

Proyecto de título

VulcaStock Integral





Índice

**Tabla de contenido**

[***1.***](#_heading=h.4f07z5gy94lt) ***Abstract 4***

[***2.***](#_heading=h.ljf9xegylfn1) ***Descripción del proyecto 5***

[***3.***](#_heading=h.w3aemvu98nk8) ***Justificando su relevancia 5***

[***4.***](#_heading=h.hq6dlc9mlcdh) ***Competencia del Perfil de Egreso 6***

[***5.***](#_heading=h.r9n3bjbfdtkb) ***Intereses Profesionales 6***

[***6.***](#_heading=h.iwraakf08rpp) ***Argumentos sobre la factibilidad del proyecto dentro de la asignatura (Capstone) 6***

[***7.***](#_heading=h.doieozaambit) ***Objetivo General 8***

[**Objetivos Específicos: 8**](#_heading=h.s3vps8wrclzc)

[***8.***](#_heading=h.gaoi0y9ruvm7) ***Propuesta de Metodología: Enfoque Ágil 8***

[**Metodología Ágil para el Proyecto APT (4 Meses) 9**](#_heading=h.nk3o11h1amne)

[Sprint 0: Preparación y Configuración Inicial (Semana 1) 9](#_heading=h.4rxskhhsb2qu)

[Sprint 1: Planificación y Diseño (Semanas 2-3) 9](#_heading=h.ohr9ayrcxae1)

[Sprint 2: Desarrollo de Módulos Principales (Semanas 4-5) 9](#_heading=h.ah2ul35yh9f8)

[Sprint 3: Desarrollo de Funcionalidades y APIs (Semanas 6-8) 10](#_heading=h.dktbh6y6jn0v)

[***8.***](#_heading=h.kh27m0emavy7) ***Plan de trabajo 10***

[***9.***](#_heading=h.kezhh7i8q36) ***Determinacion de evidencias y su justificacion 14***

[***10.***](#_heading=h.z6lqzgp1n7um) ***Carta Gantt 15***

[***11.***](#_heading=h.yijvzqrcugvv) ***Requerimientos funcionales 17***

[***12.***](#_heading=h.cxode7jl2dc2) ***Requerimientos no funcionales 17***

[***13.***](#_heading=h.jnlludjdknif) ***Tecnologías a utilizar 18***

[React 18](#_heading=h.exmq9tksd9ca)

[Node.js 19](#_heading=h.bdc74v1fmaog)

[Django (Python) 19](#_heading=h.7orgtxv0qmn6)

[PostgreSQL 19](#_heading=h.vou42sqgbio9)

[Herramientas de Integración y Colaboración 19](#_heading=h.t8kl07s7v949)

[***14.***](#_heading=h.uvca1n48id7e) ***Definición de épicas 20***

[***15.***](#_heading=h.l0brcmn6hwia) ***Historias de usuarios 20***

[***16.***](#_heading=h.yjf7otn2w9s2) ***Casos de uso 23***

[***17.***](#_heading=h.2svbpgs5fc8j) ***Sprint Backlog 23***

[***18.***](#_heading=h.rkpnfcploqqz) ***Diseño del diagrama de flujo 26***

[***19.***](#_heading=h.q0cq0bzbo3w9) ***Diseño diagrama relacional 27***

# Abstract

El proyecto APT tiene como objetivo desarrollar una plataforma web innovadora que gestione de manera eficiente los procesos operativos de una vulcanización, abordando aspectos clave como emisión de boletas, control de stock y administración de ventas. A partir de tecnologías avanzadas de análisis de datos, la plataforma implementa un *dashboard inteligente* que ofrece informes detallados en tiempo real, facilitando la revisión de métricas diarias, semanales, mensuales y anuales. Con esta información, los tomadores de decisiones pueden generar estrategias que mejoren la rentabilidad y optimicen la productividad de la empresa.

Al igual que otros sistemas de vanguardia, el proyecto APT se fundamenta en un diseño orientado a la experiencia del usuario. Este planteamiento asegura la evolución continua del proyecto, permitiendo la incorporación de mejoras que garanticen la fiabilidad y escalabilidad de la solución. De esta forma, el desarrollo se adapta a la creciente necesidad de una gestión global y automatizada, ofreciendo un valor diferencial al sector de las empresas al integrar múltiples funciones en una sola plataforma.

Por último, en términos de sostenibilidad y rentabilidad, el proyecto considera la posibilidad de expandir sus funcionalidades a mercados afines, ofreciendo complementos especializados para el control de inventarios y servicios en general para cualquier tipo de empresa. Esta estrategia de extensión de funciones y alianzas, sumada a la automatización y el análisis inteligente de datos, posiciona al proyecto APT como una propuesta sólida y competitiva dentro del mercado digital, enfocada en maximizar ganancias y mejorar la toma de decisiones de sus usuarios.

# Descripción del proyecto

El proyecto APT es una plataforma web que busca modernizar y optimizar la operación de una empresa de vulcanización, integrando en un solo sistema la gestión de boletas, el control de stock y la administración de ventas. Para lograrlo, se han diseñado módulos altamente personalizables que permiten desde la emisión de documentos tributarios hasta el registro detallado de productos y servicios, atendiendo la necesidad de un control preciso tanto de inventarios como de flujos de ingreso y egreso de dinero.

El elemento diferenciador de este proyecto radica en su *dashboard inteligente*, una funcionalidad que recopila y analiza datos operativos en tiempo real, mostrando los resultados diarios, semanales, mensuales y anuales de la empresa. De esta forma, los usuarios pueden detectar tendencias, observar de cerca el rendimiento del negocio y tomar decisiones más informadas. Además, gracias a la información acumulada, el sistema está diseñado para ofrecer sugerencias y alertas, contribuyendo de manera efectiva al mejoramiento de la productividad y la rentabilidad de la empresa.

# Justificando su relevancia

**Aplicación real y Pertinencia Practica:**

El proyecto aborda un problema genuino de una empresa (control de ingresos, inventarios y emisión de boletas), lo que otorga un carácter práctico y contextualizado a la propuesta.

**Integración de conocimientos:**

Combino habilidades de programación web (react y backend con django) con la gestión de base de datos (postgreSQL), abordando la arquitectura de software de manera integral y me permite aplicar metodologías de desarrollo agiles, así reforzando competencias de gestión de proyectos.

**Desarrollo de competencias profundas:**

Este proyecto fortalece el pensamiento crítico y la continua resolución de problemas al diseñar un sistema capaz de adaptarse a escenarios reales de facturación y seguimiento de stock.

**Escalabilidad y evolución continua:**

El proyecto está hecho de una manera modular, facilitando la incorporación de nuevas características (por ejemplo, expansión a otros tipos de negocios o mayor sofisticación en el análisis de datos).

**Contribución a la formación profesional personal:**

Al acabar este proyecto adquiero experiencia completa de desarrollo, así ofreciendo una carta de presentación óptima y sólida al mercado laboral ya que evidencio conocimientos y buenas prácticas.

# Competencia del Perfil de Egreso

En esta parte del informe se mostrarán nuestras competencias de perfil de egreso que se complementan al proyecto propuesto:

* Desarrollar una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.
* Implementar soluciones sistémicas integrales para automatizar y optimizar procesos de negocio de acuerdo a las necesidades de la organización.
* Resolver las vulnerabilidades sistémicas para asegurar que el software construido cumple las normas de seguridad exigidas por la industria.
* Gestionar proyectos informáticos, ofreciendo alternativas para la toma de decisiones de acuerdo a los requerimientos de la organización.

# Intereses Profesionales

Aquí podrán ver cuáles son nuestros intereses profesionales:

En primera parte se relaciona en función a mi capacidad para identificar los requerimientos esenciales propuestos a la problemática del desarrollo de software y la visualización de su arquitectura asegurando su integridad y calidad, la entrega de documentación hablando de sus funciones específicas y la entrega de estos acordes a fechas establecidas. De esta forma realizando una documentación que sea fácil de comprender, que sea clara y concisa, facilitando la lectura para los desarrolladores tanto como para los supervisores y cliente en cuestión. El análisis correcto de los requerimientos nos permitirá la optimización de recursos y buenos ingresos a la empresa.

