se pueden encontrar más detalles de las metodologías aplicadas en la sección “Metodologías”.

Además, a lo largo de esta fase se fueron incorporando buenas prácticas como la delegación de tareas según las fortalezas de cada integrante y la delimitación clara del alcance de cada una de ellas. Esta forma de organización ayudó a priorizar mejor, distribuir la carga de trabajo de manera más equitativa y evitar solapamientos en el desarrollo.

8.2.4. Documentación del trabajo final

La última fase del proyecto correspondió a la elaboración del informe final, que sintetiza todos los aspectos clave del trabajo realizado. Además de presentar el proceso de elicitación, análisis, diseño y desarrollo, incluye las decisiones técnicas tomadas y los resultados obtenidos, así como también documentación adicional que respalda cada etapa atravesada.

Afortunadamente, se llevó un registro constante de las actividades diarias a lo largo del proyecto, lo que facilitó su creación, asegurando que no se pasara por alto ningún detalle importante y que todo estuviera bien organizado.

En definitiva, gracias a una documentación sostenida en el tiempo, se logró un informe final completo y bien estructurado, que no solo refleja lo realizado, sino que también funcionará como base de consulta para futuras referencias.

8.3. Cumplimiento y evolución de los objetivos

A lo largo del desarrollo del proyecto, se logró avanzar significativamente en el cumplimiento del objetivo general propuesto: diseñar y construir un software genérico para la gestión de la producción por fasón, adaptable a distintos rubros industriales. El sistema cuenta actualmente con módulos fundamentales como la gestión de órdenes de producción, control de stock, trazabilidad de insumos, y administración de depósitos, lo cual demuestra un cumplimiento efectivo del núcleo funcional del sistema.

En cuanto a los objetivos específicos detallados en el alcance inicial, se lograron cubrir funcionalidades clave, tales como:

- Trazabilidad completa de insumos y productos a lo largo de la cadena de valor.
- Asignación de recursos a las órdenes de producción, incluyendo fórmulas y cantidades.
- Gestión de proveedores y control logístico de insumos.
- Registro de ingresos y egresos de materiales.
- Seguimiento de lotes en cada etapa del proceso productivo.

Durante el proceso, algunos objetivos evolucionaron a partir del *feedback* de usuarios y expertos del rubro, lo cual resultó en mejoras en la interfaz de usuario, simplificación de ciertos flujos y priorización de funcionalidades de trazabilidad. También se integraron prácticas modernas de desarrollo como el uso de APIs RESTful y arquitectura cliente-servidor desacoplada.

Si bien no se abordaron los aspectos tributarios, fiscales y contables —tal como se estableció en los límites del alcance—, el sistema quedó preparado para futuras extensiones en caso de que estas funcionalidades se requieran.

En conclusión, los objetivos iniciales fueron cumplidos, y aquellos que evolucionaron durante el desarrollo lo hicieron en favor de un producto más robusto, flexible y alineado con las necesidades reales de las empresas del sector.

8.4. Próximos pasos

Tal como se mencionó anteriormente, al momento de finalización del proyecto aún no se ha realizado el despliegue de los sistemas ni las instancias de capacitación, principalmente debido a que ninguna de las empresas ha definido aún el servicio de *hosting* a utilizar. Una vez confirmado el mismo, se procederá con el despliegue.

Posteriormente, se llevará a cabo una etapa de verificación en el entorno productivo, orientada a asegurar la correcta configuración e integración de los sistemas en las condiciones reales de uso de cada compañía. Finalizada esta instancia, se realizarán las capacitaciones correspondientes para los empleados de las distintas áreas que utilizarán la aplicación.

En el caso particular de ViaVeg, la empresa no se encuentra actualmente en condiciones económicas para afrontar los costos del alojamiento. Por este motivo, no es posible establecer con certeza si se podrá avanzar con el despliegue y la implementación del sistema en dicha organización.

9. Conclusiones

El proyecto se presentó con el objetivo de diseñar una solución innovadora para un sector productivo donde la trazabilidad y el seguimiento son fundamentales para lograr una gestión eficiente y segura de los procesos. Si bien en un principio fue concebido como una iniciativa específica para una empresa del sector alimenticio, con el tiempo evolucionó hacia un enfoque más amplio, permitiendo desarrollar un producto genérico pero adaptable a distintos contextos productivos.

Desde el inicio, el equipo enfrentó desafíos vinculados a la coordinación y gestión del trabajo, producto de la diversidad de habilidades y enfoques presentes. Sin embargo, esta misma variedad se transformó en una fortaleza, ya que motivó a los integrantes a aprender nuevas herramientas, mejorar sus capacidades y abordar en conjunto todos los aspectos del proyecto.

Otro aspecto clave fue el impacto negativo de no contar con referentes funcionales comprometidos y disponibles. La baja frecuencia de reuniones y la escasa participación del cliente generaron períodos de estancamiento, incertidumbre y necesidad de re-planificar tareas. En retrospectiva, se reconoció como una omisión no haber contemplado este escenario como un riesgo potencial, tanto en el análisis FODA como en la gestión de riesgos. La escasa iteración con el cliente redujo la capacidad de ajuste progresivo del sistema y limitó la retroalimentación continua. Para mitigar en parte esta limitación, se optó por mantener reuniones frecuentes con el director de proyecto, cuya experiencia y conocimiento del sector resultaron fundamentales para orientar decisiones y mantener la coherencia en el desarrollo. Sin embargo, esta solución no reemplazó el rol clave del cliente como referente funcional, lo que condicionó la dinámica general del proyecto. Este aprendizaje refuerza la necesidad de aplicar con mayor disciplina las metodologías adoptadas, asegurando ciclos de revisión cortos y mecanismos de evaluación que permitan sostener la calidad y la alineación con los objetivos. En futuros proyectos que requieran de metodologías ágiles, además de priorizar la presencia

de representantes activos, el equipo buscará asumir un rol más proactivo, proponiendo desde el inicio dinámicas de trabajo regulares —como reuniones semanales u otros espacios de seguimiento— que favorezcan una comunicación fluida y continua.

Uno de los principales aprendizajes que dejó este proyecto fue la necesidad de sostener una gestión activa y adaptativa durante todo el proceso de desarrollo. Si bien se comenzó con una planificación clara, con el tiempo el foco se desplazó hacia los aspectos técnicos, dejando en segundo plano el seguimiento continuo de los avances. La ausencia de un rol definido de gestión impactó en la asignación eficiente de recursos y en la implementación de mecanismos de control, dificultando la detección temprana de desvíos. Además, se evidenció que, a pesar de contar con una planificación inicial, el margen adicional incluido en la planificación resultó insuficiente ante los desvíos reales.

El desarrollo del análisis FODA y de riesgos permitió al equipo adquirir una visión estructurada sobre los factores internos y externos que afectan al proyecto. Si bien se identificó como debilidad la limitada experiencia del equipo en el desarrollo de software orientado a la industria, durante el desarrollo se comprobó que el conocimiento del dominio no representó una dificultad significativa, gracias al acompañamiento del director de proyecto y a la capacidad del equipo para interpretar los requerimientos del sector. En cambio, una debilidad que debería haber sido considerada desde un principio y que realmente impactó en el proyecto fue la falta de experiencia en gestión de proyectos de esta magnitud. Como aprendizaje, el equipo comprendió que en futuros análisis será clave prestar mayor atención a las debilidades relacionadas con la organización y el control del trabajo, e incorporar desde el inicio herramientas concretas para mejorar la gestión.

En particular, uno de los riesgos más significativos que se materializó fue el de retrasos en el desarrollo, especialmente durante la etapa inicial del proyecto. Si bien se habían previsto márgenes de tiempo y se implementaron herramientas de gestión de tareas, la falta de hitos intermedios y revisiones periódicas dificultó la detección temprana de desvíos. Esta experiencia permitió entender que el riesgo de demora debe ser gestionado proactivamente con una planificación más detallada, incluyendo cronogramas con hitos, revisiones frecuentes y una definición clara de responsabilidades.

Este aprendizaje evidenció que la formalización temprana de planes de contingencia es fundamental para una gestión preventiva, evitando improvisaciones que puedan afectar el desarrollo. En proyectos futuros, se priorizará la integración efectiva entre el análisis de riesgos y el diseño inmediato de planes de contingencia, asegurando una gestión preventiva más robusta.

En relación con la planificación temporal, se observó una estimación optimista, especialmente en el desarrollo del backend y la base de datos, donde se subestimó la complejidad y el tiempo requerido. Esta situación puso en evidencia la importancia de realizar estimaciones con márgenes adecuados de contingencia.

Si bien los plazos generales del proyecto se cumplieron, esto fue posible a costa de dedicar muchas más horas de trabajo de las previstas inicialmente, lo que implicó un aumento en los costos operativos y una presión considerable en las etapas finales. Este aprendizaje refuerza la necesidad de incorporar criterios más realistas y sostenibles en la planificación, que contemplen tanto la duración como el esfuerzo requerido para cada tarea.

Incorporar estos enfoques no solo permite un desarrollo más fluido, sino que fortalece el ejercicio profesional de la ingeniería, preparando al equipo para abordar con mayor solidez futuros proyectos.

Una de las decisiones más acertadas durante el desarrollo fue optar por un diseño modular del sistema. Esta arquitectura permitió incorporar nuevas funcionalidades de forma sencilla, sin comprometer el funcionamiento del núcleo principal. La elección no solo tuvo un impacto positivo en la mantenibilidad y escalabilidad del producto, sino que también dejó una enseñanza valiosa sobre la importancia de concebir la arquitectura con una mirada flexible y orientada a la sostenibilidad a largo plazo.

Además del crecimiento técnico, este proyecto fue clave en la consolidación de competencias propias del ejercicio profesional de la ingeniería informática. Los integrantes del equipo desarrollaron habilidades de resolución de problemas, enfrentaron obstáculos imprevistos, y aprendieron a concebir, diseñar y construir un sistema desde una perspectiva escalable y tecnológicamente viable. También fortalecieron competencias relacionadas con la planificación,

la ejecución y el control de proyectos, reconociendo el valor de las herramientas de gestión para el éxito de los mismos.

