PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA INGENIERÍA EN SISTEMAS DE LA INFORMACIÓN

Trabajo de Titulación

TEMA: DESARROLLO DE UN PROTOTIPO DE SISTEMA WEB PARA LA GESTIÓN DE PROCESOS CRÍTICOS DEL CONTROL DE INVENTARIOS.

CASO DE ESTUDIO: TIENDA DE SUPLEMENTOS DEPORTIVOS "NUTRIBOOST"

AUTOR:

STEVEN ALEXIS MENDOZA REZA

TUTOR:

JORGE ALEJANDRO ALARCÓN MENA

QUITO DM, OCTUBRE DE 2025

Dedicatoria

Este trabajo de titulación está dedicado a:

A todas las personas que creyeron en mí, pero en especial a mi familia, por ser mi guía, mi apoyo y mi sostén en cada etapa de la universidad.

Por estar en cada momento importante, en los días buenos y en los difíciles, y por recordarme siempre que era capaz de llegar hasta aquí. Sin ustedes, nada de esto habría sido posible.

También a mi abuelita, que, aunque ya no está conmigo, siempre fue un pilar fundamental en mi vida. Su apoyo, cariño y enseñanzas me acompañaron durante todo este camino, y su recuerdo sigue siendo una motivación para alcanzar cada logro.

Me dedico también este logro a mí mismo, por no rendirme, por seguir adelante incluso cuando el camino se hacía pesado, por esforzarme, ser perseverante y disciplinado.

Hoy, llegar a este punto es la prueba de todo ese esfuerzo, y me lo reconozco con gratitud.

Agradecimiento

Agradezco, en primer lugar, a mi familia, por el apoyo incondicional brindado durante todo el proceso universitario. Por sus consejos, su paciencia, su confianza y por acompañarme en cada paso de esta etapa académica y personal.

Expreso también mi agradecimiento a mi tutor, por su guía, dedicación y orientación a lo largo de este proyecto, y a cada uno de mis profesores, quienes con sus enseñanzas y exigencia académica han contribuido a que hoy sea la persona y el profesional que estoy formando, con los conocimientos y valores que me acompañarán en mi futuro.

Finalmente, me agradezco a mí mismo por no desistir, por mantenerme firme frente a las dificultades y por dar siempre lo mejor de mí.

Abstract

The present degree project aims to design and implement a prototype web system oriented toward the management of critical inventory control processes for the NutriBoost sports supplements store. The project seeks to address recurring problems such as manual product and batch registration, lack of traceability in inventory movements, difficulty in controlling expiration dates, and the absence of operational reports to support decision-making.

To this end, a three-layer architecture is proposed, consisting of a relational database in SQL Server, a REST API developed in .NET 8 responsible for business logic, and a web interface implemented in Angular, which allows users to manage products, batches, inventory movements, expiration dates, and operational queries. The construction of the prototype is managed through a lightweight adaptation of the agile Scrum methodology, organized into two sprints: the first focused on the backend and data model, and the second on the graphical interface and its end-to-end integration.

The resulting prototype allows the registration and consultation of inventory information with the application of business rules such as batch control, automatic stock updates, and price change tracking, in addition to offering basic inventory, turnover, and expiration reports. Through unit tests, integration tests, and reviews with the stakeholder, it is validated that the solution meets the defined functional requirements and provides a solid foundation for future improvements and for eventual implementation in a production environment.

Tabla de Contenidos

Capítulo 1: Introducción 4

1.1 Justificación 4

1.2 Planteamiento del Problema 5

1.3 Objetivos 6

1.3.1 Objetivo General 6

1.3.2 Objetivos Específicos 6

1.4 Alcance 7

Capítulo 2: Marco Teórico 8

2.1 Aplicación Web 8

2.2 Metodología de Desarrollo de Software 8

2.3 Sistema de Gestión de Bases de Datos (SGBD) y Modelo de Datos 9

2.3.1 SGBD relacionales 9

2.3.2 Modelo relacional de la base de datos 10

2.4 Lenguajes de Programación en Aplicaciones Web 11

2.5 Entorno de Desarrollo Integrado (IDE) 12

2.6 Framework de Desarrollo Web (frontend y backend) 12

2.7 Control de Versiones 13

2.7.1 Importancia del control de versiones 13

2.7.2 Plataformas de repositorios 14

2.8 Buenas Prácticas de Desarrollo Web 14

2.8.1 Principios de diseño y usabilidad 14

2.8.2 Seguridad en Aplicaciones Web 15

Capítulo 3: Análisis y Diseño de la Aplicación 16

3.1 Metodología y Criterios de Selección Tecnológica 16

3.1.1 Metodología de desarrollo adoptada 16

3.1.2 Adaptación de Scrum al proyecto 19

3.1.3 Plan de aplicación de Scrum 19

3.1.4 Métricas e instrumentos de evaluación 24

3.1.5 Criterios de selección tecnológica 25

3.2 Análisis de Requerimientos 31

3.2.1 Técnicas de levantamiento de información 31

3.2.2 Identificación de actores 32

3.3 Especificación de Requerimientos 33
3.3.1 Casos de Uso detallados 33
3.3.2 Requerimientos Funcionales 46
3.3.3 Requerimientos No Funcionales 49
3.3.4 Aprobación de requerimientos por parte del cliente 49
3.4 Diseño de la Arquitectura del Sistema 50
3.4.1 Vista de despliegue (infraestructura) 50
3.5 Modelado del Sistema 51
3.5.1 Modelo de datos (ERD) 51
3.6 Diseño de la Interfaz de Usuario 52
Capítulo 4: Desarrollo de la Aplicación 54
4.1 Preparación del Entorno de Desarrollo 54
4.1.1 Estándares de codificación 54
4.1.2 Entorno de desarrollo 55
4.2 Implementación de la Funcionalidad 56
4.2.1 Desarrollo iterativo con Scrum 56
4.3 Plan de Pruebas 62
4.3.1 Tipos de Pruebas 62
4.3.2 Criterios de Aceptación 63
4.3.3 Evidencias de pruebas 63
CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES 64
5.1 Conclusiones 64
5.2 Recomendaciones 65
Bibliografía 66
Anexos 69
6.1 Evidencia del Sprint 1 69
6.2 Evidencia del Sprint 2 70
6.3 Acta del Sprint 1 73
6.4 Acta del Sprint 2 74
6.5 Pruebas unitarias 75
6.6 Pruebas de aceptación 77

Lista de Figuras

Figura 1 Caso de uso requerimiento F0 33
Figura 2 Caso de uso requerimiento F1 34
Figura 3 Caso de uso requerimiento F2 36
Figura 4 Caso de uso requerimiento F3 37
Figura 5 Caso de uso requerimiento F4 39
Figura 6 Caso de uso requerimiento F5 41
Figura 7 Caso de uso requerimiento F6 43
Figura 8 Caso de uso requerimiento F7 45
Figura 9 Vista de despliegue del prototipo 50
Figura 10 Modelo entidad–relación 51
Figura 11 Interfaz principal 52
Figura 12 Aplicación de estándares de codificación en el controlador Producto 55

Lista de Tablas

Tabla 1 Comparativa de características de metodologías de software. 17
Tabla 2 Plan de sprints. 20
Tabla 3 Backlog de producto priorizado 21
Tabla 4 Comparativa de arquitecturas orientadas a servicios. 26
Tabla 5 Comparativa entre los frameworks de desarrollo Angular y Vue.js. 28
Tabla 6 Comparativa entre sistemas de gestión de base de datos. 30
Tabla 7 Resumen de actividades y resultados del Sprint 1 58
Tabla 8 Resumen de actividades y resultados del Sprint 2 62

Capítulo 1: Introducción

1.1 Justificación

El negocio NutriBoost, enfocado en la comercialización de suplementos deportivos, evidenció limitaciones relevantes en el manejo y control de su inventario, debido a que los procedimientos se ejecutaban manualmente y el registro de productos se mantenía únicamente en hojas de cálculo de Excel, situación que introdujo restricciones operativas y brechas de consistencia para administrar el stock, especialmente cuando el volumen de referencias aumentaba y el seguimiento dependía de actualizaciones realizadas de forma dispersa por el personal.

Un problema recurrente se relacionó con la inexistencia de un control automatizado de fechas de caducidad, condición que expuso al negocio a pérdidas económicas por productos vencidos y a riesgos de calidad en la rotación de mercadería, en la misma línea, los movimientos de entrada y salida no se registraban con precisión, lo que debilitó la trazabilidad de existencias, incrementó errores en conteos manuales y redujo la confiabilidad de la información usada para decidir compras, reposiciones o promociones internas.