# Argumentos sobre la factibilidad del proyecto dentro de la asignatura (Capstone)

**Alcance Modular y Escalable**

El proyecto se puede dividir en módulos (boletas, stock, dashboard, recomendaciones de optimización) que encajan perfectamente en el cronograma típico de un proyecto final o Capstone. Cada módulo ofrece objetivos de desarrollo e hitos mensurables, lo que facilita su gestión durante el semestre o periodo académico.

**Recursos Tecnológicos Asequibles**

Las tecnologías empleadas (React, Node.js, Django y PostgreSQL) son de código abierto y cuentan con abundante documentación y comunidades de soporte. No se requiere infraestructura de pago obligatoria: es factible desarrollar en un entorno local, con posibilidad de desplegar en servicios gratuitos o de bajo costo (p. ej., Heroku, Railway, etc.) para la demostración.

**Proceso de Desarrollo Ágil**

La adopción de metodologías ágiles (Scrum) permite organizar el proyecto en Sprints, obteniendo incrementos funcionales en plazos cortos. Esto promueve la retroalimentación temprana de tutores y compañeros, reduciendo el riesgo de desviarse de los requisitos y asegurando un avance constante.

**Perfil Multidisciplinario Alineado con la Asignatura**

Un proyecto Capstone suele exigir la aplicación integral de los conocimientos adquiridos en la carrera:

Desarrollo web (frontend y backend)

Manejo de bases de datos

Gestión de proyectos (Scrum)

Diseño de interfaces y experiencia de usuario

El proyecto APT permite a los estudiantes demostrar estas competencias de manera práctica y sincronizada.

**Posibilidad de Validación con Usuarios Reales**

El contacto directo con una empresa interesada en la solución garantiza feedback real y casos de uso concretos. Esto incrementa la relevancia del proyecto y cumple con el espíritu de un Capstone, que busca integrar la academia con un escenario laboral auténtico.

**Escenarios para Pruebas y Evaluaciones**

El alcance funcional (boletas, stock, dashboard) brinda oportunidades claras de evaluación y demostración:

Análisis de requerimientos

Implementación técnica

Validación de interfaz (usabilidad)

Rendimiento y calidad de datos

Cada uno de estos puntos puede ser valorado por los profesores o mentores dentro de la materia, lo cual muestra la factibilidad en términos de aprendizaje y evidencias de desempeño.

**Tiempo de Desarrollo Adecuado**

El proyecto, aunque ambicioso, tiene un tamaño razonable para completarse durante un semestre o año académico, siempre y cuando se planifique en etapas y se haga uso de buenas prácticas de gestión de tiempo. El cronograma del Capstone (típicamente entre 4 y 6 meses) es compatible con el diseño, la implementación y las pruebas finales de un sistema web de este tipo.

# Objetivo General

Desarrollar una plataforma web integral que permita optimizar y gestionar de manera eficiente los procesos de una vulcanización, incluyendo emisión de boletas, control de stock y administración de ventas, incorporando un *dashboard inteligente* para la toma de decisiones en tiempo real.

## **Objetivos Específicos:**

**Sistema de administración de boletas:** Integrar la emisión, registro y seguimiento de boletas de venta.  
Garantizar el cumplimiento de requisitos tributarios y normativas legales.  
Simplificar y agilizar la facturación para mejorar la experiencia de clientes y administradores.

**Control de stock:** Permitir el registro y actualización en tiempo real de existencias de repuestos y productos.  
Notificar sobre bajos niveles de inventario para facilitar la reposición oportuna.  
Generar reportes de rotación de productos, tendencias de uso y costos asociados.

***Dashboard inteligente:*** Ofrecer visualizaciones de datos diarios, semanales, mensuales y anuales para ventas y operaciones. Facilitar la detección de patrones y tendencias de consumo. Proveer recomendaciones basadas en métricas clave para la toma de decisiones estratégicas.

# Propuesta de Metodología: Enfoque Ágil

Para abordar el problema identificado en el Proyecto, utilizaremos una metodología ágil enfocado en el desarrollo de las funciones principales de la aplicación dentro de un periodo de 4 meses, se puede dar este enfoque en sprint, cada sprint está cuidadosamente estructurada para cumplir con los plazos, garantizar la calidad del producto, y alinear el desarrollo con los objetivos del proyecto.

Estamos utilizando esta metodología porque es la que mejor se adapta a nuestro plan de trabajo y porque fue recomendada debido al enfoque que emplearemos. Nos brinda diversos beneficios, los cuales también consideramos al decidir implementar.

Beneficios:

1. **Flexibilidad**: Facilita la adaptación a cambios en los requisitos.
2. **Entrega continua**: Proporciona valor rápidamente con entregas frecuentes.
3. **Mejora de la comunicación**: Fomenta la colaboración constante entre equipos y clientes.
4. **Satisfacción del cliente**: Asegura que el producto cumpla con las expectativas mediante retroalimentación temprana.
5. **Reducción de riesgos**: Identifica y soluciona problemas de forma rápida.
6. **Motivación del equipo**: Promueve la autonomía y el compromiso del equipo.
7. **Mejora continua**: Ajusta procesos tras cada sprint para optimizar el rendimiento.

## Metodología Ágil para el Proyecto APT (4 Meses)

### Sprint 0: Preparación y Configuración Inicial (Semana 1)

* **Objetivo**: Establecer las bases del proyecto mediante la definición inicial de requisitos y la configuración del entorno de desarrollo.
* **Actividades**:
  + Crear el Product Backlog con historias de usuario preliminares.
  + Configurar el entorno de desarrollo, repositorios, y herramientas de gestión.
  + Realizar un análisis de viabilidad técnica y definir tecnologías clave.
* **Entregables**: Product Backlog inicial, entorno de desarrollo configurado y cronograma de sprints.

### Sprint 1: Planificación y Diseño (Semanas 2-3)

* **Objetivo**: Definir y priorizar los requerimientos clave del sistema y diseñar la arquitectura básica.
* **Actividades**:
  + Realizar el Kick Off y redactar el Acta de Constitución del proyecto.
  + Refinar y priorizar el Product Backlog junto con el Product Owner.
  + Diseñar la arquitectura inicial del sistema (front-end, back-end, base de datos).
  + Crear prototipos de la interfaz de usuario (wireframes) y definir los mockups.
* **Entregables**: Acta de Constitución, Product Backlog refinado, arquitectura inicial del sistema, y prototipos de interfaz.

### Sprint 2: Desarrollo de Módulos Principales (Semanas 4-5)

* **Objetivo**: Iniciar el desarrollo de las funcionalidades principales del sistema, enfocándose en los módulos base de autenticación y estructura de la base de datos.
* **Actividades**:
  + Implementar el ambiente de desarrollo completo.
  + Desarrollar el módulo de inicio de sesión y registro.
  + Crear la base de datos inicial y su estructura.
  + Realizar pruebas unitarias para los módulos desarrollados.
* **Entregables**: Módulo de autenticación, estructura de la base de datos, y pruebas unitarias iniciales.

### Sprint 3: Desarrollo de Funcionalidades y APIs (Semanas 6-8)

* **Objetivo**: Continuar el desarrollo del sistema, implementando el front-end y funcionalidades de integración.
* **Actividades**:
  + Crear las páginas principales y su conexión con el back-end.
  + Desarrollar e integrar las APIs.
  + Realizar pruebas unitarias para el front-end y cada API.
  + Integrar de manera parcial el sistema en el entorno de desarrollo.
* **Entregables**: Front-end funcional, APIs integradas, pruebas unitarias y módulo parcialmente integrado.