En resumen, el trabajo permitió afianzar actitudes fundamentales como el trabajo en equipo, la comunicación clara y profesional, la adaptabilidad y el pensamiento crítico. El equipo valoró especialmente el aprendizaje continuo y autónomo que implicó este proyecto integrador, entendiendo que cada error, dificultad y acierto representó una oportunidad genuina para crecer.

Finalmente, el proyecto no solo permitió consolidar conocimientos técnicos, sino también adquirir y fortalecer competencias profesionales, que serán útiles en el inicio de la carrera profesional en ingeniería informática.

10. Anexo

Requerimientos funcionales

Identificador	**RF01**
Nombre	Iniciar sesión
Descripción	El software deberá permitir ingresar los datos de inicio de sesión (nombre de usuario y contraseña).

Identificador	**RF02**
Nombre	Cerrar sesión
Descripción	El software deberá permitir cerrar la sesión del usuario conectado cuando este quiera finalizarla.

Identificador	**RF03**
Nombre	Alta, baja y modificación de insumos
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de insumos.

Identificador	**RF04**
Nombre	Alta, baja y modificación de productos
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de productos.

Identificador	**RF05**
Nombre	Alta, baja y modificación de depósitos
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de depósitos.

Identificador	**RF06**
Nombre	Alta, baja y modificación de proveedores
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de proveedores.

Identificador	**RF07**
Nombre	Alta, baja y modificación de usuarios
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de usuarios.

Identificador	**RF08**
Nombre	Alta, baja y modificación de fórmulas
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de fórmulas.

Identificador	**RF09**
Nombre	Alta, baja y modificación de órdenes de compra

Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de órdenes de compra.

Identificador	**RF10**
Nombre	Alta, baja y modificación de órdenes de producción
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de orden de producción.

Identificador	**RF11**
Nombre	Alta, baja y modificación de lotes
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de lotes.

Identificador	**RF12**
Nombre	Alta, baja y modificación de remitos
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de registros de remitos.

Identificador	**RF13**
Nombre	Alta, baja y modificación orden de pedido
Descripción	El sistema deberá permitir la creación, modificación, eliminación y visualización de una orden de pedido a

	la cual se asociará un remito.

Identificador	**RF14**
Nombre	Crear plantilla para fórmula
Descripción	El usuario debe poder personalizar plantillas para la elaboración de los productos, para así poder reutilizarlas en un futuro. La plantilla debe indicar los insumos a utilizar con sus cantidades, y el producto final con su cantidad esperada.

Identificador	**RF15**
Nombre	Descontar stock para producción
Descripción	Al generar una orden de producción, se debe especificar el depósito del cual se extraerán los insumos. El depósito va a aparecer bloqueado o disponible dependiendo si existe la cantidad necesaria de cada insumo.

Identificador	**RF16**
Nombre	Calcular costo final del producto
Descripción	El sistema deberá juntar todos los costos de los insumos utilizados para una receta y calcular el costo final del producto, teniendo en cuenta la proporción de cada insumo utilizado.

Identificador	**RF17**
Nombre	Administrar permisos a usuarios
Descripción	Un usuario con rol de administrador debe poder controlar qué funciones tienen habilitadas otros usuarios y cuáles no.

Identificador	**RF18**
Nombre	Movimiento de stock
Descripción	El sistema deberá permitir registrar y visualizar los movimientos de stock entre depósitos, controlando las entradas y salidas de insumos y productos.

Identificador	**RF19**
Nombre	Registro de cambios en stock
Descripción	El sistema deberá registrar qué usuario (nombre, ID, timestamp) realizó un cambio en el stock.

Identificador	**RF20**
Nombre	Registrar usuario
Descripción	El sistema deberá permitir registrar nuevos usuarios, capturando información básica como nombre de usuario, contraseña y correo electrónico.

Requerimientos no funcionales

Identificador	**RNF01**
Nombre	Uso desde dispositivos móviles
Descripción	El sistema deberá ser accesible desde dispositivos móviles, garantizando una experiencia de usuario adecuada.

Identificador	**RNF02**
Nombre	Integridad de la información
Descripción	El sistema deberá garantizar la integridad de la información, asegurando que los datos no sean alterados o corrompidos.

Identificador	**RNF03**
Nombre	Manejo granular de permisos
Descripción	El sistema deberá permitir la asignación granular de permisos a cada usuario, definiendo accesos específicos a funcionalidades del sistema.

Identificador	**RNF04**
Nombre	Usabilidad
Descripción	La interfaz del sistema deberá ser intuitiva y de fácil uso, permitiendo que usuarios sin capacitación previa puedan operar las funciones básicas del sistema.

Identificador	**RNF05**
Nombre	Alta disponibilidad
Descripción	El sistema deberá tener un *downtime* menor a 10 minutos.

Identificador	**RNF06**
Nombre	Escalabilidad
Descripción	El sistema deberá estar diseñado para escalar horizontalmente y manejar un incremento en la cantidad de usuarios o volumen de datos sin degradar su desempeño.

Identificador	**RNF07**
Nombre	Extensibilidad
Descripción	El sistema deberá permitir la incorporación de nuevas funcionalidades o módulos en el futuro, con un mínimo impacto en los componentes existentes.

Identificador	**RNF08**
Nombre	Mantenibilidad
Descripción	El sistema deberá facilitar la corrección de errores y la modificación del código sin afectar al resto de las funcionalidades.

Requerimientos específicos para ViaVeg

Identificador	**RFV01**
Nombre	Control de producción en batch
Descripción	El sistema deberá controlar la producción en batch, registrando los insumos, su cantidad, el lote al que pertenecen, el estado del batch, la hora de inicio y finalización.

Identificador	**RFV02**
Nombre	Calcular merma de la producción
Descripción	El sistema debe llevar un registro de la pérdida de producción para poder tomar decisiones en un futuro.

Requerimientos específicos para Coppens

Identificador	**RFC01**
Nombre	Control de calidad en recepción del producto
Descripción	El sistema deberá permitir introducir resultados de las pruebas de control de calidad

Planes de contingencia

R02: Resistencia de empresas tradicionales

Para mitigar la resistencia que presentan las empresas con procesos tradicionales ante la adopción del nuevo producto, se planteó inicialmente desarrollar estrategias de marketing y educación que destaquen claramente los beneficios del sistema y aborden las principales preocupaciones de estos clientes.

A posteriori, se considera que este plan debería haberse complementado con acciones más concretas, tales como la creación de materiales educativos claros y accesibles, además de ofrecer demostraciones en vivo del producto para mostrar su funcionamiento y ventajas de manera práctica. También habría sido beneficioso implementar pruebas piloto sin compromiso que permitieran a los clientes experimentar el sistema en su propio entorno, reduciendo así las barreras iniciales, respondiendo dudas y facilitando la integración del producto.

Finalmente, en caso de resistencia o rechazo, se deberían haber aplicado acciones reactivas como la comunicación directa para entender las objeciones, la implementación de beneficios temporales que incentiven la prueba y el uso, y la adaptación del producto basada en el *feedback* de los usuarios.

R03: Falta de monitoreo del mercado y pérdida de diferenciación

Para mantener la competitividad del producto en un entorno dinámico, se planteó inicialmente la necesidad de realizar un monitoreo constante del mercado y buscar formas de diferenciación y mejora continua.

En retrospectiva, se debería haber implementado un sistema de monitoreo activo con dashboards y herramientas que midan métricas de uso, satisfacción y desempeño del sistema. Por ejemplo, podrían haberse considerado métricas como usuarios activos, tiempo promedio de uso por sesión, y tiempo de carga promedio por funcionalidad o pantalla, entre otras. Estas métricas habrían permitido detectar patrones de adopción, identificar posibles cuellos de botella técnicos y evaluar el grado de compromiso de los usuarios con la solución. Además, se deberían haber realizado investigaciones mediante encuestas y entrevistas, dirigidas a usuarios y áreas involucradas para lograr identificar funcionalidades clave y detectar áreas de mejora reales.

En caso de disminución en el interés o adopción del sistema, se debería haber investigado internamente las causas a través de análisis de *feedback* y métricas de desempeño, para luego ejecutar ajustes técnicos o funcionales que mejoren la experiencia del usuario.

R05: Falta de flexibilidad del producto

Para garantizar que el sistema pueda adaptarse a las necesidades de distintos clientes y evitar que su rigidez limite la expansión comercial o la adopción del producto en nuevos rubros, se planteó inicialmente como parte del plan de contingencia investigar y priorizar los rubros más importantes para los potenciales clientes.

A posteriori, se reconoce que este enfoque fue limitado y debería haber incluido un *roadmap* con instancias de relevamiento a trabajadores de las distintas áreas de las empresas para permitir detectar necesidades reales, priorizar funcionalidades útiles y planificar actualizaciones enfocadas en mejorar el trabajo de los usuarios finales.

Además, podría haberse complementado con pruebas piloto en sectores específicos para validar las funcionalidades antes de incorporarlas al producto general.

R06: Uso inadecuado del software

Para reducir los errores derivados del uso incorrecto del sistema por parte de los usuarios, se planteó la implementación de programas de capacitación continua y documentación clara.

Luego de finalizada la etapa de despliegue e implementación del sistema, se debería haber previsto una capacitación orientada a garantizar que los usuarios comprendan el uso general del sistema y puedan operar con autonomía.

La capacitación se debería organizar en sesiones breves y prácticas, adaptadas a los distintos perfiles de usuarios. Asimismo, se debería planificar la elaboración de un conjunto de materiales educativos, que incluyan instructivos descargables y videotutoriales con el fin de ofrecer un soporte asincrónico.