Ante ese escenario se propuso el desarrollo de un prototipo de sistema web orientado a la gestión de procesos críticos del control de inventarios, con base en una organización estructurada de productos, el registro formal de entradas y salidas y la administración de fechas de caducidad como eje de control, de manera que el prototipo funcionara como evidencia de validación para demostrar mejoras frente al esquema manual, aportando un punto de partida para migrar hacia una gestión automatizada con mayor eficiencia y menor dependencia de tareas repetitivas.

Con la incorporación del prototipo, NutriBoost pudo encaminarse hacia una operación más ordenada y profesional al disponer de información más consistente para apoyar decisiones y mantener disciplina en los procesos internos, el alcance se definió para uso exclusivo del personal del negocio sin interacción directa con clientes, concentrándose en el control administrativo del inventario y reemplazando el uso de hojas de cálculo y actividades manuales que consumían tiempo, con lo cual se fortaleció la continuidad del registro y la claridad sobre el estado real del stock.

1.2 Planteamiento del Problema

NutriBoost es un negocio local dedicado a la venta de suplementos deportivos. En su operación se evidenciaron dificultades recurrentes para gestionar el inventario con orden y consistencia, el control de productos se ejecutó de manera manual y con apoyo de hojas de cálculo en Excel, este enfoque resultó limitado frente al crecimiento del negocio y al aumento de referencias administradas de forma simultánea, en consecuencia el registro dependió de actualizaciones individuales y de revisiones posteriores, lo que redujo la confiabilidad de la información disponible para el control operativo y el análisis básico del stock.

Bajo ese método de trabajo se presentaron fallas asociadas al desorden en el registro de productos y a la baja precisión en las entradas y salidas de inventario, también se observaron dificultades para mantener un control sistemático de las fechas de caducidad, con ello se incrementó el riesgo de pérdidas económicas por vencimiento de mercancía y por discrepancias en las cantidades realmente disponibles, desde una perspectiva de sistemas de información el problema se reflejó en datos poco consistentes y en trazabilidad limitada, situación que afectó la toma de decisiones al no contar con registros verificables.

La ausencia de un prototipo de sistema web orientado a la gestión de procesos críticos del control de inventarios restringió la posibilidad de explorar alternativas preliminares para ordenar el stock y establecer reglas de operación más claras, aun cuando un prototipo no cubre por completo las necesidades de una solución final sí permite validar funcionalidades esenciales como el registro de productos el control de existencias y el seguimiento de fechas de caducidad, con ese respaldo se identificaron oportunidades de mejora se estimó la viabilidad de determinadas funciones y se definió una guía inicial para el desarrollo futuro de un sistema más completo, por tanto el objetivo del prototipo no fue reemplazar la solución definitiva sino aportar insumos prácticos que orientaran decisiones técnicas y operativas durante la transición.

Con base en la problemática descrita, se formuló la pregunta principal:

- ¿Cómo puede el desarrollo de un prototipo de sistema web para la gestión de procesos críticos del control de inventarios ayudar a reducir errores comunes, facilitar el control de fechas de caducidad y mejorar la organización del stock en la tienda de suplementos deportivos NutriBoost?

Y las siguientes preguntas secundarias:

- ¿Qué limitaciones presentó el método de control de inventarios en NutriBoost?
- ¿Qué funcionalidades mínimas debió tener un sistema web para cubrir las necesidades básicas de control de stock?
- ¿Cómo afectó la falta de registro de fechas de caducidad a la gestión del inventario?
- ¿Qué beneficios aportó un sistema automatizado frente al uso de Excel u otros métodos manuales?

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar un prototipo de sistema web para la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost, orientado a mejorar la precisión del registro de productos, el seguimiento de fechas de caducidad y la trazabilidad de los movimientos de stock.

1.3.2 Objetivos Específicos

- Realizar el análisis de requerimientos funcionales y no funcionales del prototipo, con el fin de identificar las necesidades prioritarias del control de inventarios en NutriBoost.
- Seleccionar la metodología de desarrollo de software más adecuada para el proyecto, considerando las características y necesidades del prototipo de sistema web para el control de inventarios.
- Diseñar la arquitectura y los modelos del sistema web, incluyendo el modelo entidad–relación como representación conceptual de los datos, y el diseño lógico y físico de la base de datos relacional.
- Implementar el prototipo del sistema web con las funcionalidades críticas del inventario, tales como registro de productos, control de stock y gestión de fechas de caducidad.
- Aplicar pruebas de validación del prototipo con datos simulados, evaluando el registro de stock, las fechas de caducidad y los reportes básicos.
- Elaborar conclusiones y recomendaciones basadas en la validación del prototipo, proponiendo mejoras y lineamientos para un futuro sistema completo de gestión de inventarios.

1.4 Alcance

El proyecto contempló únicamente el desarrollo de un prototipo de sistema web para la gestión de procesos críticos del control de inventarios, orientado al uso interno del negocio NutriBoost. Se enfocó exclusivamente en los procesos operativos esenciales relacionados con el inventario: registro de ingreso y salida de productos, seguimiento de fechas de caducidad, control de stock disponible y generación de reportes básicos.

Fue importante señalar que no se incluyeron funcionalidades relacionadas con la gestión de ventas ni con la interacción directa con clientes o plataformas externas. No obstante, el prototipo permitió mantener actualizado el inventario sin necesidad de integrar módulos de ventas, gracias a la implementación de registros manuales de movimiento de stock. Estos registros fueron gestionados directamente por el personal autorizado, quien ingresó en el sistema las salidas y entradas correspondientes cada vez que ocurrió una venta, una reposición o una pérdida, asegurando así la consistencia del inventario.

Este enfoque permitió reflejar en el sistema lo que efectivamente existía en bodega, sin requerir un sistema de ventas integrado. Asimismo, el sistema incluyó controles para evitar duplicidad o registros incompletos, mejorando la trazabilidad de cada movimiento.

Adicionalmente, este proyecto no constituyó un sistema completo en fase de producción, sino un prototipo funcional que permitió validar el impacto operativo, evaluar la viabilidad técnica y servir como punto de partida para futuras implementaciones más robustas.

Capítulo 2: Marco Teórico

2.1 Aplicación Web

Una aplicación web corresponde a un tipo de software que se lleva a cabo en un servidor remoto y en el que los usuarios acceden a través de un navegador web mediante Internet o una red local, sin necesidad de instalar programas en los equipos personales (Amazon Web Services, s.f.).

Su acceso mediante navegador facilita el uso desde distintos dispositivos, lo que resulta útil cuando el personal requiere consultar información sin instalaciones locales.

Se caracterizan por su arquitectura cliente-servidor. El frontend (lado del cliente) gestiona la interfaz visual y la interacción con el usuario, mientras que el backend (lado del servidor) procesa la lógica del negocio, las peticiones y el acceso a la base de datos. Esta división entre capas permite un desarrollo más estructurado, mantenible y adaptable a futuras mejoras.

Según (Amazon Web Services, s.f.), las aplicaciones web resultan especialmente útiles para organizaciones que necesitan manejar procesos en tiempo real, como transacciones, gestión de usuarios o control de inventarios. Además, al estar alojadas en la web, facilitan su disponibilidad desde cualquier dispositivo, garantizando eficiencia, accesibilidad y un acceso rápido para el personal autorizado.

En el contexto de NutriBoost, una aplicación web permite centralizar y consultar en tiempo real la información de inventario desde cualquier dispositivo con acceso a Internet, reduciendo registros duplicados y facilitando la trazabilidad de productos. Por ello, este proyecto propone una solución web orientada al control de existencias, el registro de movimientos y la generación de reportes para apoyar la toma de decisiones operativas.

2.2 Metodología de Desarrollo de Software

Las metodologías de desarrollo de software comprenden un conjunto estructurado de procesos, técnicas y reglas que ayudan en la planificación, creación, prueba e implementación de un sistema informático. De esta forma, el desarrollo no es improvisado y mantiene un orden que garantiza la calidad, la eficiencia y el cumplimiento de los objetivos planteados (Maida & Pacienzia, 2015).

Con el tiempo han surgido diversos tipos de metodologías, cada una con características propias. Dentro del proceso de desarrollo de software, las más utilizadas son las metodologías tradicionales y las metodologías ágiles.

Las metodologías tradicionales se caracterizan por seguir un proceso secuencial y rígido, en el que cada fase (análisis, diseño, desarrollo, pruebas e implementación) debe finalizarse completamente antes de iniciar la siguiente. Este tipo de enfoque cuenta con una documentación exhaustiva y ofrece un mayor control sobre las etapas del proyecto, aunque su principal limitación es la falta de flexibilidad ante los cambios (Zumba Gamboa & León Arreaga, 2018).