# Plan de trabajo

A continuación, se detalla la planificación del proyecto APT, especificando las competencias y actividades claves necesarias para cumplir con los objetivos del proyecto:

| **Competencias** | **Nombre de Actividad** | **Descripción de Actividades** | **Sprint asociado** | **Recursos** | **Duración de la Actividad** | **Responsable** | **Observaciones** |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Utiliza metodologías ágiles para planificar y ejecutar proyectos de software. | Definición de requerimientos | Recopilar y priorizar las funcionalidades esenciales del sistema (boletas, stock, dashboard inteligente), identificando objetivos y alcance del proyecto con la empresa interesada. | Sprint 1 | Documentos de levantamiento de requerimientos, reuniones de retroalimentación. | 5 días hábiles | Product Owner & Equipo de Análisis | Es importante establecer criterios de aceptación claros para cada requerimiento. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Aplica principios de arquitectura de software para diseñar soluciones escalables y mantenibles. | Diseño de arquitectura | Elaborar diagramas de arquitectura (frontend con React/Node.js, backend con Django/Python y base de datos PostgreSQL), definiendo la interacción entre cada capa. | Sprint 1 | Herramientas de modelado (Lucidchart, Draw.io, etc.) | 3 días hábiles | Arquitecto del proyecto / Equipo de Desarrollo | Validar con el equipo la viabilidad técnica y compatibilidad con los lineamientos de la empresa. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Configura entornos de desarrollo y asegura la correcta integración de las herramientas seleccionadas. | Configuración del entorno de trabajo | Instalar y poner a punto Node.js, Django, PostgreSQL. Crear repositorios (Git) y pipelines de integración continua (CI/CD) si fuera requerido. | Sprint 2 | Computadores, acceso a repositorios, documentación de instalación | 4 días hábiles | Líder de DevOps / Equipo de Desarrollo | Asegurarse de que todo el equipo tenga acceso y un entorno homogéneo para evitar problemas de compatibilidad. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Diseña y desarrolla módulos de software orientados a satisfacer requerimientos específicos de la empresa. | Desarrollo del módulo de boletas | Implementar la lógica para la generación, registro y visualización de boletas. Incluir validaciones y actualizaciones en la base de datos, junto con pruebas funcionales básicas. | Sprint 3 | Editor de código (VS Code), Django, PostgreSQL, Postman | 7 días hábiles | Equipo de Backend / Full Stack | Es fundamental asegurar el cumplimiento de requisitos legales/tributarios según normativas aplicables. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Realiza pruebas de validación para asegurar la calidad y confiabilidad de los módulos desarrollados. | Pruebas de integración | Probar la interoperabilidad entre el módulo de boletas, el control de stock y el dashboard (versión preliminar) para garantizar un flujo de trabajo coherente y sin errores críticos. | Sprint 4 | Frameworks de prueba (pytest, unittest), Postman, herramientas de debugging | 5 días hábiles | QA / Equipo de Pruebas | Documentar cualquier hallazgo y generar reportes de errores. Priorizar la corrección de fallos críticos antes de avanzar al siguiente sprint. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Analiza datos y presenta información de forma comprensible para la toma de decisiones. | Implementación del Dashboard | Implementar consultas y visualizaciones de ventas diarias, semanales, mensuales y anuales. Agregar un módulo de recomendaciones para maximizar ganancias. | Sprint 5 | Librerías de visualización, ORM de Django, herramienta de reporting | 6 días hábiles | Equipo de Backend / Frontend | Validar con el usuario final que los indicadores y gráficas respondan a sus necesidades. Ajustar el diseño de ser necesario. |
| --- | --- | --- | --- | --- | --- | --- | --- |

| Documenta y presenta el proyecto, garantizando la transferencia de conocimiento y la posibilidad de mantenimiento a futuro. | Documentación y entrega final | Redactar manuales de usuario, manual técnico y plan de mantenimiento. Presentar la plataforma operativa al cliente y/o institución académica. | Sprint 6 | Herramientas de documentación (Word, Confluence) | 5 días hábiles | Líder de Proyecto / Equipo de Desarrollo | Asegurar que el cliente reciba capacitación suficiente para usar y administrar la plataforma. |
| --- | --- | --- | --- | --- | --- | --- | --- |

A continuación, se presentan los facilitadores y obstaculizadores en el desarrollo de las actividades:

| **Nombre de Actividades** | **Facilitadores** | **Obstaculizadores** |
| --- | --- | --- |

| **Definición de requerimientos** | - Comunicación fluida con la empresa interesada. - Documentación previa (procesos, Excel) que orienta la identificación de necesidades. | - Requerimientos ambiguos o en constante cambio. - Falta de claridad en las prioridades definidas por los stakeholders. |
| --- | --- | --- |

| **Diseño de arquitectura** | - Conocimiento de patrones de diseño y buenas prácticas de software. - Disponibilidad de herramientas de modelado (Lucidchart, Draw.io). | - Limitaciones técnicas o de infraestructura que no fueron detectadas previamente. - Dificultad para alinear distintos enfoques del equipo. |
| --- | --- | --- |

| **Configuración del entorno de trabajo** | - Documentación clara de instalación (Node.js, Django, PostgreSQL). - Experiencia previa del equipo en entornos similares. | - Falta de acceso o permisos necesarios en servidores o servicios externos. - Diferencias de versiones y dependencias que generan conflictos. |
| --- | --- | --- |

| **Desarrollo del módulo de Boletas** | - Requerimientos funcionales bien definidos para la emisión y registro de boletas. - Uso de un marco de trabajo ágil que permita iterar rápido. | - Falta de requisitos legales o tributarios actualizados. - Baja experiencia del equipo en la integración de pagos o facturación electrónica, si aplica. |
| --- | --- | --- |

| **Pruebas de integración** | - Herramientas de prueba automatizadas (Postman, pytest, etc.). - Un código base coherente, siguiendo convenciones y buenas prácticas. | - Datos de prueba desorganizados o incoherentes. - Cambios frecuentes en los servicios o endpoints sin adecuada sincronización. |
| --- | --- | --- |

| **Implementación del Dashboard** | - Claridad en las métricas clave que se desean mostrar (ingresos, egresos, stock). - Librerías de visualización maduras y compatibles con React. | - Datos históricos incompletos o inconsistentes que dificultan la generación de reportes. - Requerimientos de visualización que cambian de forma imprevista. |
| --- | --- | --- |

| **Documentación y entrega final** | - Uso de plantillas y estándares de documentación (manuales de usuario, manuales técnicos). - Retroalimentación constante del cliente. | - Falta de tiempo para redactar documentos finales. - Cambios de último minuto que alteran la versión final de la aplicación. |
| --- | --- | --- |

# Determinación de evidencias y su justificación

A continuación, se presenta una tabla de evidencias con su tipo, nombre, descripción y justificación, siguiendo el formato provisto. Estas evidencias servirán para demostrar el progreso en cada fase y el cumplimiento de los objetivos del proyecto.

| **Tipo de evidencia (avance o final)** | **Nombre de la evidencia** | **Descripción** | **Justificación** |
| --- | --- | --- | --- |

| **Avance** | Prototipo inicial de la Interfaz (React) | Muestra las pantallas principales de la plataforma: registro de boletas, visualización de stock y diseño preliminar del dashboard. Incluye navegación básica entre secciones. | Permite validar tempranamente la usabilidad y el flujo de usuario, comprobando la coherencia del diseño antes de avanzar con el desarrollo completo. |
| --- | --- | --- | --- |

| **Avance** | Configuración funcional del Backend (Django) | Presenta la estructura del proyecto Django con los modelos y endpoints iniciales (p. ej., boletas, productos), junto con pruebas unitarias básicas. | Evidencia que el proyecto tiene una base sólida, asegurando que la lógica de negocio y la persistencia de datos funcionen en la etapa temprana. |
| --- | --- | --- | --- |

| **Avance** | Dashboard de Datos en Desarrollo | Prototipo de gráficas y reportes parciales (p. ej., ventas diarias, alertas de bajo stock) con la integración entre Django y la base de datos PostgreSQL. | Permite verificar que se están capturando y procesando los datos correctamente, y que existe un avance concreto hacia la funcionalidad de análisis y toma de decisiones. |
| --- | --- | --- | --- |