Además de las acciones orientadas a los usuarios, se deberían implementar mecanismos técnicos para reducir la posibilidad de errores en la carga de información. Entre ellos, se destaca el uso de formularios estructurados para el ingreso de datos y validaciones

automáticas que verifican el tipo y formato de los datos ingresados. Esto permite prevenir incongruencias, minimizar errores de tipeo y asegurar la consistencia de la información dentro del sistema.

En caso de presentarse errores recurrentes o uso indebido del sistema, se deberían haber implementado acciones reactivas como el análisis de patrones de error para detectar causas comunes, la asignación de tutorías personalizadas para reforzar el aprendizaje y la revisión de la interfaz del sistema en busca de oportunidades de mejora en términos de usabilidad.

R07: Retrasos en el desarrollo del proyecto

Una forma de mitigar los retrasos del proyecto fue contemplar márgenes de tiempo para imprevistos y posibles desvíos en las tareas más grandes. Además, se utilizó un sistema de gestión de tareas para organizar el trabajo y hacer seguimiento del avance de cada actividad.

Sin embargo, se deberían haber establecido hitos de control con revisiones periódicas del avance, para facilitar la detección temprana de problemas y la implementación de medidas correctivas sin comprometer los plazos generales ni aumentar los costos del proyecto.

Otro aspecto fundamental que se debería haber realizado es una asignación clara y eficiente de roles y tareas dentro del equipo desde el principio del proyecto para evitar solapamientos, demoras en la toma de decisiones y sobrecarga de trabajo en ciertos perfiles. Además, se deberían haber documentado las modificaciones en los requerimientos o los plazos y evaluado su impacto.

R08: Falta de planificación en el despliegue e improvisación ante imprevistos

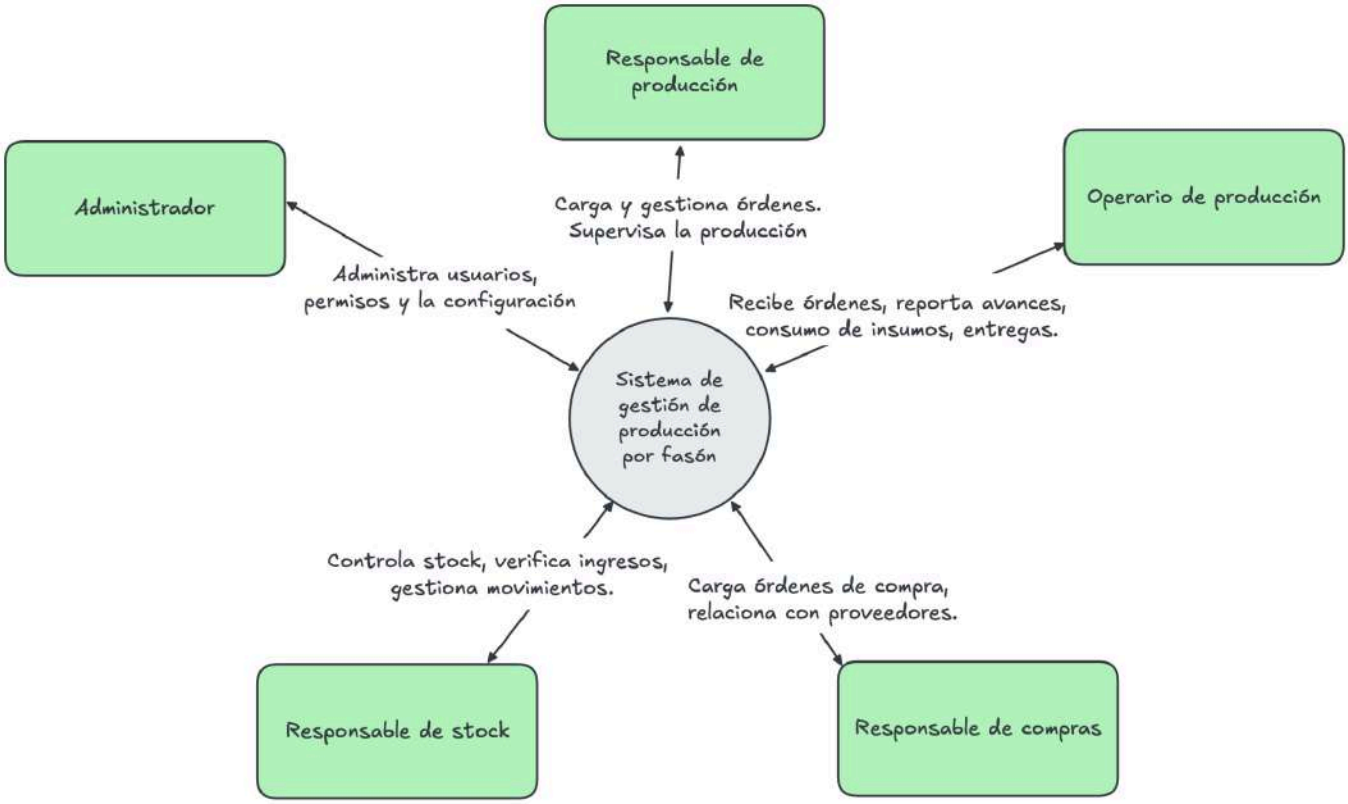
Durante la planificación inicial, se identificó la importancia de prever tiempos adecuados para el despliegue del sistema y contar con soporte técnico oportuno. Se planteó como medida clave planificar el despliegue con un margen de tiempo suficiente que permitiera absorber imprevistos sin comprometer la operación del cliente. Asimismo, se definió que habría un equipo de soporte disponible durante la implementación para atender incidentes de forma inmediata.

A posteriori, se consideró que se debieron establecer mejores estrategias para el despliegue, incluyendo la planificación del mismo por etapas o módulos, permitiendo así una

implementación progresiva y controlada que minimizara riesgos y facilitara la detección temprana de problemas

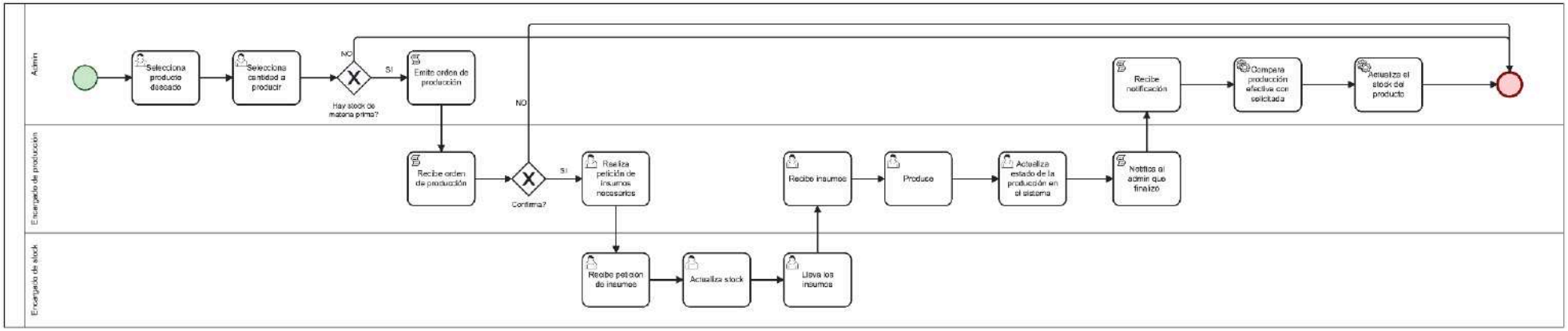
Se deberían haber realizado simulacros de despliegue o pruebas piloto en ambientes controlados antes de la implementación definitiva, para detectar fallos potenciales sin afectar al entorno real del cliente. Todos los incidentes deberían documentarse para retroalimentar la planificación futura y minimizar la probabilidad de ocurrencia de problemas similares.