Por su parte, las metodologías ágiles surgieron como una respuesta a las limitaciones de los enfoques tradicionales, ya que se enfocan en la colaboración entre el equipo y el cliente, la entrega temprana de resultados y la capacidad de adaptación frente a los cambios. Según (González, s.f.), las metodologías ágiles buscan desarrollar software funcional, fomentar la interacción entre las personas y responder de manera efectiva y rápida ante los cambios, priorizando el valor real del desarrollo sobre la documentación excesiva.

De acuerdo con lo expuesto, las metodologías que se tomaron en consideración para el desarrollo de este proyecto fueron Scrum y Extreme Programming, ambas pertenecientes al enfoque ágil. Estas metodologías se analizaron con el fin de determinar cuál se adaptó mejor a las necesidades y particularidades del prototipo de sistema web para la gestión de procesos críticos del control de inventarios, considerando factores como el tamaño del proyecto, el nivel de interacción con el usuario y la dinámica del desarrollo.

2.3 Sistema de Gestión de Bases de Datos (SGBD) y Modelo de Datos

2.3.1 SGBD relacionales

Un sistema gestor de base de datos (SGBD) se compone de un conjunto de programas que permite definir esquemas y tablas, manipular datos a través del lenguaje del sistema y consultas de datos de forma eficiente. Tiene mecanismos de seguridad en roles, permisos y usuarios, control de concurrencia, es decir, acceso a varios usuarios simultáneamente manteniendo la integridad de los datos, control de transacciones, respaldos y recuperación ante fallos (Marqués, 2011).

Los SGBD relacionales son utilizados en diferentes entornos empresariales y en aplicaciones web porque permiten trabajar con grandes volúmenes de datos de manera

estructurada y confiable. Algunos SGBD relacionales populares en el mercado son SQL Server, PostgreSQL, MySQL y Oracle Database (Marqués, 2011).

En el caso del prototipo de sistema web para la gestión de procesos críticos del control de inventarios propuesto para NutriBoost, un SGBD relacional resultó ideal porque facilitó el registro de productos, movimientos y fechas de caducidad, asegurando que toda la información estuviera relacionada, íntegra y actualizada de forma coherente.

Dentro de las metodologías ágiles más utilizadas se consideran Scrum y Extreme Programming, Scrum se orienta a ordenar el trabajo por iteraciones con entregables visibles, manejo de backlog priorizado, roles definidos y espacios de revisión que permiten controlar avance por módulos sin perder el foco del producto, Extreme Programming pone el acento en la calidad técnica durante la construcción mediante prácticas como refactorización, simplicidad de diseño, estándares de codificación, integración frecuente y validación continua del comportamiento. Ambas encajan con un prototipo web de inventario como el de NutriBoost, porque el desarrollo se entiende por funcionalidades concretas como productos, lotes y movimientos, existen reglas y validaciones que se ajustan mientras se avanza, y se necesita mantener consistencia de datos sin frenar el progreso. En ese escenario las dos podrían resultar óptimas según el énfasis buscado, Scrum si se prioriza organización y seguimiento por incrementos, XP si se prioriza rigor técnico mientras se afinan cambios. Otras metodologías ágiles como Crystal o DSDM también son válidas, aunque suelen depender más del contexto organizacional, del tamaño del equipo o de reglas de gobernanza más marcadas, por lo que para este tipo de prototipo el análisis se entiende mejor si se concentra en Scrum y en XP.

2.3.2 Modelo relacional de la base de datos

Un modelo relacional representa los datos mediante tablas y relaciones lógicas. Cada tabla describe una entidad (por ejemplo, producto, cliente, proveedor, sucursal) y cada fila (tupla) corresponde a una instancia de esa entidad, mientras que las columnas (atributos) describen sus características (Silberschatz, Korth, & Sudarshan, 2014).

Las relaciones entre tablas se definen a partir del análisis del negocio y del modelo conceptual (diagrama entidad–relación). Para garantizar conexiones consistentes se utilizan claves primarias (identifican de forma única cada fila) y claves foráneas (enlazan una tabla

con otra), lo que mantiene la integridad referencial. En este modelo se cumple con tres propiedades:

- **Integridad:** los datos deben ser precisos, no adulterados.
- **Consistencia:** las relaciones entre tablas deben ser válidas.
- **No redundancia:** se evita duplicidad en los datos mediante normalización.

El modelo relacional constituye un pilar esencial en el diseño de sistemas de información, ya que permite estructurar los datos de forma lógica y organizada, garantizando la coherencia, la integridad y la eficiencia en el acceso a la información.

2.4 Lenguajes de Programación en Aplicaciones Web

Un lenguaje de programación es el medio por el cual los desarrolladores crean la lógica de una aplicación web, permitiendo que los datos se procesen correctamente. De esta forma, tenemos el lado del cliente, es decir, cómo se comporta una página en el navegador, y el lado del servidor, que se encarga de responder ante las peticiones del usuario (Pressman & Maxim, 2020).

En el frontend, los lenguajes más utilizados son HTML, CSS y JavaScript, que trabajan de forma conjunta para construir la parte visual e interactiva de la aplicación web.

- **HTML (HyperText Markup Language):** establece la estructura y define el contenido principal de una página web.
- **CSS (Cascading Style Sheets):** se encarga de determinar la apariencia visual del sitio, controlando la disposición de los elementos, los colores y el estilo general.
- **JavaScript:** aporta interactividad, validaciones y conexión dinámica con el servidor, permitiendo realizar operaciones sin necesidad de recargar la página constantemente.

Por otra parte, el backend utiliza lenguajes como Java, PHP, C#, Python y JavaScript (Node.js). El backend se encarga de procesar la información, aplicar las reglas del negocio, gestionar la base de datos y enviar los datos al cliente (Pressman & Maxim, 2020).

La combinación entre ambos entornos, cliente-servidor, permite construir aplicaciones web dinámicas, seguras y escalables, capaces de adaptarse a las necesidades del usuario y del sistema. Actualmente se requiere que los lenguajes de programación permitan una comunicación fluida entre el cliente, el servidor y la base de datos, favoreciendo que el

sistema se divida en partes más pequeñas e independientes, que facilitan el mantenimiento y la evolución del software (Sommerville, 2011).

2.5 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado (IDE) es una herramienta informática que ayuda en la programación de software de una manera más eficiente. El IDE por lo general incorpora un editor de código, un compilador o intérprete y un depurador; en la mayoría de las situaciones, esta herramienta permite escribir y probar código con mayor rapidez, reducir los fallos y mantener una estructura organizada del código.

Ayuda en la automatización de la edición, ya que el IDE reconoce las reglas de los lenguajes de programación y contiene funciones que ayudan a escribir o editar automáticamente el código fuente. En la parte de la sintaxis, analiza dónde está el error y lo resalta para que sea más visual; de esta forma da retroalimentación instantánea. También compila el código, es decir, lo convierte a un lenguaje que el sistema operativo pueda entender. Permite realizar pruebas de manera local antes de que el código se comparta o se ejecuten pruebas más complejas (Amazon Web Services, s.f.).

Los IDE modernos cuentan también con integraciones con servicios en la nube, bases de datos y frameworks, ayudando a optimizar tiempos y a mejorar la colaboración con el equipo.

Entre los IDE más usados para desarrollo web se encuentran Visual Studio Code, Eclipse, NetBeans y PyCharm; ofrecen soporte para varios tipos de lenguajes.

2.6 Framework de Desarrollo Web (frontend y backend)

En los sistemas que se desarrollan en web, se cuenta con dos partes que se complementan entre sí:

Frontend, aquello que los usuarios logran ver como elementos visuales, gráficos, botones y la interfaz gráfica; y por otro lado contamos con el backend, el lado del servidor donde se procesa la lógica y se responde a las peticiones. En esta capa está la infraestructura que permite que la aplicación funcione.

La interfaz gráfica de usuario cuenta con tres lenguajes principales:

- **HTML:** estructura el frontend y los diferentes elementos de la página.

- **CSS:** define el estilo de la aplicación web, es decir, colores, tipografía, diseño y presentación.
- **JavaScript:** permite la interactividad mediante eventos, animaciones y lógica en el cliente.

Backend corresponde al componente del servidor donde está la lógica del negocio, es decir, las reglas del negocio, sus procesos, la gestión del API para que el frontend logre hacer las consultas y enviar la información, la seguridad y la interacción con el motor de la base de datos. En esta capa hay varios tipos de lenguajes, entre los más comunes: Python, Java, C#, entre otros (Amazon Web Services, Inc., 2024).