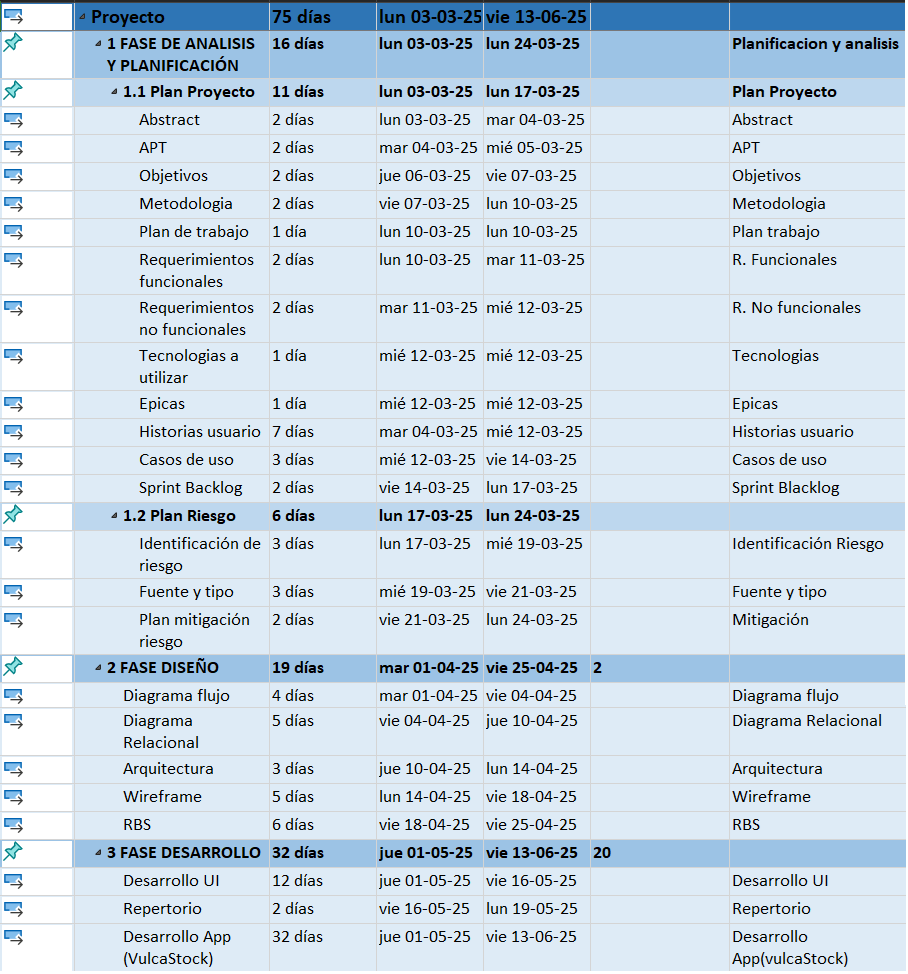
| **Final** | Plataforma Web Operativa (Versión Beta) | Presentación de la versión casi completa del sistema, con funcionalidades de boletas, stock y dashboard integrado, accesible desde un entorno de pruebas o servidor temporal. | Demuestra la **completitud funcional** en un entorno cercano a producción, permitiendo al usuario final probar y validar la solución antes del lanzamiento oficial. |
| --- | --- | --- | --- |

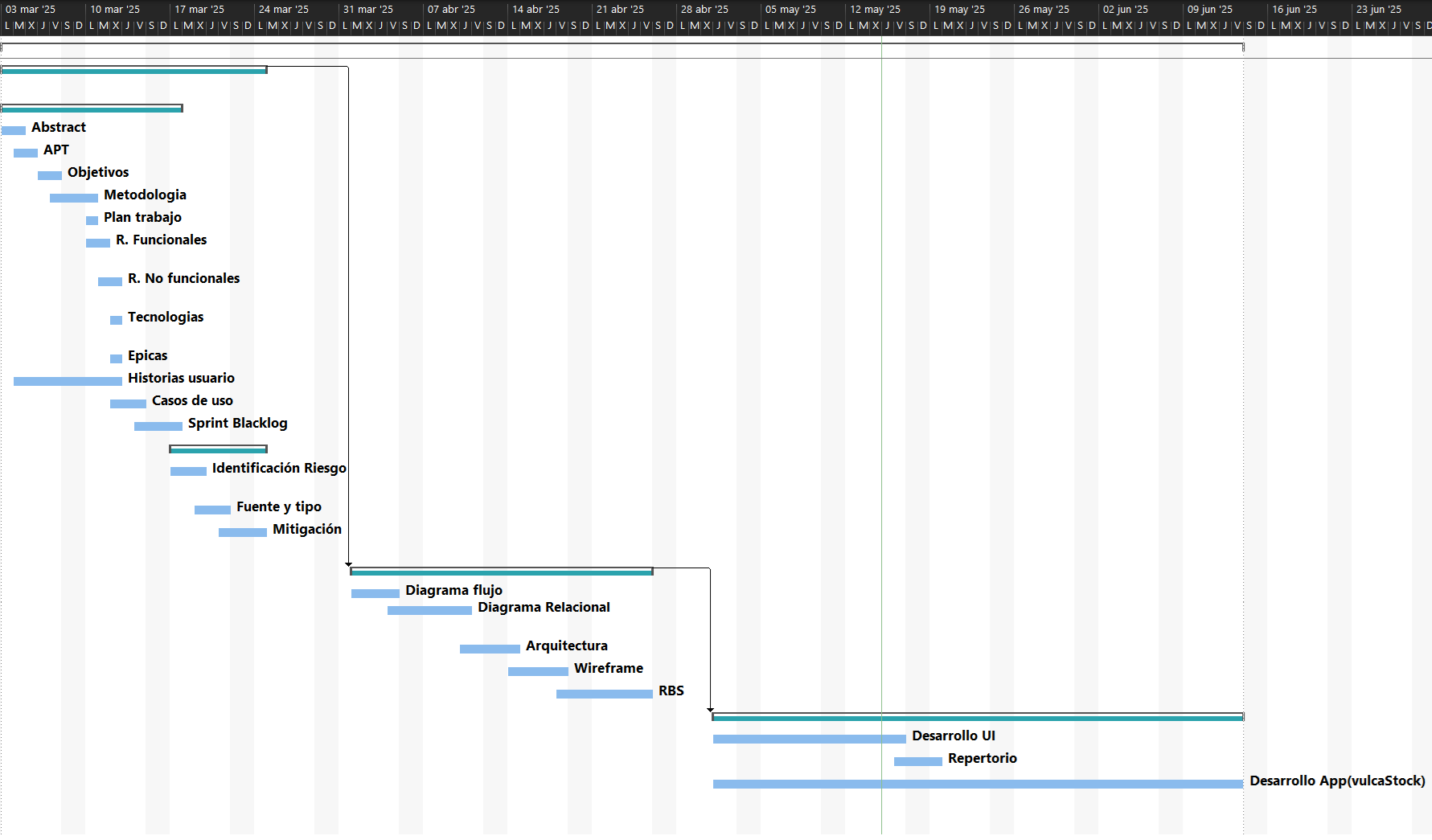
| **Final** | Informe de Cierre y Documentación | Entrega de un documento que incluye el manual de usuario, la descripción técnica (arquitectura, modelos de datos, endpoints) y el registro de pruebas realizadas (unitarias, de integración). | Garantiza la **transferencia de conocimiento** y el respaldo de toda la información necesaria para futuros mantenimientos, expansiones o modificaciones del proyecto. |
| --- | --- | --- | --- |

| **Final** | Entrega de Resultados y Retroalimentación | Sesión de demostración frente a la empresa interesada y el equipo académico, recopilando feedback final y evidencia de que se cumplieron los objetivos planteados. | Asegura el **cierre formal** del proyecto, permitiendo documentar mejoras y la conformidad del cliente, así como la alineación con los requisitos académicos (Capstone). |
| --- | --- | --- | --- |

# Carta Gantt

El cronograma del proyecto “VulckaStock” ha sido creado para garantizar el desarrollo eficiente, dividido en 3 fases las cuales cubren desde la planificación y análisis hasta la fase de desarrollo. Estas fases permiten gestionar el tiempo de manera óptima, asegurando que cada sprint se realice de manera organizada. A continuación, se presenta la carta Gantt.