Diagrama de contexto



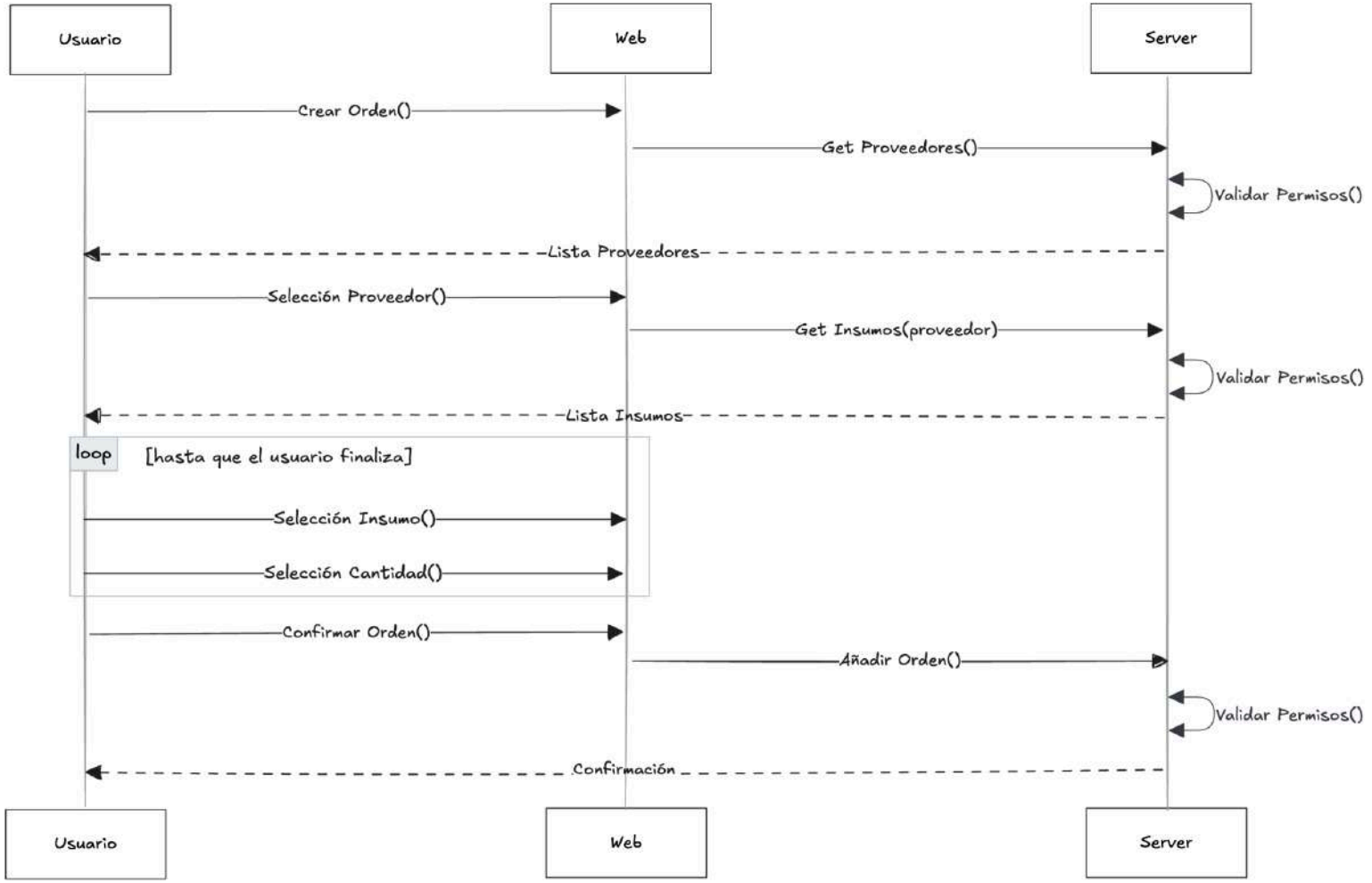
BPMN

Proceso de órdenes de producción



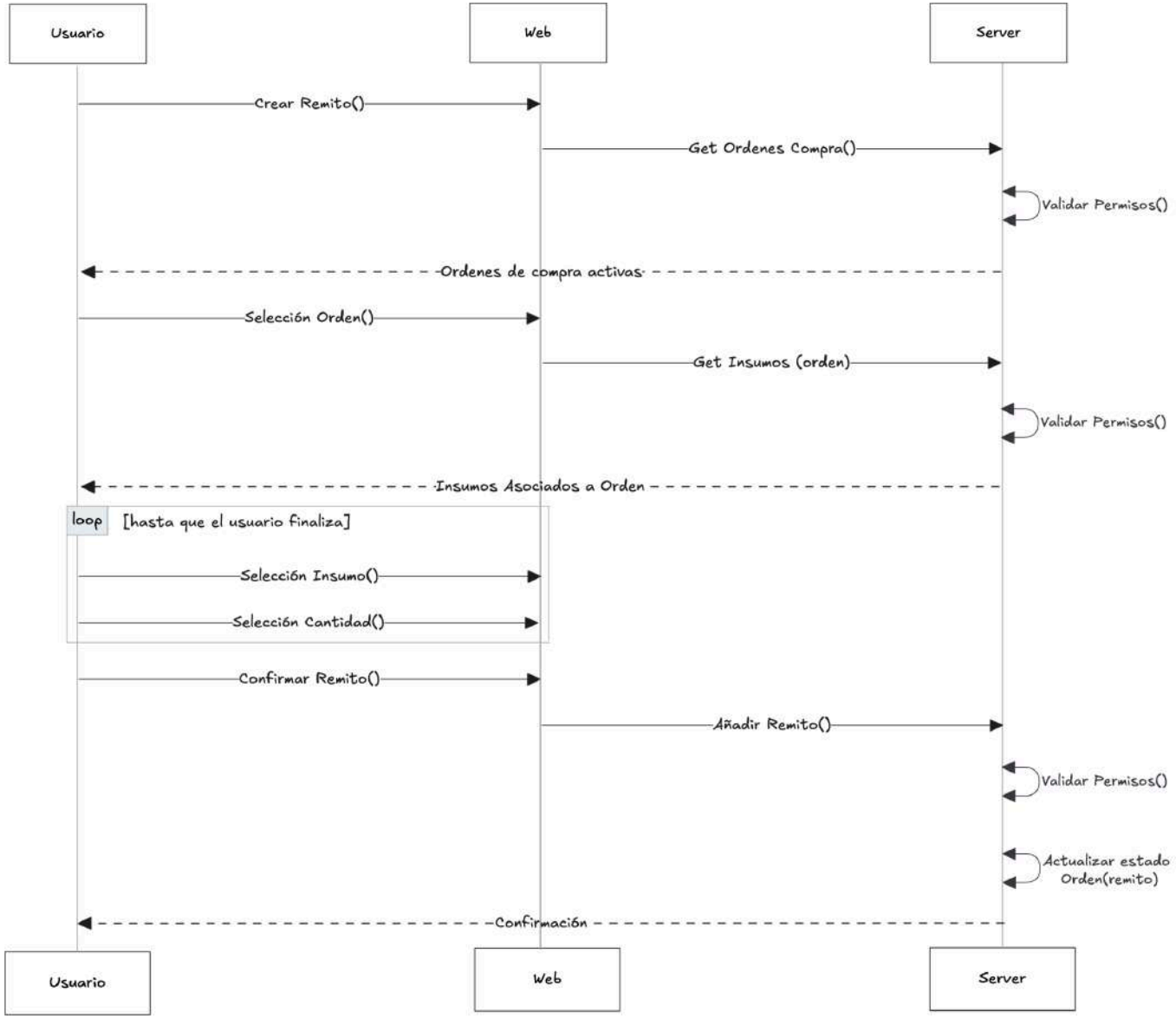
Diagramas de secuencia[1]

Crear orden de compra

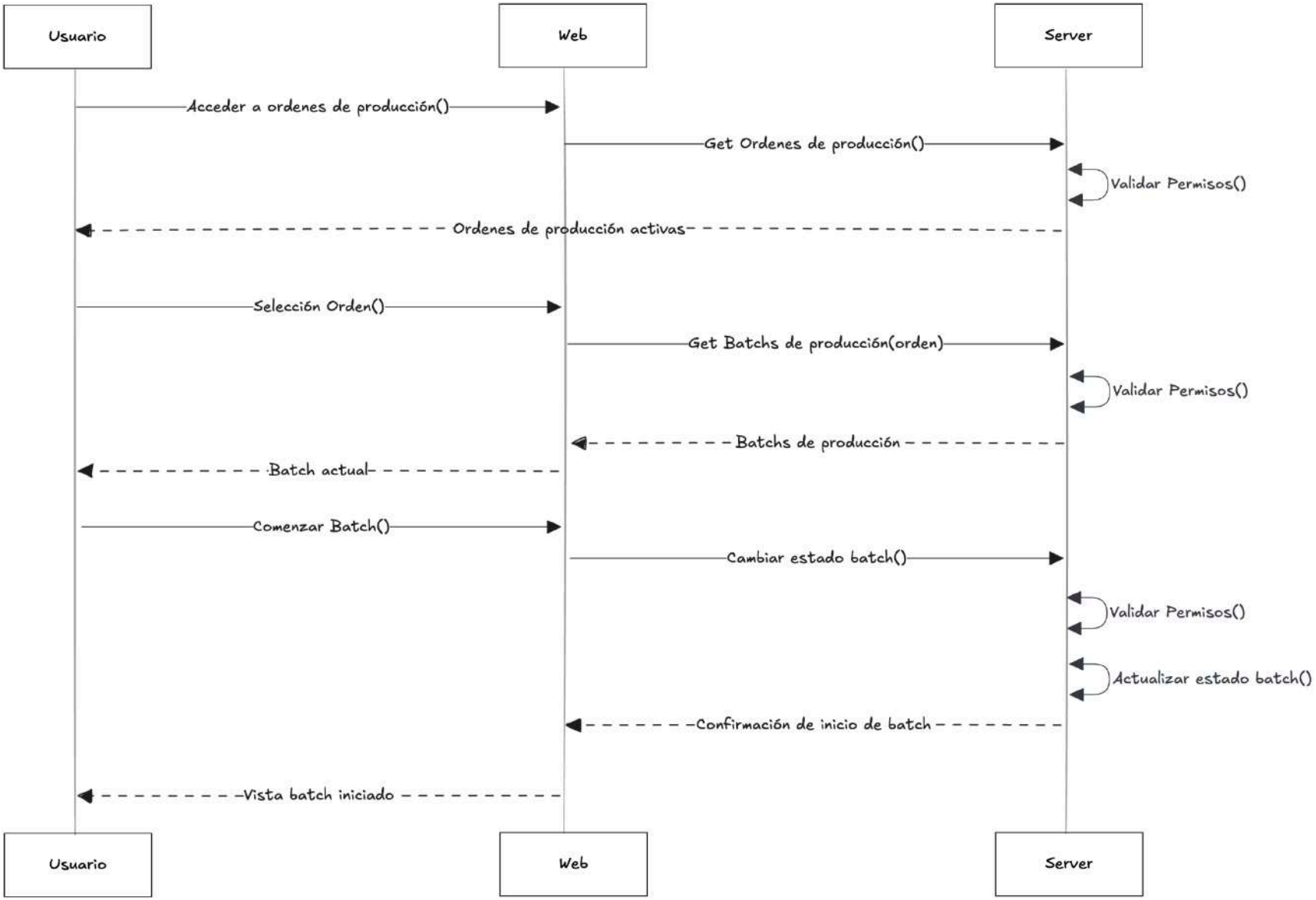


[1]Se optó por diagramar aquellos casos que se consideraron más complejos y, por lo tanto, más relevantes e interesantes para el análisis.