Esta forma de desarrollo permite a la aplicación tener capacidad de concurrencia (es decir, que puedan estar varios dispositivos conectados a la vez), lo que permite que el backend gestione varias peticiones al mismo tiempo.

2.7 Control de Versiones

Es un sistema que permite gestionar los cambios cuando se realizan modificaciones en el código fuente de un programa de software. Su objetivo es ayudar en el registro de las modificaciones, indicando cuándo y por quién fueron realizadas. De esta forma, es posible regresar a versiones anteriores, comparar diferencias y trabajar de manera más organizada.

Según (Microsoft, 2023), entre las ventajas del control de versiones está la posibilidad de crear flujos de trabajo que controlan las versiones, evitando el caos que se puede generar al usar herramientas incompatibles. Además, proporciona permisos y facilita procesos que mantienen a todos los integrantes en sintonía. Los usuarios pueden trabajar en paralelo, y cada integrante puede desarrollar una función sin afectar el código principal.

También ayuda a mantener transparencia y trazabilidad en el proceso de desarrollo, lo que facilita la detección de errores, mejora la colaboración y garantiza la integridad del proyecto a lo largo del tiempo.

2.7.1 Importancia del control de versiones

- Permite mantener la historia completa de los cambios.
- Mejora la coordinación entre equipos distribuidos.
- Reduce riesgos de pérdida de información.
- Facilita la auditoría y revisión del código.

2.7.2 Plataformas de repositorios

- **GitHub:** repositorio basado en Git, ideal para proyectos colaborativos y de código abierto.
- **GitLab:** ofrece control de versiones integrado con herramientas DevOps.
- **Bitbucket:** orientado a equipos que usan Atlassian (como Jira o Trello).

2.8 Buenas Prácticas de Desarrollo Web

Las buenas prácticas de desarrollo web son un conjunto de técnicas y reglas que ayudan a que el desarrollo sea ordenado y de calidad. Según (García Sandoval, Ariza Torrado, Pinzón, & Flórez Fuentes, 2016), este enfoque permite identificar los requerimientos y la arquitectura del sistema, seguir los estándares de codificación, usar un control de versiones, realizar pruebas y mantener la documentación actualizada del desarrollo para próximas actualizaciones.

Estas prácticas ayudan a que el ciclo de vida del desarrollo sea más eficiente y predecible, reduciendo errores y facilitando el trabajo en equipo. Además, permiten que el código sea más claro, mantenible y fácil de mejorar con el tiempo.

También es importante aplicar principios de diseño y usabilidad, asegurando que la aplicación sea fácil de usar, intuitiva y agradable para el usuario. Junto con esto, deben considerarse medidas de seguridad web, que protejan los datos, las cuentas de usuario y la integridad del sistema durante todo el proceso de desarrollo.

2.8.1 Principios de diseño y usabilidad

La usabilidad es un aspecto esencial en un sitio web, ya que permite que sea fácil de aprender, efectivo y satisfactorio para el usuario al realizar las tareas del sistema. Según la (International Organization for Standardization, 2018), la usabilidad se define como la capacidad de un software para que usuarios específicos, en un contexto de uso determinado, logren objetivos en:

- **Eficacia:** los usuarios deben poder alcanzar sus objetivos con el software, con precisión y completitud.
- **Eficiencia:** los objetivos deben alcanzarse con el mínimo esfuerzo, tiempo o recursos posibles.
- **Satisfacción:** la experiencia de uso debe ser aceptable, cómoda y satisfactoria para el usuario.

2.8.2 Seguridad en Aplicaciones Web

La seguridad forma parte esencial de la calidad en el desarrollo de software, ya que ayuda a proteger tanto los procesos internos como los servidores donde se ejecuta la aplicación. Según (Herrera Infante, 2025), su principal objetivo es proteger el sitio web para garantizar que todo funcione de manera segura, estable y sin interrupciones.

Los pilares fundamentales de la seguridad son la integridad, la disponibilidad y la confidencialidad.

- **Integridad:** asegura que la información no sea modificada o alterada por usuarios no autorizados.
- **Disponibilidad:** garantiza que el sistema esté activo y accesible cuando los usuarios lo necesiten.
- **Confidencialidad:** protege los datos sensibles para que solo sean vistos por las personas autorizadas.

Existen varias prácticas que ayudan a mantener un sitio web seguro, entre las más importantes están:

- Implementar cifrado para proteger la información sensible, como contraseñas o datos de inventario, tanto durante su almacenamiento en la base de datos como en el proceso de transmisión entre el cliente y el servidor.
- Usar contraseñas robustas y almacenarlas mediante hash seguro dentro del sistema gestor de base de datos, evitando guardarlas en texto plano.
- Mantener el servidor web y las dependencias del sistema actualizadas, para corregir fallos de seguridad conocidos en librerías, frameworks o componentes.
- Realizar copias de seguridad (backups) periódicas de la base de datos y del sistema, con el fin de restaurar la información en caso de fallos, pérdida o ataques.

Capítulo 3: Análisis y Diseño de la Aplicación

3.1 Metodología y Criterios de Selección Tecnológica

3.1.1 Metodología de desarrollo adoptada

Scrum

Scrum se evaluó debido a que es uno de los marcos de trabajo ágiles principales aplicados en el desarrollo ágil. De acuerdo con (Schwaber & Sutherland, 2020), es un marco de trabajo ágil que permite que los equipos y organizaciones puedan mejorar el trabajo en problemas complejos. Esta metodología cuenta con tres pilares del empirismo: transparencia, inspección y adaptación. Dichos pilares permiten que el proceso de desarrollo sea estructurado, es decir, visible, verificable y mejorado de manera continua.

Los roles principales que tiene Scrum son el Product Owner, el Scrum Master y los Developers. Sin embargo, el presente proyecto fue un desarrollo que se realizó de manera individual, por lo cual dichos roles se asumieron individualmente, permitiendo una organización autónoma y un control directo de las tareas.

El trabajo en Scrum se organiza en iteraciones llamadas sprints, que son ciclos cortos de tiempo donde se planifican, desarrollan y entregan funcionalidades específicas. En cada sprint se incluye fases de planificación, ejecución, revisión y retrospectiva, con el objetivo de obtener incrementos funcionales del sistema que aporten valor real.

En el caso del prototipo de sistema web para la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost, el uso de Scrum se enfocó principalmente en los principios de transparencia y adaptación, ya que permitió una revisión constante del prototipo y una adaptación continua conforme a los resultados obtenidos. Scrum permitió garantizar las actividades del desarrollo de forma controlada, mantener una visión clara de los objetivos y facilitar la integración de los cambios que surgieron en el proceso.

Los principios esenciales de Scrum son:

- **Transparencia:** las tareas, avances y entregables se registraron y se mantuvieron visibles durante el desarrollo.
- **Inspección:** se revisaron periódicamente los resultados obtenidos en cada incremento para detectar mejoras o ajustes.

- **Adaptación:** se realizaron cambios puntuales cuando fue necesario, con base en la validación del avance y las necesidades funcionales del prototipo.

Extreme Programming (XP)

La metodología ágil Extreme Programming (XP) se evaluó, ya que está orientada a garantizar la calidad del software, y en prácticas rigurosas y ciclos de desarrollo cortos. Según (GoDaddy, 2024), XP se basa en cinco fundamentos esenciales: comunicación, simplicidad, retroalimentación, respeto y valentía, promoviendo entregas frecuentes y mejora continua del código.

Entre las prácticas más conocidas de XP se encuentra que el desarrollo se realice en parejas de programación, donde un integrante codifica y el otro revisa en tiempo real, lo cual mejora la detección temprana de errores y fortalece la calidad del producto final. Además, XP incorpora la integración continua, la refactorización del código y pruebas automatizadas, para que cada cambio o nueva funcionalidad mantenga la estabilidad del sistema.

Otra característica importante es su enfoque en la retroalimentación constante del cliente o usuario final, lo que permite ajustar rápidamente el software a los requerimientos cambiantes y reducir el riesgo de desviaciones técnicas o funcionales. Esta metodología resulta especialmente óptima para equipos de desarrollo colaborativos, donde los programadores revisan y comparten código de manera continua, generando un entorno de trabajo dinámico, cooperativo y enfocado en la mejora técnica permanente.

Tabla 1

Comparativa de características de metodologías de software.

Características	Scrum	Extreme Programming (XP)
Enfoque	Metodología ágil orientada a la gestión de proyectos iterativos e incrementales.	Metodología ágil orientada a la calidad técnica y la mejora continua del código.
Aplicación	Proyectos con entregas funcionales planificadas mediante sprints.	Proyectos con entregas continuas y prácticas de desarrollo intensivas.
Ciclos de trabajo	Iterativos e incrementales (Sprints de corta duración).	Iterativos con entregas constantes y pruebas continuas.