# Requerimientos funcionales

| **Código** | **Nombre** | **Actor Relacionado** | **Descripción** | **Prioridad** |
| --- | --- | --- | --- | --- |

| RF-1 | Emisión de Boletas | Cajero / Usuario | El sistema debe permitir la creación y emisión de boletas, registrando fecha, hora, productos/servicios vendidos y valor total de la transacción. | Alta |
| --- | --- | --- | --- | --- |

| RF-2 | Control de Stock | Administrador | Debe existir una sección para el ingreso, actualización y visualización de inventario, incluyendo alertas de stock bajo en tiempo real. | Alta |
| --- | --- | --- | --- | --- |

| RF-3 | Dashboard de Ventas | Gerente / Usuario | El sistema mostrará estadísticas de ventas diarias, semanales y mensuales, con posibilidad de filtrar por rango de fechas, permitiendo la toma de decisiones. | Media |
| --- | --- | --- | --- | --- |

| RF-4 | Gestión de Usuarios | Administrador | La plataforma debe permitir registrar nuevos usuarios, asignar roles y administrar privilegios (visualizar, editar, emitir boletas, etc.). | Media |
| --- | --- | --- | --- | --- |

| RF-5 | Generación de Reportes Exportables | Gerente / Administrador | El sistema debe contar con la opción de exportar los reportes generados (boletas emitidas, estado de stock, etc.) en formatos comunes (PDF, Excel). | Baja |
| --- | --- | --- | --- | --- |

# Requerimientos no funcionales

| **Código** | **Nombre** | **Actor Relacionado** | **Descripción** | **Prioridad** |
| --- | --- | --- | --- | --- |

| RNF-1 | Rendimiento | Todos | El sistema debe responder a cualquier petición (emisión de boletas, consultas de stock, etc.) en un tiempo máximo de 3 segundos bajo carga media. | Alta |
| --- | --- | --- | --- | --- |

| RNF-2 | Seguridad de Datos | Administrador / TI | Debe implementarse un protocolo de autenticación y autorización seguro (HTTPS, tokens o similar) para proteger la información de ventas, boletas y usuarios. | Alta |
| --- | --- | --- | --- | --- |

| RNF-3 | Escalabilidad | Gerente / TI | El diseño de la plataforma debe permitir añadir más usuarios, mayores volúmenes de datos y nuevos módulos sin que se vea afectado el rendimiento de manera significativa. | Media |
| --- | --- | --- | --- | --- |

| RNF-4 | Usabilidad de la Interfaz | Usuario | La interfaz debe ser intuitiva, con menús claros y consistentes, de modo que un usuario con conocimientos básicos pueda realizar las operaciones sin requerir formación extensa. | Media |
| --- | --- | --- | --- | --- |

| RNF-5 | Mantenibilidad | Equipo de Desarrollo | El código y la arquitectura deben estar documentados y organizados, de manera que resulte fácil actualizar, corregir errores o agregar nuevas funcionalidades. | Baja |
| --- | --- | --- | --- | --- |

# Tecnologías a utilizar

A continuación, se describen las principales tecnologías que se emplearán en el proyecto, destacando sus características clave y la forma en que contribuyen al desarrollo de la plataforma.

### React

Es una librería de JavaScript enfocada en la construcción de interfaces de usuario (UI) interactivas y modulares.

* **Componentes Reutilizables**: Facilita la creación de interfaces basadas en bloques independientes, lo que agiliza el desarrollo y la mantenibilidad del código.
* **Virtual DOM**: Optimiza el rendimiento al actualizar solo los elementos que han cambiado, en lugar de toda la página.
* **Ecosistema Amplio**: Dispone de un gran número de librerías y herramientas (React Router, Redux, etc.) que cubren diversas necesidades de desarrollo web.

### Node.js

Node.js es un entorno de ejecución de JavaScript orientado a servidores, utilizado para gestionar dependencias, scripts de desarrollo y, en ocasiones, para crear microservicios o APIs.

* **Asincronía**: Permite manejar múltiples operaciones concurrentes sin bloquear el hilo principal.
* **NPM (Node Package Manager)**: Ofrece acceso a una gran variedad de paquetes y librerías open source.
* **Versatilidad**: Puede usarse tanto para tareas de build (compilación de la app React) como para la implementación de servicios backend ligeros (Express).

### Django (Python)

Framework web de alto nivel y de código abierto basado en Python, que promueve el desarrollo rápido, seguro y escalable.

* **Patrón MTV (Model-Template-View)**: Separa la lógica de negocio, la presentación y el manejo de datos.
* **ORM (Object-Relational Mapping)**: Facilita el acceso a la base de datos sin necesidad de escribir consultas SQL manuales constantemente.
* **Seguridad Integrada**: Incluye protecciones contra inyección SQL, cross-site scripting (XSS) y otros ataques comunes.

### PostgreSQL

Un **sistema de gestión de bases de datos relacional** de código abierto, conocido por su confiabilidad y robustez.

* **Estándares ACID**: Garantiza transacciones seguras y consistentes, esenciales para operaciones financieras y de inventario.
* **Escalabilidad**: Maneja eficientemente grandes volúmenes de datos y puede ampliarse según la demanda.
* **Funciones Avanzadas**: Soporta tipos de datos complejos (JSON, arrays) y ofrece la posibilidad de crear vistas y funciones propias.

### Herramientas de Integración y Colaboración

* **Git / GitHub o GitLab**: Sistemas de control de versiones y repositorios remotos para la colaboración y seguimiento de cambios.
* **Postman**: Ayuda a probar y documentar los endpoints de la API, asegurando que las peticiones y respuestas funcionen correctamente.