Crear remito



Iniciar batch de producción (ViaVeg)



Completar batch de producción (ViaVeg)

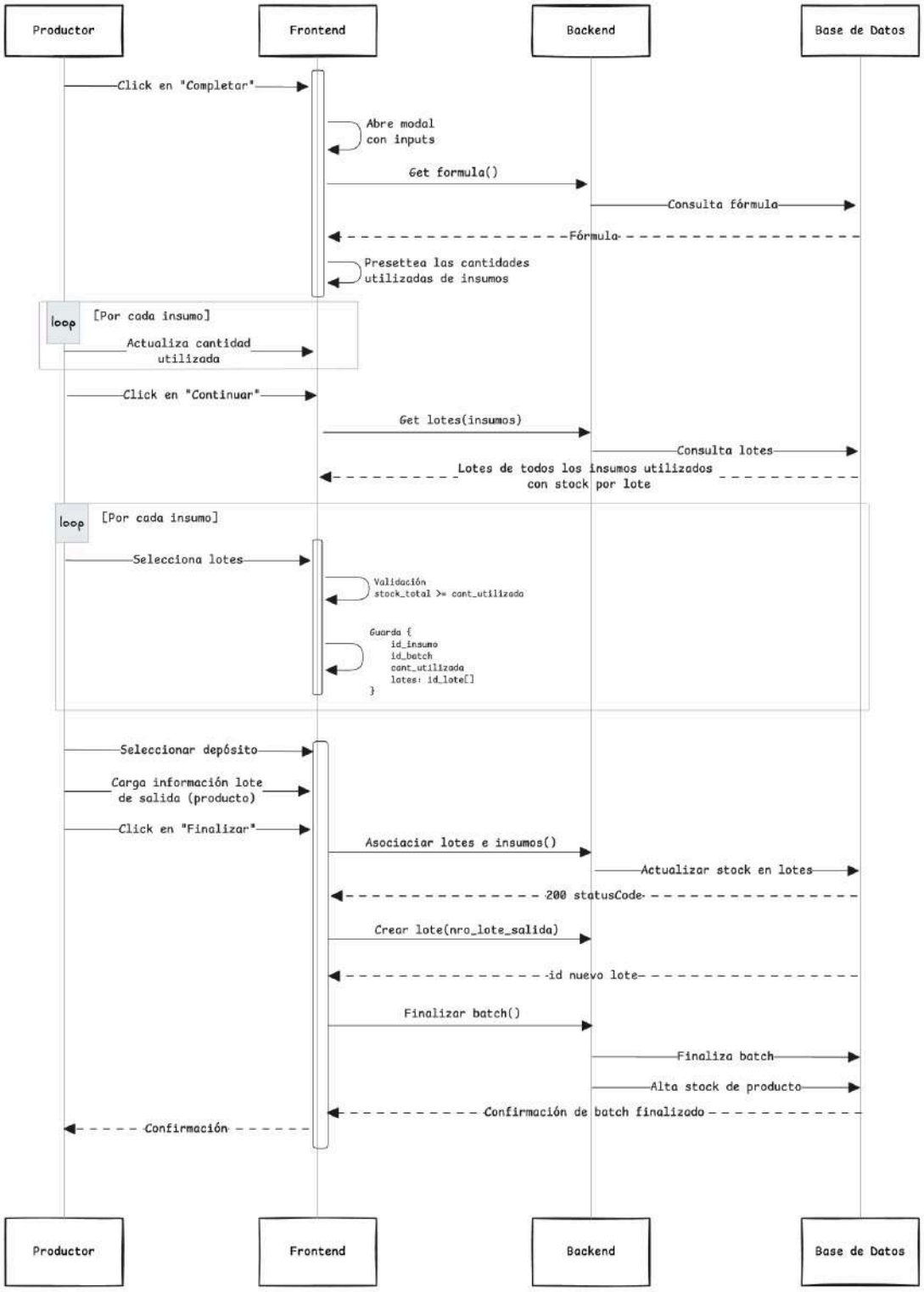
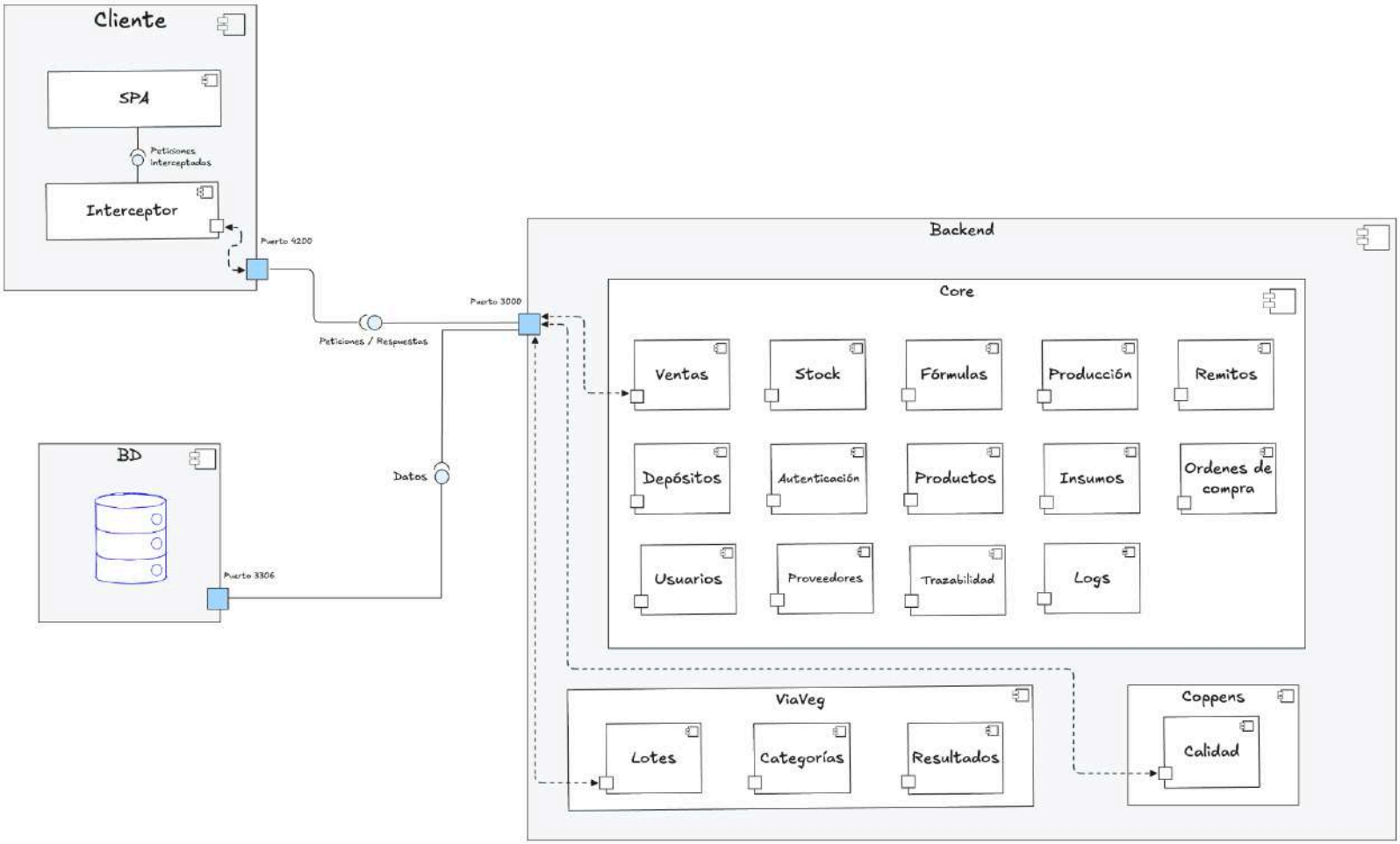
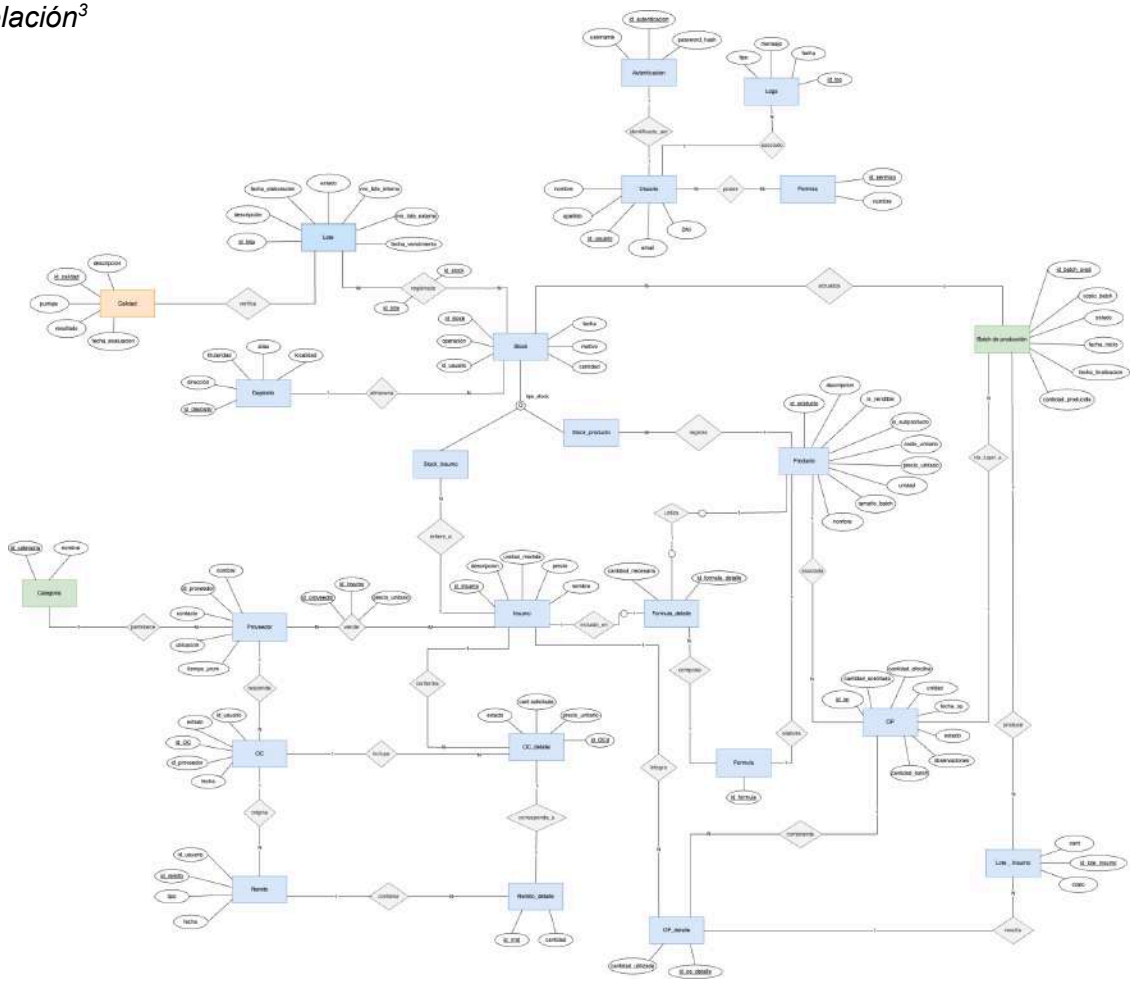


Diagrama de Componentes[2]



[2] Con el fin de mejorar la claridad del diagrama, se omitieron las líneas que representan la conexión de los módulos con el puerto del *backend*, ya que todos comparten el mismo. Tampoco se incluyeron las interfaces que vinculan los módulos entre sí, por la misma razón.