Roles principales	Product Owner, Scrum Master y Equipo de Desarrollo.	Cliente, Programador, Coach, Tracker y Tester.
Artefactos principales	Product Backlog, Sprint Backlog, Incremento.	Historias de usuario, pruebas automatizadas, código refactorizado.
Reuniones clave	Sprint Planning, Daily Scrum, Sprint Review y Retrospective.	Reuniones diarias y sesiones de revisión técnica continua.
Técnicas de modelado	Diagramas UML e historias de usuario.	Historias de usuario y casos de prueba automatizados.
Representación gráfica	Tableros de tareas y burndown charts.	Diagramas de flujo de código e integración continua.
Personalización	Estructurada; requiere planificación de tiempos y entregas definidas.	Flexible; prioriza la simplicidad y adaptación técnica.
Etapas	• Inicio • Planificación • Implementación • Revisión • Retrospectiva	• Exploración • Planificación • Iteración • Producción • Mantenimiento

Nota. Comparativa entre las metodologías ágiles Scrum y Extreme Programming (XP), resaltando sus diferencias en estructura, roles, prácticas y enfoque dentro del desarrollo de software. Elaboración propia, 2026.

De acuerdo con el análisis comparativo, para el desarrollo del prototipo de sistema web para la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost se optó por emplear la metodología Scrum, ya que, a diferencia de Extreme Programming (XP), se adaptó mejor al tipo de desarrollo planteado al ofrecer una estructura más clara basada en sprints, planificación iterativa y revisión constante de avances. Mientras que XP está más orientada a la calidad del código y al trabajo colaborativo mediante prácticas como la programación en pareja, la integración continua y las pruebas automatizadas, Scrum resultó más adecuado para un desarrollo individual, permitiendo mantener un control ordenado del progreso, priorizar tareas y realizar entregas incrementales del sistema. Además, los principios de transparencia, inspección y adaptación

facilitaron la revisión continua del prototipo, la incorporación de mejoras en cada iteración y la alineación del producto con los objetivos propuestos, garantizando así un desarrollo ágil, flexible y de calidad.

3.1.2 Adaptación de Scrum al proyecto

Debido a que el desarrollo del sistema se ejecutó de manera unipersonal, el responsable del prototipo asumió simultáneamente los roles de Scrum Master y Developer, mientras que un vendedor de NutriBoost intervino como Product Owner funcional al representar las necesidades del negocio y validar los requerimientos planteados, bajo esa dinámica se sostuvo un intercambio constante con el usuario final y se cuidó que cada incremento del sistema respondiera a actividades reales de la tienda, en ese marco el uso de Scrum conservó sus principios esenciales mediante transparencia apoyada en tableros de avance visibles, inspección a través de revisiones periódicas con el representante del negocio, y adaptación mediante ajustes iterativos derivados del feedback recibido, con lo cual el desarrollo se mantuvo ágil, verificable y centrado en el uso cotidiano del inventario.

3.1.3 Plan de aplicación de Scrum

Para el desarrollo del prototipo de sistema web orientado a la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost se adoptó Scrum con un enfoque iterativo e incremental, en este contexto el trabajo se organizó en dos Sprints de dos semanas cada uno para planificar entregas parciales y aplicar correcciones derivadas de la retroalimentación obtenida al cierre de cada ciclo, a la par se conformó un Product Backlog con una lista priorizada de funcionalidades del sistema, entre las que se incluyeron la gestión de productos, el registro de lotes, los movimientos de inventario y la generación de reportes operativos, con ese sustento se definieron objetivos por Sprint privilegiando los módulos que aportaban mayor valor al control y a la trazabilidad del inventario.

Cada iteración se planificó mediante una reunión inicial en la que se delimitaron tareas y entregables esperados, de forma paralela se mantuvo un seguimiento continuo mediante revisiones periódicas con el cliente para confirmar el avance, atender incidencias y ajustar requerimientos cuando la implementación evidenció cambios necesarios, al término de cada Sprint se revisó el incremento construido y se presentaron las funcionalidades terminadas, quedando registradas las observaciones levantadas para incorporarlas en la siguiente iteración como parte del refinamiento progresivo del prototipo.

Tabla 2

Plan de sprints.

Sprint	Duración	Objetivo principal	Entregables	Validación
Sprint 1	2 semanas	Implementar el módulo de productos e inventario base	CRUD de productos, base de datos estructurada, lógica de negocio	Aprobado por el cliente tras revisión funcional
Sprint 2	2 semanas	Desarrollar el módulo de lotes y movimientos FIFO	Registro de entradas/salidas, control de stock, reportes operativos	Validado por el Product Owner funcional y listo para integración final

Nota: Detalle de los sprints realizados durante el desarrollo del proyecto, incluyendo su duración, objetivos, entregables y validación correspondiente. Elaboración propia, 2026.

Este plan permitió mantener una secuencia de desarrollo ordenada, con entregas incrementales y la posibilidad de incorporar correcciones en función de la retroalimentación obtenida en cada revisión. De esta manera, se logró mantener un ritmo constante de trabajo y una mejora progresiva en la funcionalidad y estabilidad del sistema.

3.1.3.1 Product Backlog Prioritario

La columna “Requisitos (F0–F7)” indica el código de cada requisito funcional para su referencia y trazabilidad en el Product Backlog.

Tabla 3

Backlog de producto priorizado

ID	Historia de usuario	Prioridad	Puntos historia	Requisitos (F0–F7)	Criterios de aceptación
HU1	Como administrador, quiero iniciar sesión con credenciales válidas para acceder al sistema.	Alta	3	F0	Acceso exitoso con datos correctos; mensaje de error con datos inválidos.
HU2	Como administrador, quiero registrar un nuevo producto (nombre, categoría, precio, etc.) para incluirlo en el inventario.	Alta	5	F1	El producto se almacena en la base de datos y aparece en la lista de productos.
HU3	Como administrador, quiero editar los datos de un producto existente para mantener la información actualizada.	Media	3	F1	Las modificaciones se guardan y se reflejan en la lista de productos.

HU4	Como usuario autenticado, quiero ver los productos con su stock disponible.	Alta	5	F3	Se muestra la lista de productos con existencias actualizadas según la BD.
HU5	Como usuario autenticado, quiero registrar entradas y salidas de mercancía para actualizar el stock de cada producto.	Alta	8	F3	El stock se incrementa/disminuye correctamente tras el movimiento registrado.
HU6	Como administrador, quiero generar reportes de inventario filtrados por producto o periodo para analizar existencias históricas.	Media	5	F5	El reporte muestra la información correcta según el filtro; puede exportarse o visualizarse.
HU7	Como administrador, quiero registrar y gestionar lotes (cantidad, fecha de caducidad, estado) para controlar el inventario por lote.	Alta	5	F2	El lote se registra correctamente; muestra cantidad, caducidad y estado actualizado.

HU8	Como administrador, quiero visualizar productos próximos a vencer para tomar decisiones de reposición o descarte.	Media	3	F4	Los productos se muestran según el umbral configurado; se generan alertas si aplica.
HU9	Como administrador, quiero consultar reportes operativos (rotación, valoración, inventario y caducidades) para análisis general.	Media	5	F6	La información coincide con los datos de la BD; filtros funcionan correctamente.
HU10	Como administrador, quiero registrar, modificar y deshabilitar usuarios para controlar los accesos al sistema.	Alta	3	F7	El usuario se crea/edita; roles aplican correctamente; no se permiten duplicados.

Nota. Product Backlog del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost, elaborado con base en los requerimientos funcionales F0–F7. Elaboración propia, 2026.

3.1.4 Métricas e instrumentos de evaluación

En esta sección se precisaron las métricas e instrumentos de evaluación utilizados para valorar el avance y la calidad del desarrollo del prototipo de sistema web enfocado en la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost durante la aplicación del marco ágil Scrum, con ello se procuró trabajar con indicadores claros y fáciles de interpretar que reflejaran el cumplimiento de los objetivos definidos en cada iteración, además se establecieron criterios que permitieran observar si el proceso aplicado estaba dando resultados consistentes y que, al mismo tiempo, brindaran sustento para decidir ajustes puntuales cuando el trabajo entregado mostraba brechas frente a lo esperado conforme los incrementos se iban consolidando.