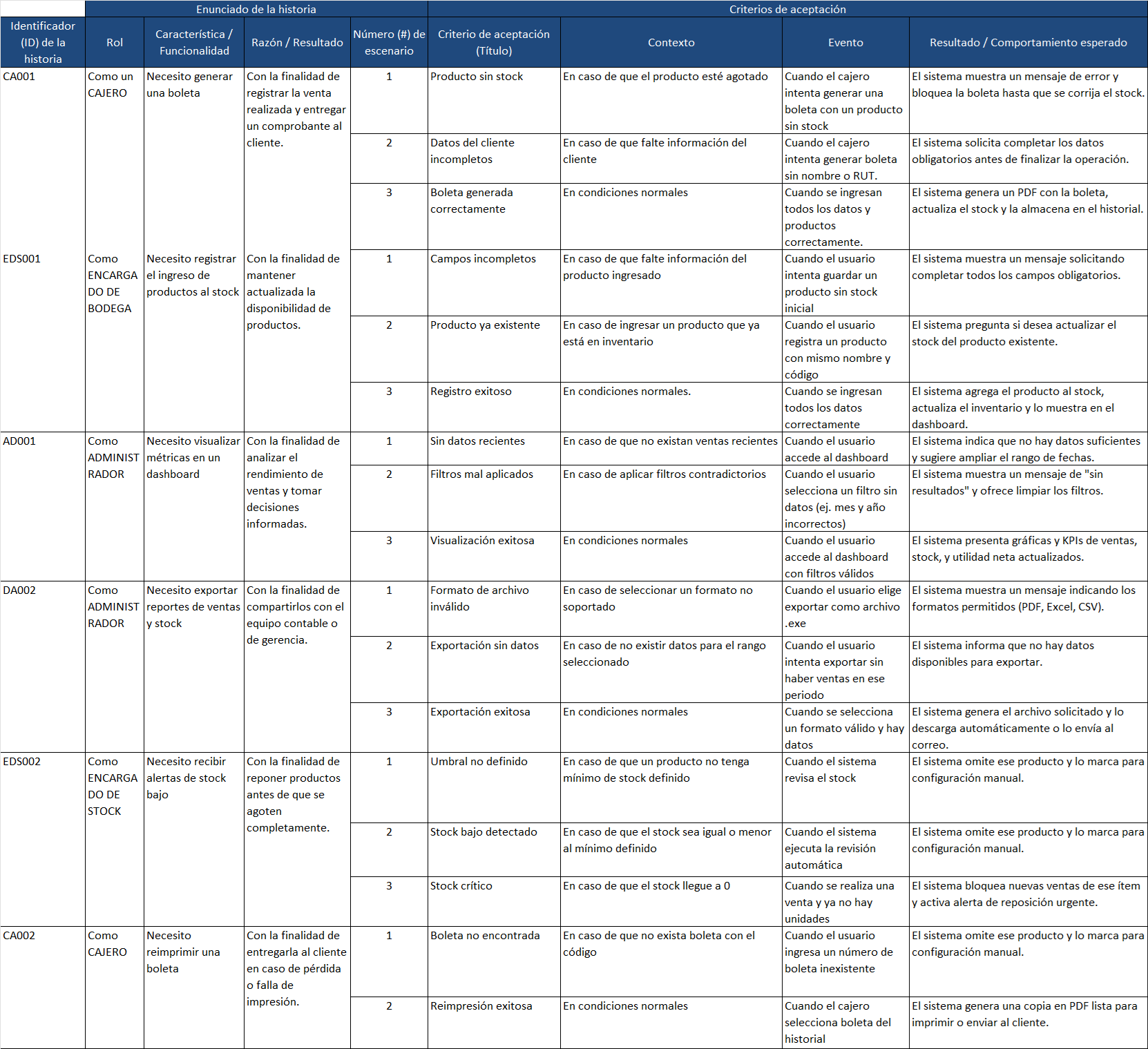
# Definición de épicas

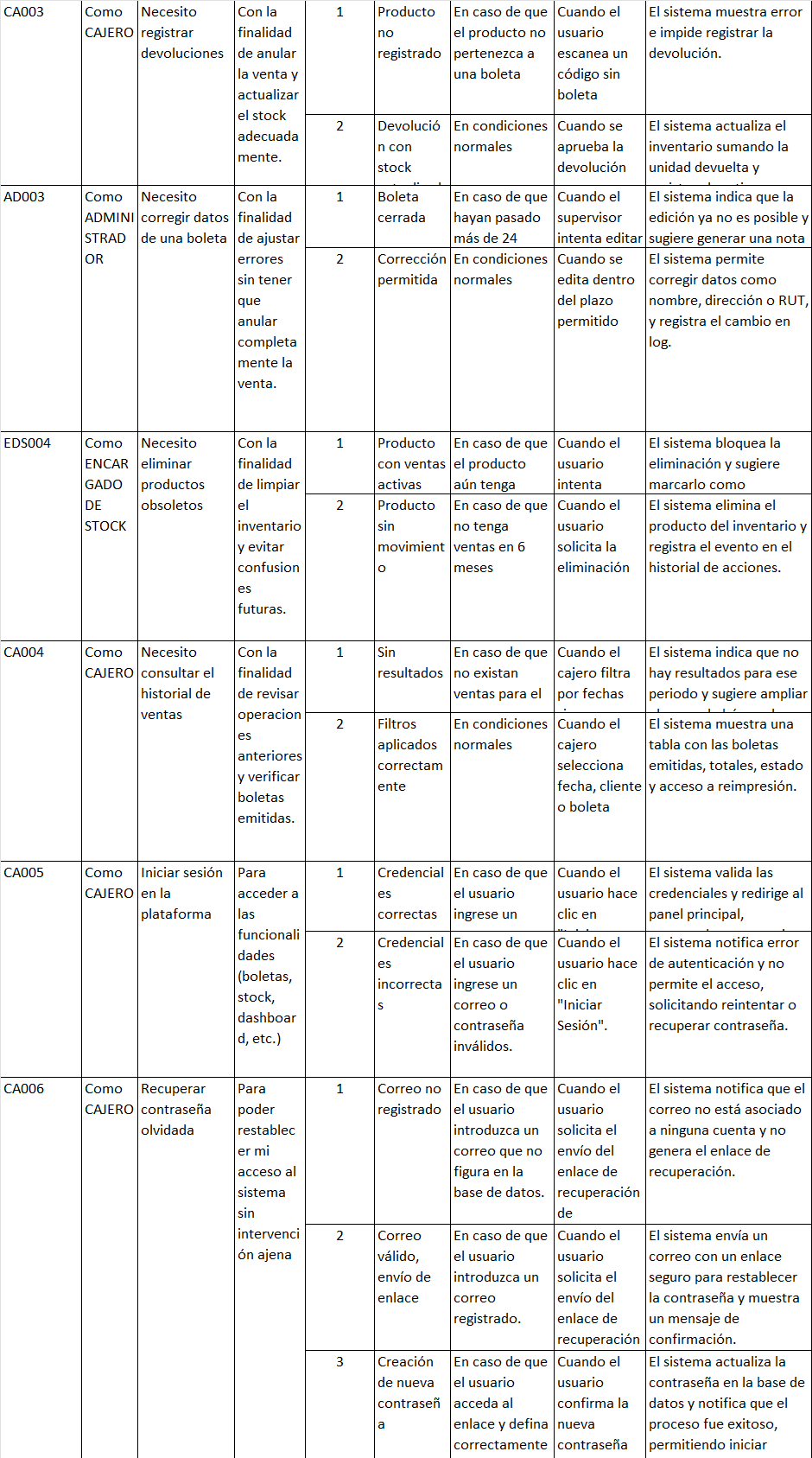
Se definieron épicas, que representan componentes importantes del proyecto, agrupando funcionalidades o necesidades claves. Estas épicas se dividieron en historias de usuario, lo que permite un desarrollo ordenado y una gestión eficiente del proyecto.

| Código | Epica | Descripción |
| --- | --- | --- |
| E1 | Epica 1 | Login |
| E2 | Epica 2 | Descripción y Funcionalidades |