Diagrama entidad-relación[3]



[3] Se decidió incluir en el diagrama las entidades propias de las adaptaciones del sistema para ViaVeg y Coppens, utilizando los colores verde y naranja, respectivamente.

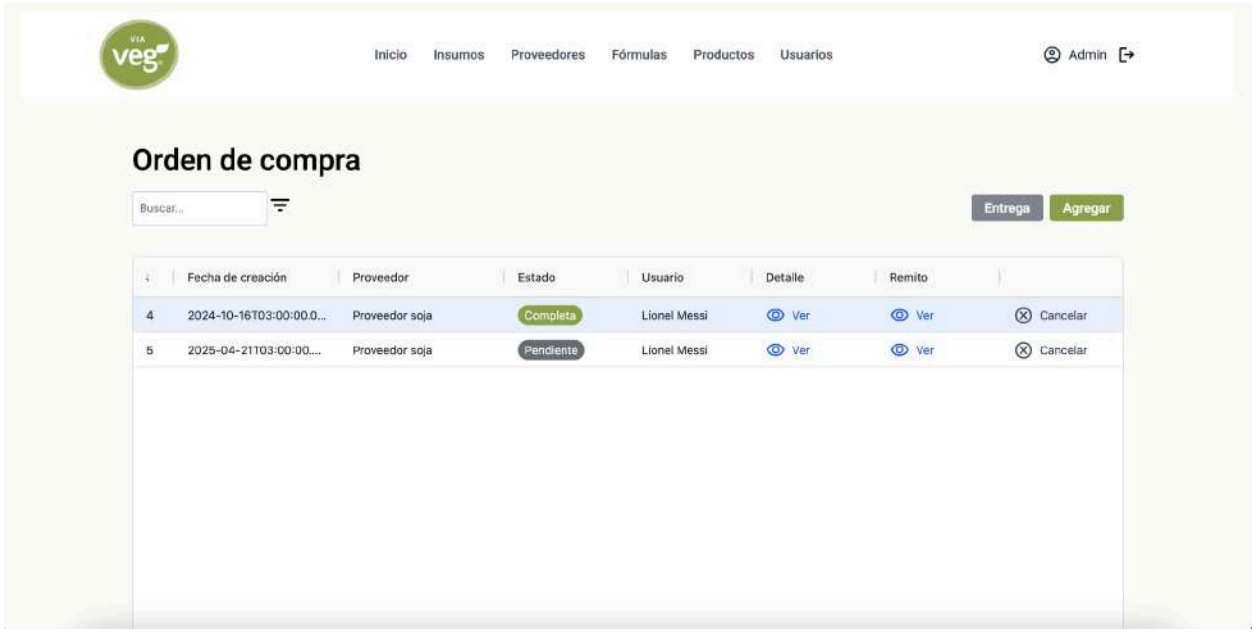
Tabla de permisos

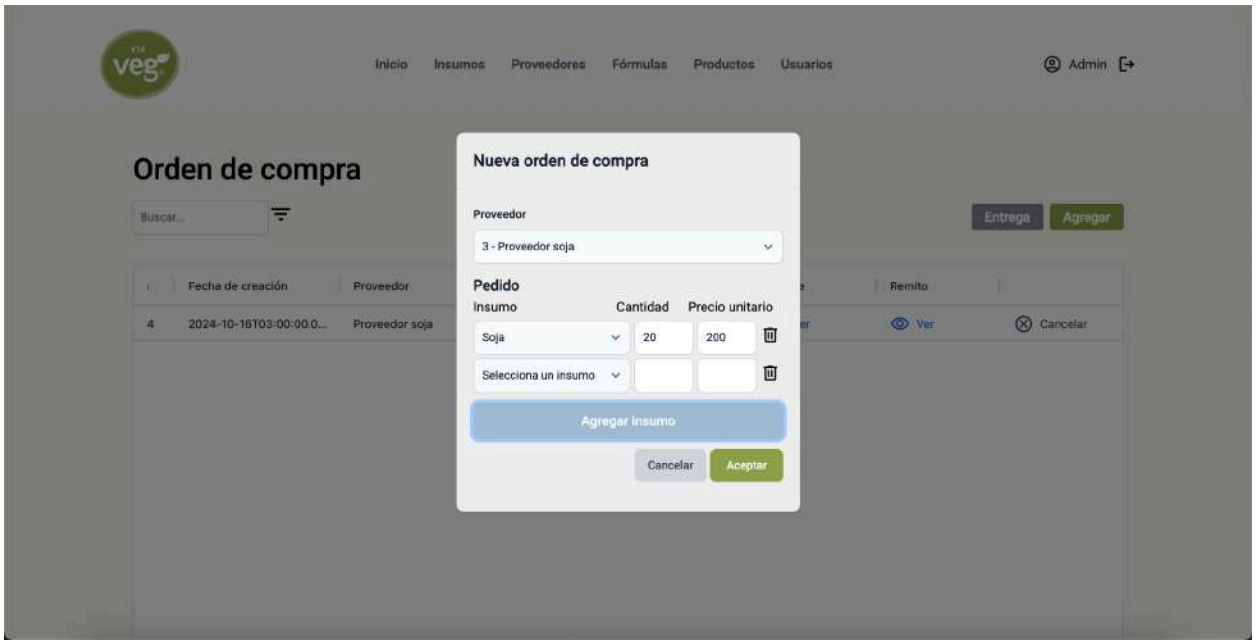
Acción	Habilidad funcional	Admin	Usuario maestro	Responsable de producción	Responsable de compras	Responsable de stock	Operario de planta	Usuario básico
Órdenes de Compra								
Escritura OC	write_oc	❌	✅	❌	✅	❌	❌	❌
Lectura OC	read_oc	❌	✅	❌	✅	✅	❌	❌
Órdenes de Producción								
Escritura OP	write_op	❌	✅	✅	✅	❌	✅	❌
Lectura OP	read_op	❌	✅	✅	✅	✅	✅	❌
Remitos								
Escritura remito	write_remito	❌	✅	❌	✅	✅	❌	❌
Lectura remito	read_remito	❌	✅	❌	✅	✅	❌	❌
Productos y Proveedores								
Escritura productos	write_producto	❌	✅	❌	✅	✅	❌	❌
Lectura productos	read_producto	❌	✅	✅	✅	✅	✅	✅
Escritura proveedores	write_proveedor	❌	✅	❌	✅	❌	❌	❌
Lectura proveedores	read_proveedor	❌	✅	✅	✅	✅	❌	❌
Insumos y Fórmulas								
Escritura insumos	write_insumo	❌	✅	❌	✅	✅	❌	❌
Lectura insumos	read_insumo	❌	✅	✅	✅	✅	✅	✅
Escritura fórmula	write_formula	❌	✅	❌	✅	✅	❌	❌
Lectura fórmula	read_formula	❌	✅	✅	✅	✅	✅	❌
Stock y Depósitos								
Escritura stock	write_stock	❌	✅	✅	❌	✅	❌	❌

Lectura stock	read_stock	❌	✅	✅	✅	✅	✅	✅
Escritura depósito	write_deposito	❌	✅	❌	❌	✅	❌	❌
Lectura depósito	read_deposito	❌	✅	✅	✅	✅	✅	❌
Usuarios								
Escritura usuario	write_usuario	✅	❌	❌	❌	❌	❌	❌
Lectura usuario	read_usuario	✅	❌	❌	❌	❌	❌	❌

Vistas del sistema[4]