Las métricas seleccionadas fueron las siguientes:

- **Cumplimiento del objetivo del Sprint:** se midió el grado de avance alcanzado en cada iteración, considerando el porcentaje de tareas completadas respecto a las planificadas en el Sprint Backlog.
- **Cumplimiento de la Definition of Done (DoD):** se verificó si cada funcionalidad desarrollada cumplía con los criterios definidos como "terminada", tales como funcionamiento correcto, validación con datos reales y aprobación del cliente.
- **Retrabajo o incidencias:** se registró el número de ajustes o correcciones requeridos tras las revisiones del cliente, con el fin de evaluar la claridad de los requerimientos y la calidad del desarrollo.
- **Satisfacción del cliente:** al cierre de cada revisión, el cliente calificó el cumplimiento de los objetivos y la utilidad del incremento mediante una escala de 1 a 5, sirviendo como indicador de validación del producto.

Como instrumento de apoyo se emplearon:

- Actas de revisión de cada Sprint, en las que se documentaron las observaciones del cliente y las decisiones para la siguiente iteración.

Estas métricas e instrumentos permitieron mantener un control sistemático del proyecto, garantizando la trazabilidad del proceso de desarrollo y la mejora continua entre los Sprints.

3.1.5 Criterios de selección tecnológica

3.1.5.1 Arquitectura del sistema

Para sustentar el desarrollo del prototipo de sistema web orientado a la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost se examinó el uso de dos arquitecturas de servicios vigentes en escenarios empresariales actuales, SOAP y REST, ambas habilitan el intercambio de información entre cliente y servidor mediante servicios web, aunque difieren en la estructuración de mensajes, en la carga técnica de implementación y en la facilidad de mantenimiento, bajo ese criterio el análisis se orientó a reconocer cuál alternativa se ajusta mejor a un sistema que exige respuestas ágiles y consumo responsable de recursos sin descuidar consideraciones de seguridad, rendimiento y escalabilidad.

SOAP (Simple Object Access Protocol)

SOAP se caracteriza por ser un protocolo estándar que establece comunicación entre aplicaciones distribuidas mediante mensajes formales basados en XML, esta formalidad aporta robustez en la definición y validación de datos, favorece la integridad durante el intercambio y suele integrarse con mecanismos consolidados de seguridad, en particular (Amazon Web Services , s. f.), señala que el enfoque puede apoyarse en WS-Security como conjunto de extensiones para incorporar cifrado y autenticación en los mensajes, no obstante la obligatoriedad de XML incrementa la complejidad de procesamiento y eleva el tamaño de los mensajes, situación que demanda mayor ancho de banda y puede penalizar el rendimiento cuando se requieren interacciones ligeras o de baja latencia.

REST (Representational State Transfer)

REST se entiende como un estilo arquitectónico que aprovecha métodos estándar de HTTP como GET, POST, PUT y DELETE para transferir datos, su adopción resulta frecuente en aplicaciones web por su enfoque ligero y flexible, en concordancia con (Amazon Web Services , s. f.), el intercambio suele realizarse mediante JSON, lo que simplifica el procesamiento y optimiza el uso de recursos al reducir la verbosidad de los mensajes, adicionalmente el diseño stateless permite que cada solicitud se atienda de forma independiente sin conservar estado de sesiones previas, característica que favorece la escalabilidad, facilita el mantenimiento y contribuye a tiempos de respuesta más consistentes

en entornos donde el sistema debe crecer sin aumentar de forma desproporcionada la complejidad operativa.

Desde la perspectiva de un estudiante de Sistemas de la Información el contraste entre ambas alternativas se aprecia con claridad al relacionar necesidades de integración y control con la realidad operativa de un prototipo web, SOAP aporta un marco más rígido y orientado a contratos formales con capas de seguridad estandarizadas, mientras REST suele alinearse con implementaciones más ágiles y con menor sobrecarga en el tráfico de datos, por lo tanto la decisión técnica se fortalece cuando se pondera el equilibrio entre seguridad requerida, facilidad de evolución, capacidad de escalar y eficiencia en el consumo de recursos dentro del contexto específico de NutriBoost.

Tabla 4

Comparativa de arquitecturas orientadas a servicios.

Arquitectura	SOAP	REST
Tipo	Protocolo estructurado basado en XML.	Estilo arquitectónico que usa HTTP.
Formato de datos	Solo XML.	JSON, XML u otros formatos.
Complejidad	Alta; requiere configuración detallada.	Baja; fácil de implementar y mantener.
Velocidad	Menor, por el peso de XML.	Mayor, por su estructura ligera.
Seguridad	WS-Security con cifrado y autenticación.	HTTPS y tokens; más simple pero efectiva.
Escalabilidad	Limitada, dependiente del servidor.	Alta, ideal para aplicaciones web y móviles.
Adecuación al proyecto	Requiere infraestructura avanzada.	Ideal para desarrollos ligeros.

Nota. La tabla muestra una comparación entre las arquitecturas SOAP y REST, destacando sus diferencias principales en estructura, velocidad y facilidad de integración dentro de aplicaciones web modernas. Elaboración propia, 2026.

Según el análisis anterior, para la arquitectura del prototipo web que apoya el control de inventarios en la tienda de suplementos NutriBoost se escogió REST, porque es más ligero, envía datos de forma eficiente y presenta menor complejidad de implementación, esto

permitió una comunicación clara entre el cliente y el servidor usando recursos y métodos HTTP comunes, además el uso de JSON hizo los mensajes más pequeños y fáciles de manejar, redujo el consumo de ancho de banda y ayudó a integrar los servicios sin fricción, logrando un flujo de información rápido, escalable y acorde a lo que necesita el prototipo.

3.1.5.2 Framework de desarrollo

Para la construcción del prototipo sistema web que apoya el control de inventarios en la tienda de suplementos deportivos NutriBoost se revisaron dos frameworks frontend muy usados hoy: Angular y Vue.js, los dos ayudan a crear pantallas dinámicas, que pueden crecer y mantenerse por módulos, pero se diferencian en cómo están armados, en lo que cuesta aprenderlos y en el respaldo empresarial.

Angular

Angular es un framework para desarrollo web que trabaja con TypeScript, una extensión de JavaScript que añade tipos y orden, lo que ayuda a escribir código más claro y fácil de mantener, su enfoque principal son las aplicaciones SPA de una sola página, donde el contenido se actualiza de forma dinámica sin recargar todo el sitio, de acuerdo con (Angular Team, 2025), utiliza una arquitectura por componentes inspirada en el patrón Model View ViewModel (MVVM) e integra inyección de dependencias, enrutamiento y data binding bidireccional para fortalecer la modularidad y el mantenimiento del proyecto.

Sus principales características son:

- Data Binding bidireccional, que sincroniza automáticamente los datos entre el modelo y la vista.
- Inyección de dependencias, que mejora la reutilización y el control del código.
- Enrutamiento avanzado y un sistema de componentes reutilizables que optimizan la navegación y el rendimiento.
- Soporte a largo plazo (LTS) y actualizaciones periódicas mantenidas directamente por Google.

Vue.js

Vue.js es un framework progresivo que permite construir un proyecto de manera gradual según la necesidad del sistema y del equipo. De acuerdo con (Vue.js Core Team, 2025), Vue.js está desarrollado en JavaScript y combina librerías ligeras con funciones

avanzadas de reactividad, entendida como la capacidad de la aplicación para actualizar la interfaz cuando cambian los datos sin recargar la página. Además, incorpora un manejo eficiente del Document Object Model (DOM), que corresponde a la estructura jerárquica del contenido HTML, lo que facilita modificar y renderizar elementos de forma dinámica y optimizada.

Sus principales características son:

- Ligereza y rapidez en la carga, lo que mejora el rendimiento en aplicaciones pequeñas y medianas.
- Curva de aprendizaje sencilla, ideal para desarrolladores que inician en el desarrollo web moderno.
- Sistema reactivo, que actualiza automáticamente los elementos visuales cuando cambian los datos.
- Integración simple, ya que puede combinarse con otras bibliotecas o insertarse en proyectos existentes sin reescribir todo el código.

Tabla 5

Comparativa entre los frameworks de desarrollo Angular y Vue.js.

Framework	Angular	Vue.js
Lenguaje base	TypeScript	JavaScript
Arquitectura	MVVM y modular, basada en componentes.	Progresiva, adaptable y ligera.
Curva de aprendizaje	Media; requiere conocimientos previos de TypeScript.	Baja; sencilla y rápida de dominar.
Rendimiento	Alto en aplicaciones grandes y estructuradas.	Muy alto en proyectos pequeños o medianos.
Soporte	Respaldado por Google, con actualizaciones regulares.	Comunidad independiente, menor respaldo institucional.
Escalabilidad	Alta; ideal para aplicaciones grandes y mantenibles.	Media; adecuada para proyectos pequeños.
Adecuación al proyecto	Requiere mayor estructuración inicial, pero garantiza estabilidad.	Ideal para desarrollos rápidos y de bajo mantenimiento.