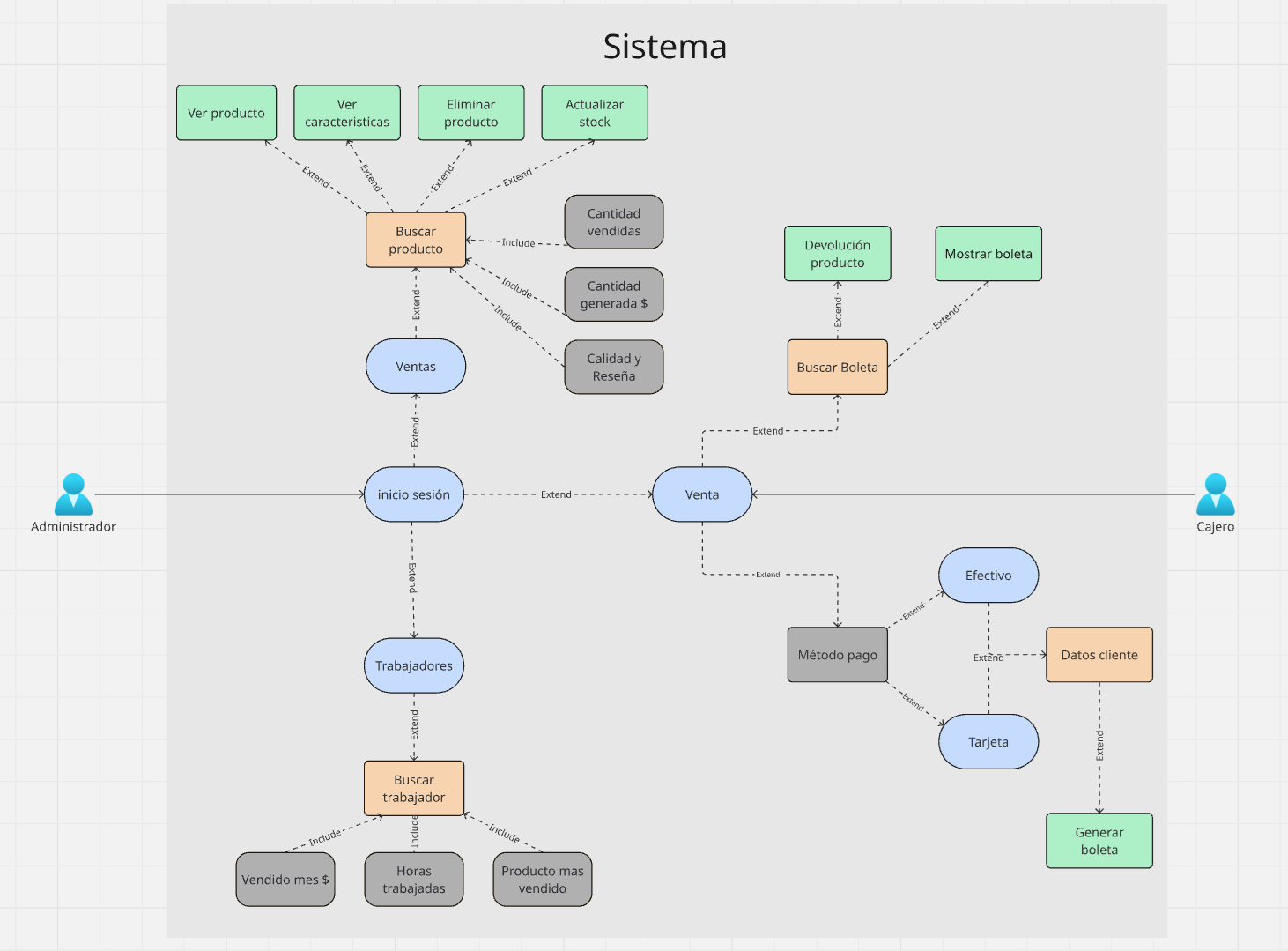
# Historias de usuarios

Las historias de usuario son una herramienta fundamental en el desarrollo de software, ya que permiten expresar de forma clara y sencilla las necesidades y expectativas del usuario. Cada historia describe una funcionalidad o característica del sistema desde la perspectiva del usuario final, lo que mejora la comunicación entre el equipo de desarrollo y los stakeholders del proyecto.