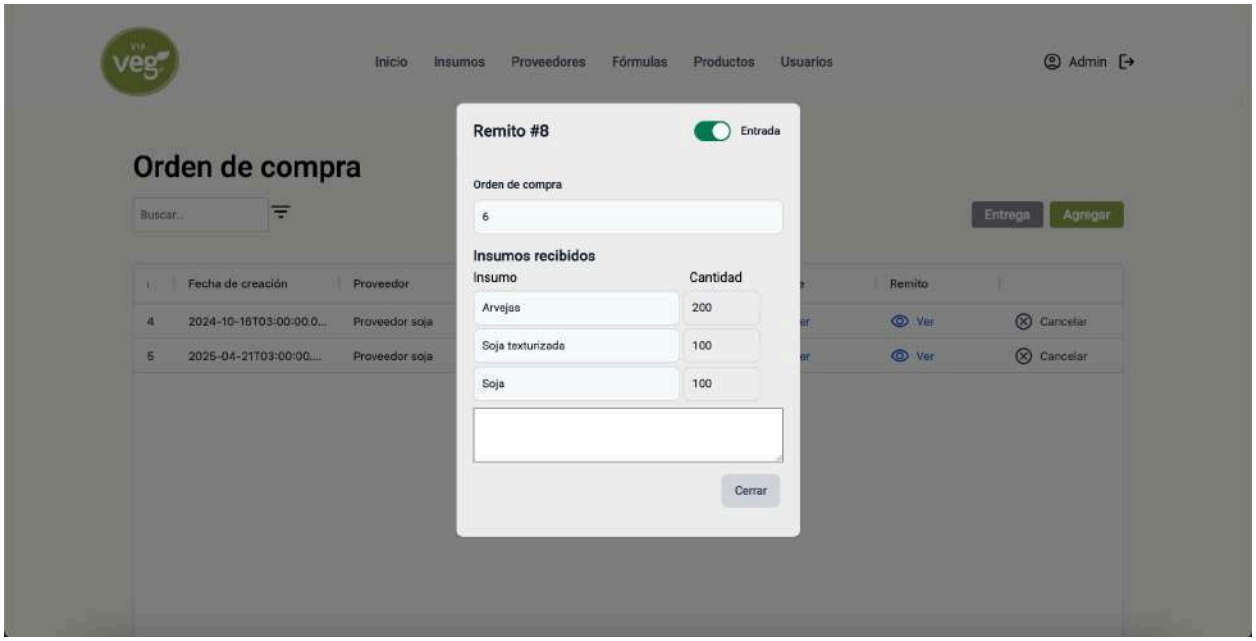
Orden de compra



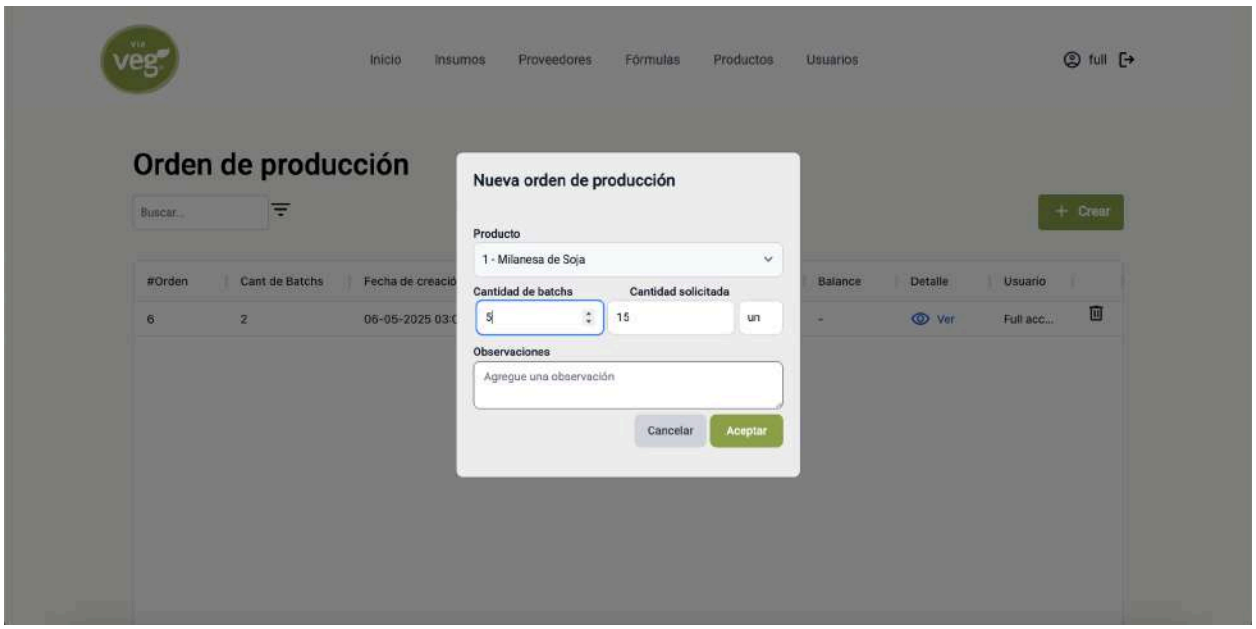


[4] Se decidió limitar las capturas de pantalla a sólo aquellas más representativas para mejorar la legibilidad del documento.