Nota. La tabla muestra una comparación entre los frameworks Angular y Vue.js, resaltando sus principales diferencias en estructura, rendimiento y soporte. Elaboración propia, 2026.

De acuerdo con lo analizado, para el desarrollo del prototipo de sistema web para la gestión de procesos críticos del control de inventarios en la tienda de suplementos deportivos NutriBoost, se seleccionó Angular, ya que proporcionó una arquitectura sólida, una estructura organizada basada en componentes y un soporte empresarial continuo que garantizó su estabilidad a largo plazo. Aunque Vue.js ofreció mayor facilidad de aprendizaje y rapidez inicial, Angular resultó más adecuado para mantener la consistencia y escalabilidad del proyecto.

3.1.5.3 Sistema gestor de base de datos

Para la gestión de la información del prototipo web de control de inventarios de NutriBoost, se evaluaron los sistemas de bases de datos relacionales SQL Server y PostgreSQL, ambos fueron seleccionados por su estabilidad, compatibilidad con diversos lenguajes y soporte para transacciones seguras.

SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacional (RDBMS) robusto desarrollado por Microsoft que permite a las aplicaciones y herramientas conectarse con una instancia de base de datos utilizando Transact-SQL (T-SQL). Según (Microsoft, 2025), su motor de base de datos administra el almacenamiento, procesamiento y seguridad de los datos, ofreciendo soporte completo a las transacciones con propiedades ACID (atomicidad, consistencia, aislamiento y durabilidad), lo que lo convierte en una opción confiable para entornos que requieren integridad, disponibilidad y control riguroso de la información.

PostgreSQL

Este sistema de gestión de bases de datos relacional es avanzado, orientado a objetos y de código abierto. Según (Microsoft, 2025), PostgreSQL ofrece una arquitectura robusta, un mayor nivel de integridad en los datos y soporte completo para transacciones; permite el manejo de tipos de datos personalizados, consultas complejas, procedimientos almacenados y funciones. Su capacidad de escalabilidad lo convierte en una opción óptima para proyectos que pueden crecer o que requieren un análisis más detallado de los datos a futuro.

Tabla 6

Comparativa entre sistemas de gestión de base de datos.

Criterio	Microsoft SQL Server	PostgreSQL
Tipo	Relacional (T-SQL).	Relacional y orientado a objetos (extensible).
Soporte transaccional	Completo (ACID); aislamientos configurables, incluye SNAPSHOT/READ COMMITTED SNAPSHOT.	Completo (ACID); MVCC por defecto.
Consultas y rendimiento	Muy sólido en OLTP; optimizador maduro, Query Store, índices columnstore para analítica ligera.	Excelente en consultas complejas; CTE, funciones de ventana, JSONB, extensiones potentes.
Integridad y objetos	Claves foráneas, CHECK, triggers, funciones, vistas, procedimientos almacenados (T-SQL).	Claves foráneas, CHECK, triggers, funciones (PL/pgSQL), vistas, procedimientos (desde v11).
Alta disponibilidad / escalabilidad	Always On Availability Groups, replicación, particionamiento; opciones de clustering.	Replicación en streaming y lógica; particionamiento; clustering con herramientas externas.
Seguridad	TDE, Always Encrypted, Row-Level Security, enmascarado de datos; integración con AD.	Row-Level Security, cifrado en tránsito; cifrado a nivel de datos vía extensiones (p. ej., pgcrypto).
.NET / EF Core	Integración nativa (SqlClient); tooling y scaffolding sólidos.	Proveedor Npgsql disponible para EF Core.
Herramientas de administración	SQL Server Management Studio (SSMS), Azure Data Studio.	pgAdmin, psql, ecosistema de extensiones.
Adecuación al prototipo	Óptimo: integración nativa con .NET/EF Core, fuerte en	Muy bueno: potente y extensible; integra con .NET

	transacciones y triggers para inventario; administración madura.	vía Npgsql, con mayor ajuste fuera del stack Microsoft.

Nota. La tabla muestra una comparación entre los motores de bases de datos SQLServer y PostgreSQL. Elaboración propia, 2026.

De acuerdo con lo analizado, para el sistema gestor de base de datos del proyecto se decidió utilizar Microsoft SQL Server, debido a su arquitectura robusta y cumplimiento estricto de los principios ACID, lo que garantiza la integridad de los datos, este motor permite ejecutar consultas complejas y mantener la consistencia de la información aún bajo operaciones simultáneas, además de ofrecer excelentes capacidades de administración (SSMS) e integración nativa con .NET/Entity Framework Core. Por estas características, se consideró una opción confiable para el desarrollo de NutriBoost, al ofrecer estabilidad y precisión en el manejo de los registros del sistema.

3.2 Análisis de Requerimientos

3.2.1 Técnicas de levantamiento de información

Para el levantamiento de información se aplicó una entrevista semiestructurada dirigida a un colaborador que trabajó directamente con NutriBoost y que participaba en el proceso de venta y control de stock de los suplementos deportivos, la entrevista se realizó mediante videollamada por Google Meet y permitió conocer cómo se llevaba el inventario en la operación diaria, a partir de esta conversación se identificaron dificultades asociadas al manejo manual de los registros y al seguimiento de fechas de caducidad, con lo cual se obtuvo un panorama claro sobre los puntos críticos que debían corregirse mediante el prototipo.

De forma complementaria se revisaron documentos internos utilizados por la tienda, principalmente hojas de cálculo en Excel donde se registraban ingresos, salidas, existencias y fechas de caducidad, esta revisión permitió observar de manera directa el funcionamiento del control manual y sus principales limitaciones, en especial la falta de alertas o mecanismos que faciliten detectar a tiempo productos próximos a vencer, con esta evidencia se reforzó la necesidad de automatizar el registro y mejorar la consistencia de la información.

Con los resultados obtenidos a partir de estas técnicas se logró comprender con mayor precisión el flujo real del negocio y se definieron los requerimientos que debía cubrir el prototipo, el enfoque se orientó a mejorar la organización del inventario, asegurar

trazabilidad en los movimientos y fortalecer el control de fechas de caducidad, de esta manera se estableció una base sólida para el análisis de funcionalidades y para la validación posterior del sistema junto con el usuario del negocio.

3.2.2 Identificación de actores

Durante el análisis que se realizó en la tienda NutriBoost, se identificó que los actores principales fueron quienes interactuaban en la gestión de los productos y cumplían un rol específico dentro de las operaciones del prototipo, permitiendo diferenciar los niveles de acceso y, con su segregación, las funciones que podían realizar.

Actor 1: Administrador

Perfil: persona dueña del negocio o encargada principal del prototipo, tiene acceso total a todas las funcionalidades, como registrar, seleccionar, modificar productos, realizar consultas del inventario, generar reportes, controlar los lotes de los productos para las fechas de caducidad y gestionar los usuarios del prototipo. Además, supervisó los movimientos del inventario y recibió las notificaciones generadas automáticamente por el sistema.

Actor 2: Empleado

Perfil: responsable que apoyó las operaciones diarias de la tienda. Su función fue principalmente registrar entradas y salidas de los productos en el prototipo, mantener actualizada la información del stock y verificar los productos próximos a caducar. Su acceso fue limitado, ya que únicamente realizó acciones operativas, sin permisos para generar reportes o gestionar permisos en el prototipo.

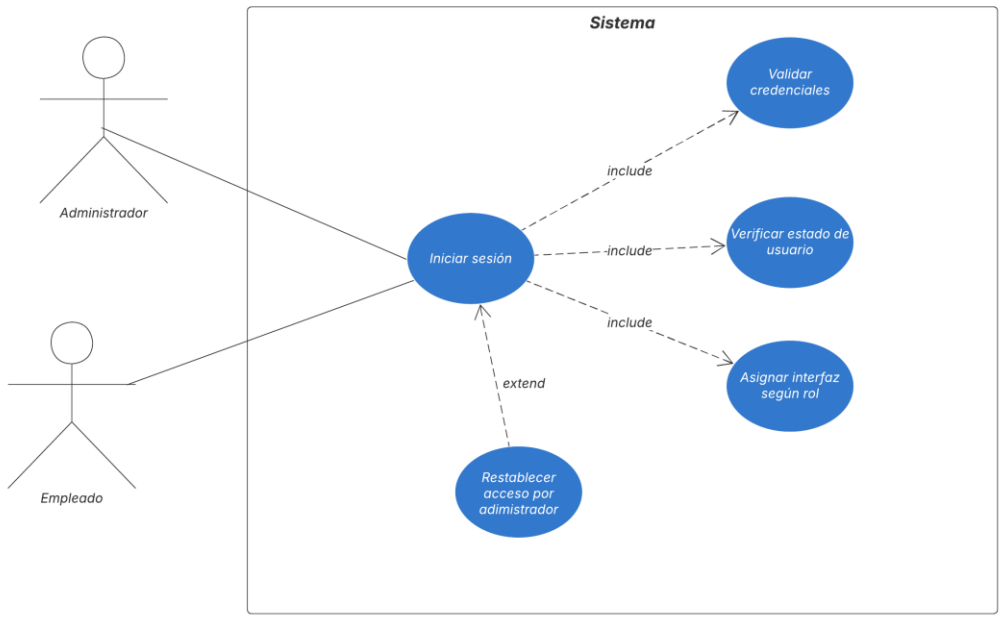
3.3 Especificación de Requerimientos

3.3.1 Casos de Uso detallados

3.3.1.1 Requerimiento F0

Figura 1

Caso de uso requerimiento F0



Nota. Diagrama de casos de uso del proceso de inicio de sesión en el prototipo, mostrando las interacciones entre los actores y los casos de uso asociados. Elaboración propia, 2026.