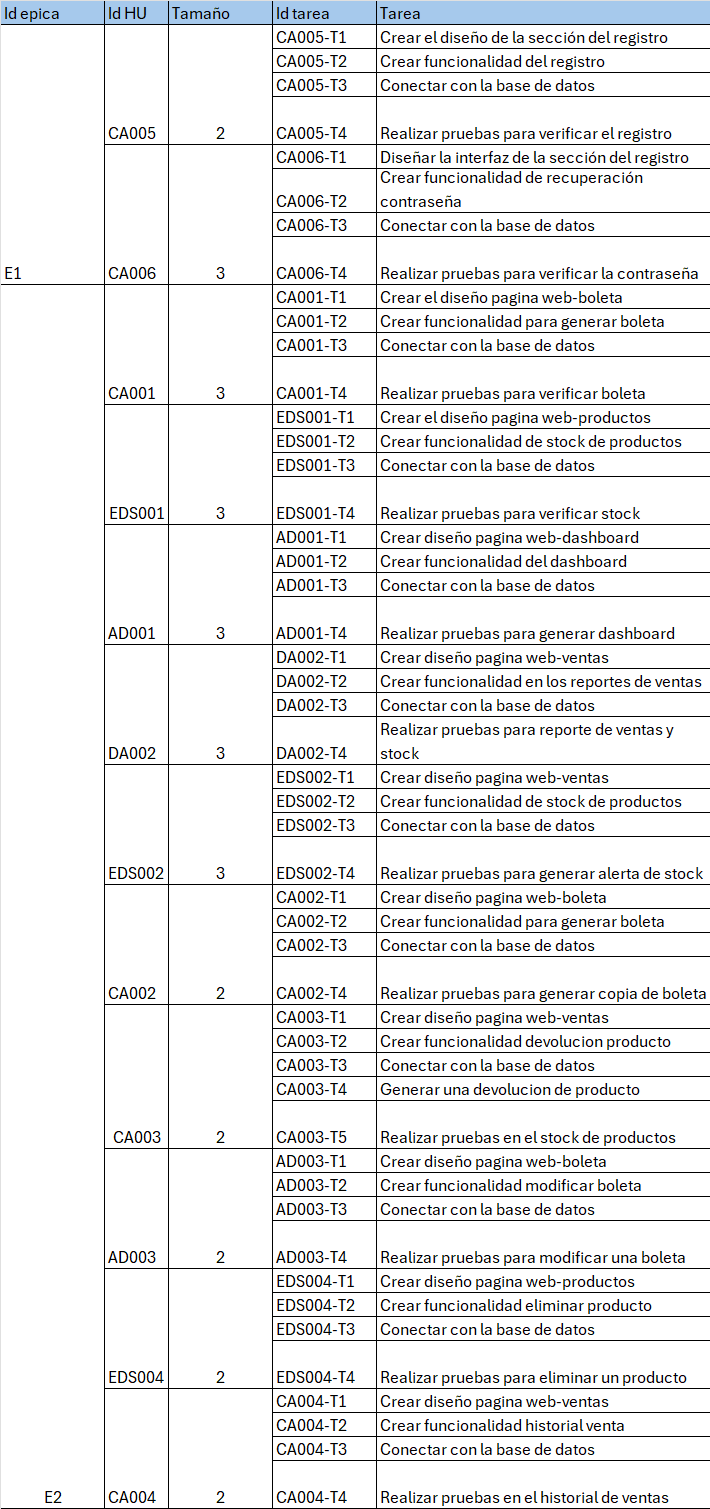
# Casos de uso

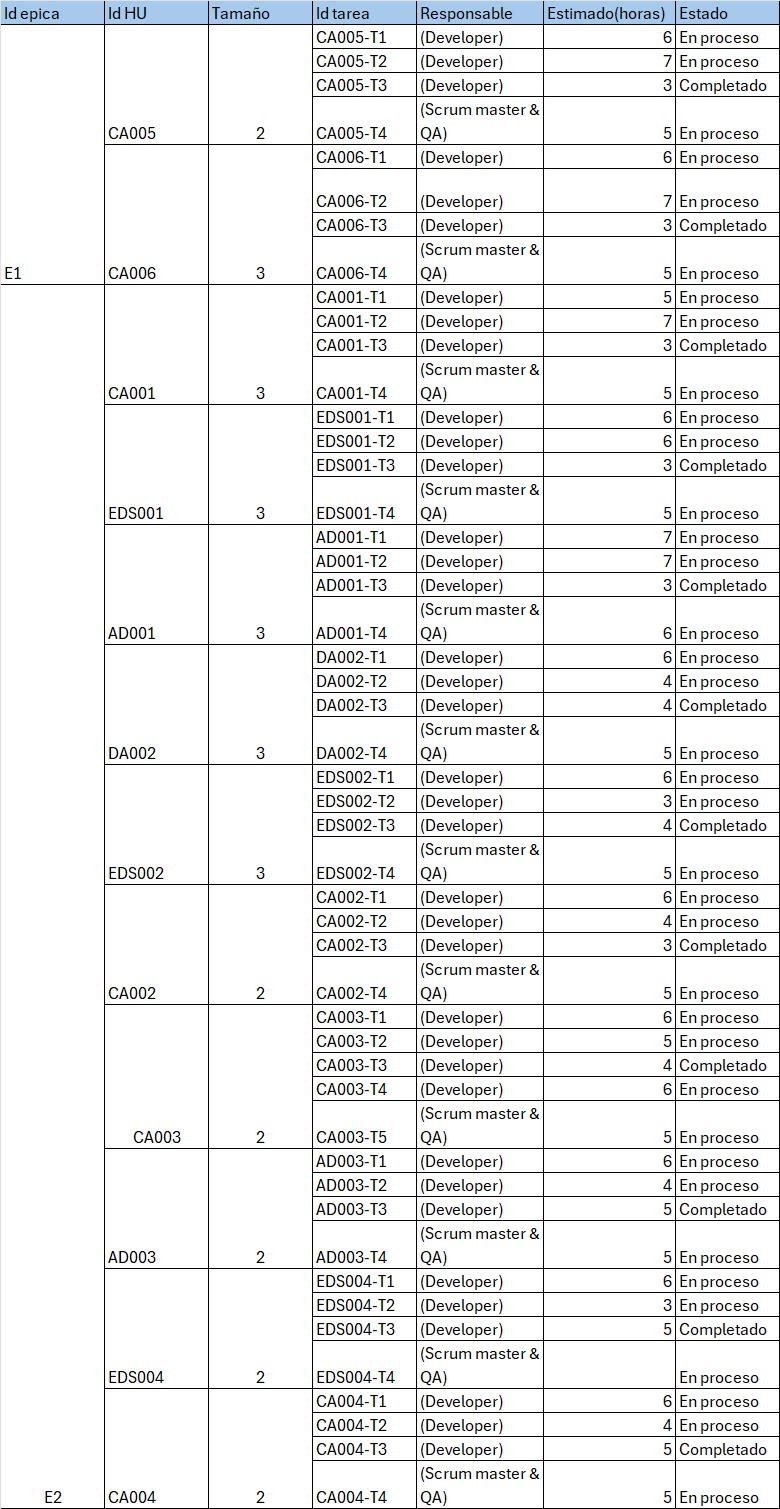


# Sprint Backlog

El sprint backlog junta las épicas seleccionadas para este proyecto, junto con sus respectivas historias de usuario y las tareas necesarias para completarlas. A cada historia se le asigna un responsable y se estima la cantidad de horas requeridas por tarea. También se realiza un seguimiento del estado de cada tarea para asegurar un control adecuado del avance del proyecto.

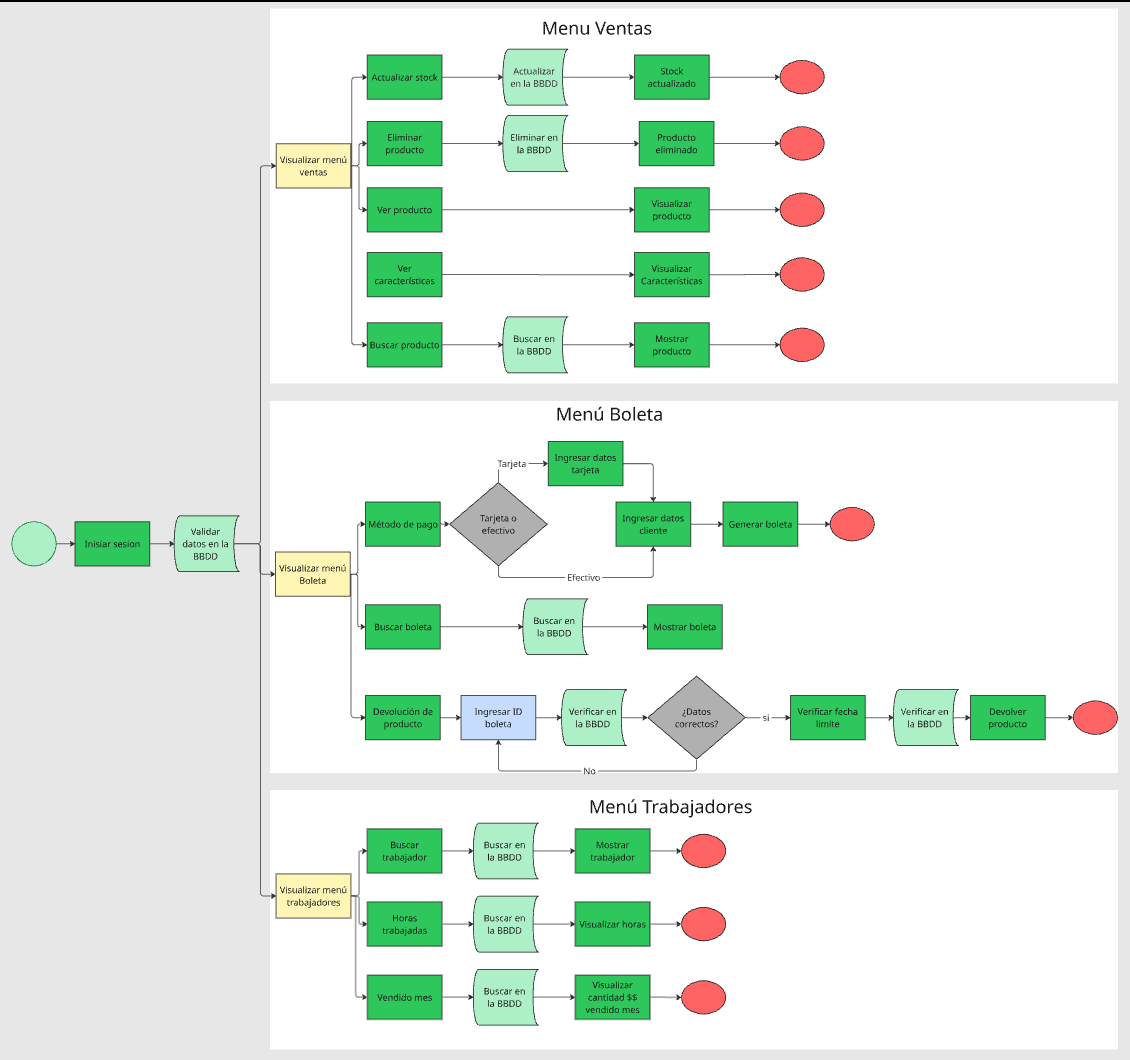
Para facilitar la lectura, esta información se presenta en dos tablas: la primera muestra las épicas, los identificadores de las historias de usuario y las tareas asociadas con sus respectivas descripciones; la segunda tabla incluye también los identificadores de épicas, historias y tareas, así como el responsable asignado, la estimación de horas y el estado actual de cada tarea.





# Diseño del diagrama de flujo

En esta sección se puede observar con más detalle el flujo del funcionamiento del sistema.



# Diseño diagrama relacional

El diagrama relacional de la base de datos del proyecto “VulcaStock” fue diseñado para administrar de manera eficiente la información de los usuarios y sus interacciones dentro de la aplicación. A continuación, se presenta una descripción de las tablas principales:

* Usuario: Esta tabla almacena la información básica de cada usuario, que incluye un identificador, nombre, correo, contraseña y fecha de registro. Además, se relaciona con la tabla Vehiculo, Boleta e Historial inventario.
* Vendedor: Esta tabla guarda la información de los vendedores, la cual incluye un identificador, nombre, email, dirección y fecha\_ingreso. Además, se relaciona con la tabla boleta.
* Proveedor: Esta tabla contiene la información de los proveedores del negocio, la cual incluye un identificador, nombre, contacto, teléfono, email, dirección. Además, se relaciona con la tabla producto.
* Vehículo: Esta tabla contiene la información de todos los vehículos que ingresa al negocio, la cual incluye un identificador, marca, modelo, año y kilometraje. Además, se relaciona con la tabla Usuario.
* Boleta: Esta tabla contiene la información de las boletas emitidas en el negocio, la cual incluye un identificador, fecha\_venta, total, estado y tipo. Además, se relaciona con la tabla Usuario, Pago, Detalle\_boleta y Vendedor.
* Servicio: Esta tabla contiene la información sobre los servicios que se ofrecen en el negocio, la cual incluye un identificador, nombre, descripción, precio\_base y duración\_estimada. Además, está relacionada con la tabla Detalle\_boleta.
* Producto: Esta tabla contiene la información sobre los productos que se almacenan en el negocio, la cual incluye un identificador, nombre, descripción, precio, cantidad\_disponible, tipo, marca, es\_alternativo y fecha\_agregado. Además se relaciona con la tabla Proveedor, Detalle\_boleta e Historial\_inventario.
* Pago: Esta tabla contiene toda la información sobre los pagos realizados en las ventas del negocio, la cual incluye fecha\_pago, monto, método\_pago y estado. Además se relaciona con la tabla Boleta.
* Detalle\_boleta: Esta tabla contiene la información sobre el detalle de las boletas emitidas en el negocio, la cual incluye un identificador, cantidad, precio\_unitario y subtotal. Además, se relaciona con la tabla Boleta, Servicio y Producto.
* Historial\_inventario: Esta tabla contiene la información sobre el historial de inventarios del negocio, la cual incluye un identificador, tipo\_movimiento y fecha. Además, se relaciona con la tabla Usuario y Producto.