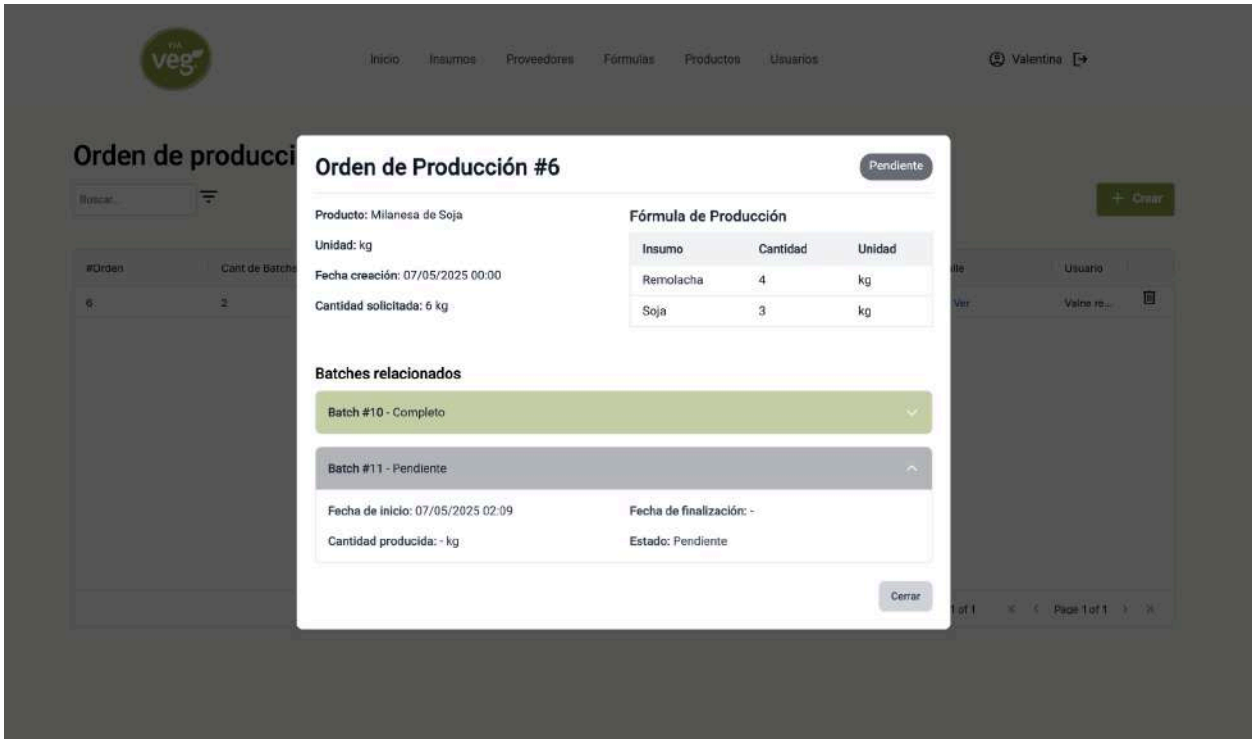
Remito



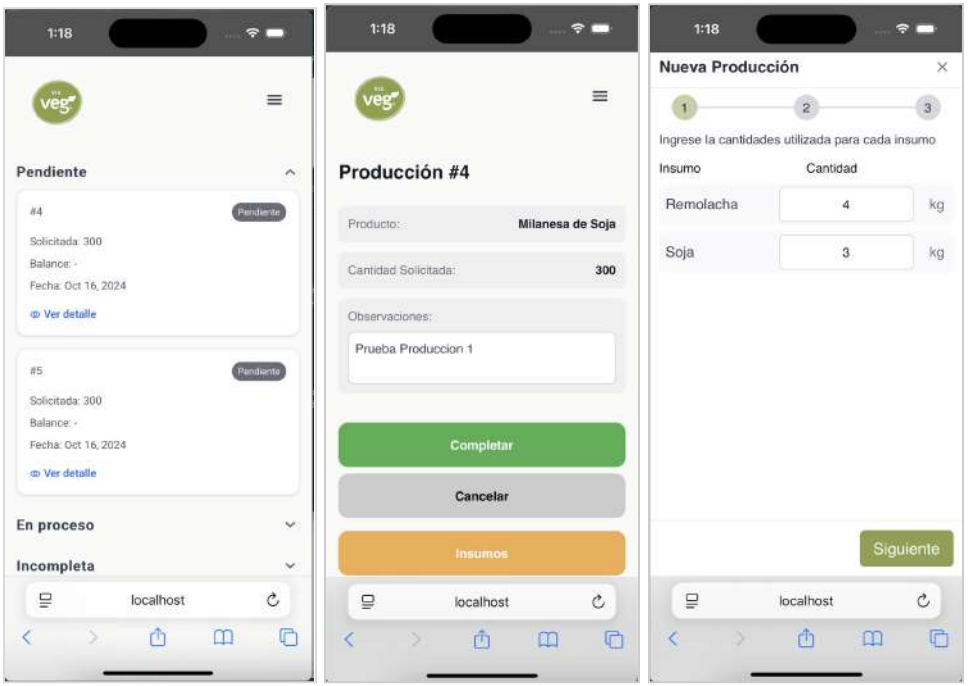
Orden de producción - Administrador



Detalles de orden de producción



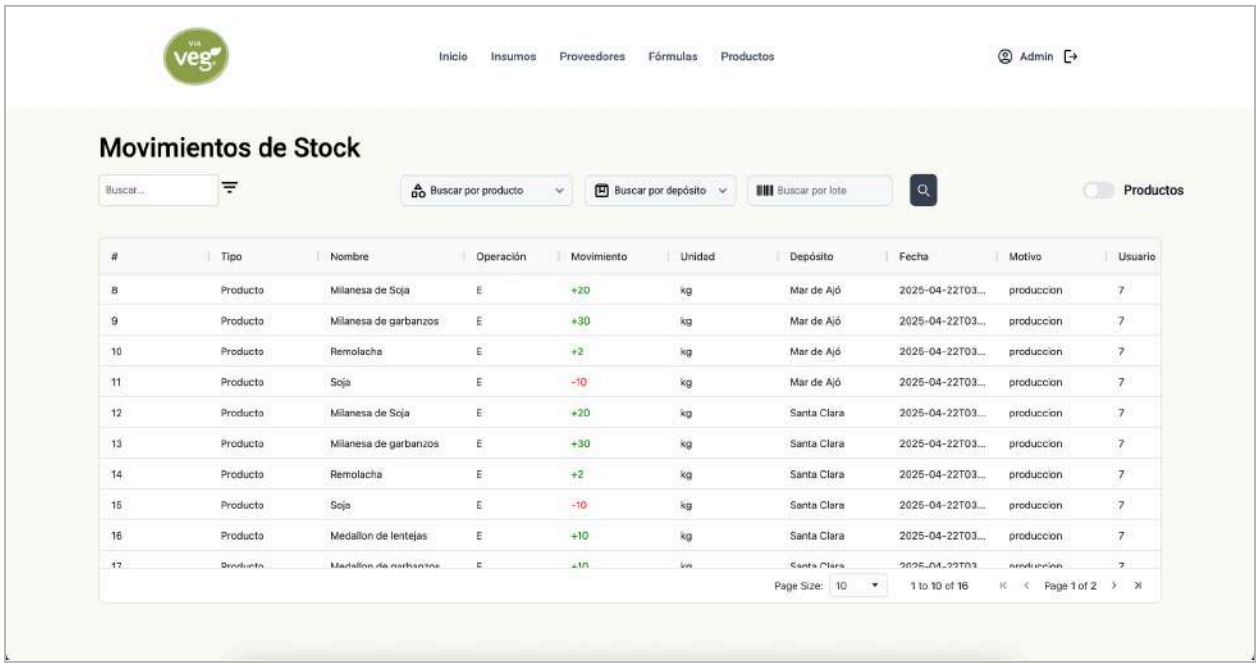
Orden de producción - Productor



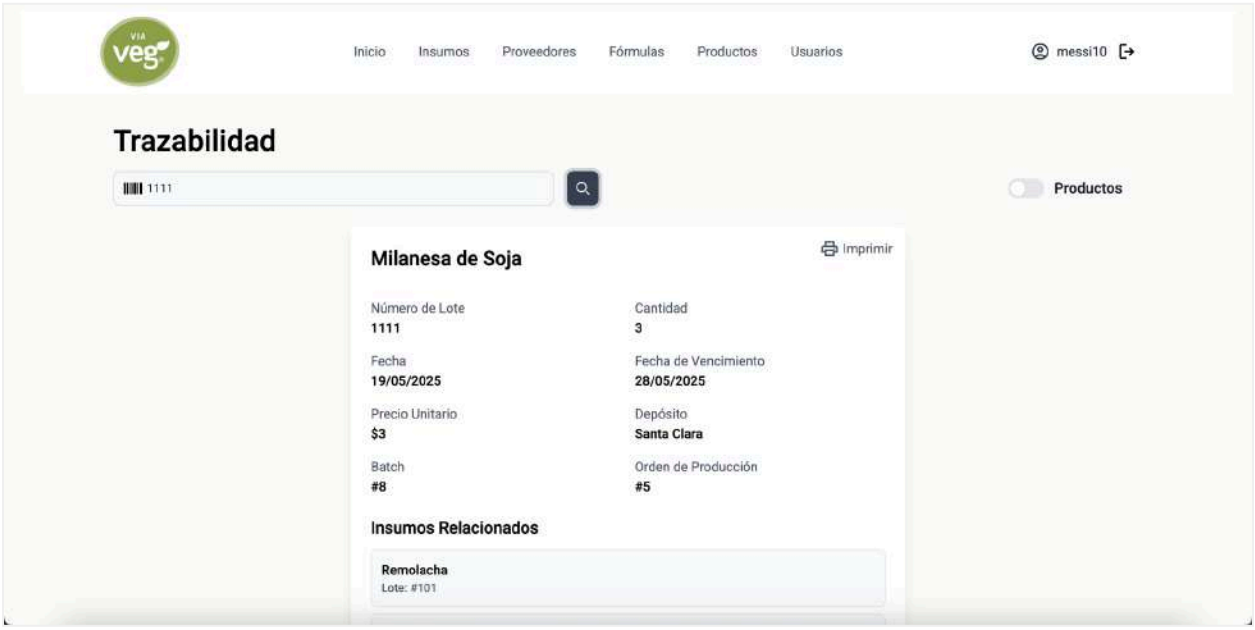

1:18
Nueva Producción
1
2
3
Seleccione los lotes utilizados para cada insumo
Remolacha
20/4 kg
Lote 101 - 20 kg
Lote 103 - 10 kg
Soja
20/3 kg
Lote 102 - 20 kg
Anterior
Siguiente
localhost

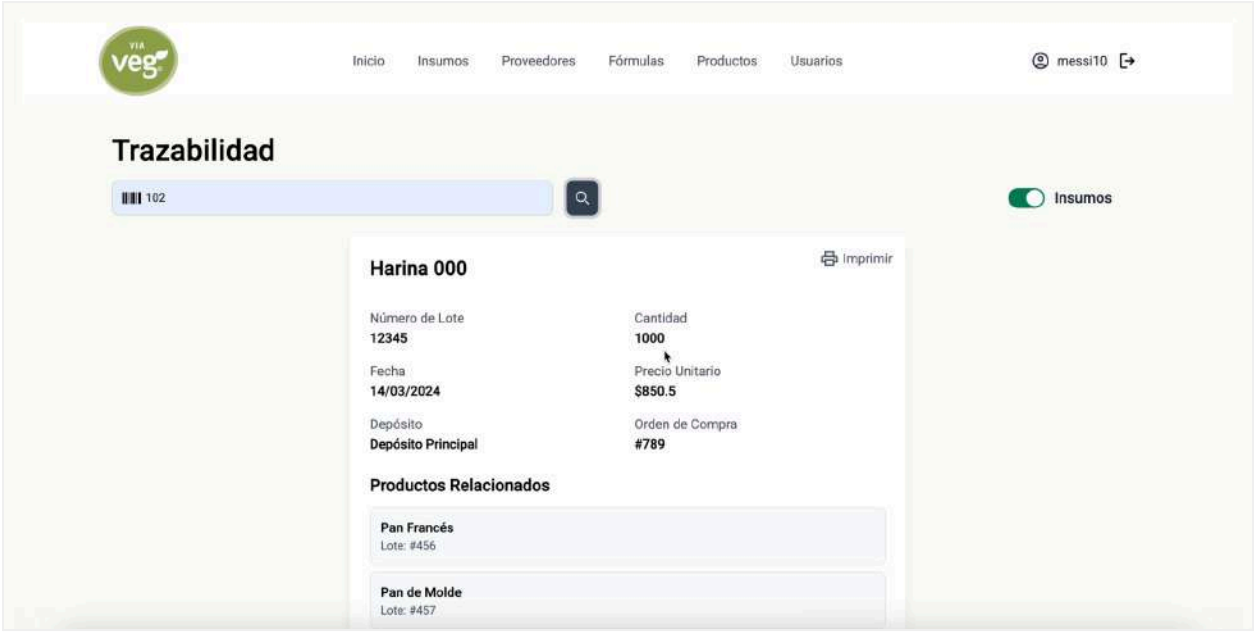

1:18
Nueva Producción
1
2
3
Información de salida
Santa Clara
Descripción
Esto es una prueba
Número de Lote
12345455
Fecha de Vencimiento
28 ago 2025
Anterior
Finalizar
localhost

Stock


Inicio
Insumos
Proveedores
Fórmulas
Productos
Admin
Movimientos de Stock
Buscar...
Buscar por producto
Buscar por depósito
Buscar por lote
Productos
Tipo Nombre Operación Movimiento Unidad Depósito Fecha Motivo Usuario
8 Producto Milanesa de Soja E +20 kg Mar de Ajó 2025-04-22T03... produccion 7
9 Producto Milanesa de garbanzos E +30 kg Mar de Ajó 2025-04-22T03... produccion 7
10 Producto Remolacha E +2 kg Mar de Ajó 2025-04-22T03... produccion 7
11 Producto Soja E -10 kg Mar de Ajó 2025-04-22T03... produccion 7
12 Producto Milanesa de Soja E +20 kg Santa Clara 2025-04-22T03... produccion 7
13 Producto Milanesa de garbanzos E +30 kg Santa Clara 2025-04-22T03... produccion 7
14 Producto Remolacha E +2 kg Santa Clara 2025-04-22T03... produccion 7
15 Producto Soja E -10 kg Santa Clara 2025-04-22T03... produccion 7
16 Producto Medallon de lentejas E +10 kg Santa Clara 2025-04-22T03... produccion 7
Page Size: 10
1 to 10 of 16
Page 1 of 2

Trazabilidad





11. Bibliografía

Netlify. (s.f.). *Role-based access control*. Netlify. Recuperado de https://docs.netlify.com/security/secure-access-to-sites/role-based-access-control/

Jovanovic, M. (s.f.). *What is a modular monolith?* Recuperado de https://www.milanjovanovic.tech/blog/what-is-a-modular-monolith

Jest. (s.f.). *Delightful JavaScript testing*. Recuperado de https://jestjs.io

Swagger. (s.f.). *API development for everyone*. Recuperado de https://swagger.io

Docker. (s.f.). *Sitio web oficial*. Recuperado de https://www.docker.com/

Angular. (s.f.). *Angular official site*. Recuperado de https://angular.dev/

Node.js. (s.f.). *Node.js API documentation*. Recuperado de https://nodejs.org/docs/latest/api/

Objection.js. (s.f.). *Objection.js ORM*. Recuperado de https://vincit.github.io/objection.js/