Actores: Administrador **y** Empleado

Descripción: Permite que un usuario acceda al prototipo mediante su nombre de usuario y contraseña. El prototipo valida las credenciales, verifica que la cuenta esté activa y asigna la interfaz correspondiente según el rol del usuario (Administrador o Empleado).

Precondición:

- El usuario debe estar previamente registrado en la base de datos.
- La cuenta del usuario debe estar activa y no deshabilitada.

Flujo principal:

1. Usuario ingresa usuario/contraseña.

2. El prototipo valida las credenciales utilizando contraseñas almacenadas con hash seguro y determina el rol del usuario.
3. Prototipo concede acceso con perfil (ADMIN/EMPLEADO).

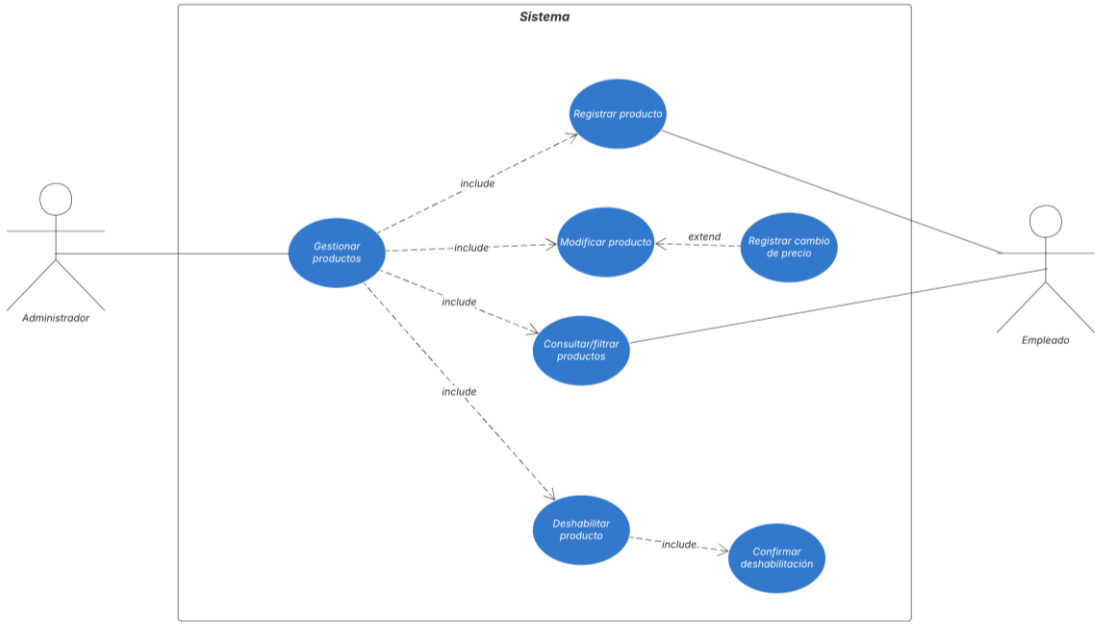
Flujos alternos:

4. Si las credenciales son inválidas, el prototipo muestra un mensaje de error y permite reintentar.
5. Si la cuenta está inactiva o bloqueada, el acceso se deniega.

3.3.1.2 Requerimiento F1

Figura 2

Caso de uso requerimiento F1



Nota. Diagrama de casos de uso del proceso de gestión de productos en el prototipo, mostrando las acciones disponibles para los actores y sus relaciones. Elaboración propia, 2026.

Actores: Administrador, Empleado

Descripción: El prototipo permite registrar, modificar, consultar y deshabilitar productos. El módulo controla el catálogo de productos, sus precios de compra y venta, tipos, marcas y estados. El Administrador tiene acceso total (registro, modificación y deshabilitación), mientras que el Empleado solo puede registrar y consultar productos.

Precondición: El usuario ha iniciado sesión en el prototipo y tiene el rol correspondiente (Administrador o Empleado).

Flujo principal:

1. El usuario accede a la opción “Gestionar productos”.
2. El prototipo muestra la lista de productos registrados.
3. El usuario selecciona la acción a realizar:
    - Registrar producto: ingresa nombre, sabor, marca, tipo, precio de compra, precio de venta y estado.
    - Modificar producto: actualiza datos generales o precios.
    - Consultar/filtrar producto: busca productos por nombre, tipo, marca o código.
    - Deshabilitar producto: cambia su estado para evitar su uso en futuras operaciones.
4. Al modificar precios, el prototipo registra automáticamente el cambio en el historial de precios con los valores anteriores y nuevos.
5. Al deshabilitar un producto, el prototipo solicita confirmación antes de proceder.
6. El prototipo valida los datos y guarda los cambios realizados.
7. El prototipo muestra un mensaje confirmando la acción ejecutada.

Flujos alternos:

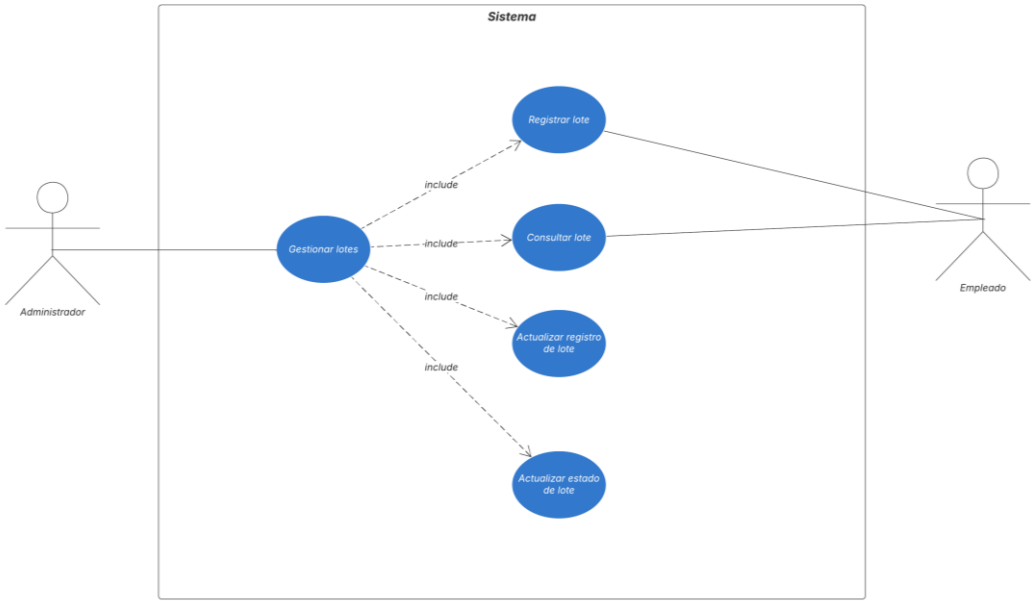
8. Si el producto ya existe (mismo código), el prototipo muestra el mensaje “El producto ya está registrado.”
9. Si el usuario cancela la confirmación de deshabilitación, el prototipo mantiene el producto activo y no realiza cambios.

Postcondición: El prototipo mantiene actualizado el catálogo de productos, con precios históricos y estados correctos.

3.3.1.3 Requerimiento F2

Figura 3

Caso de uso requerimiento F2



Nota. Diagrama de casos de uso del proceso de gestión de lotes en el prototipo, mostrando las acciones y relaciones entre los actores involucrados. Elaboración propia, 2026.

Actores: Administrador, Empleado

Descripción: Permite registrar, consultar y actualizar los lotes asociados a cada producto. El módulo controla la trazabilidad de las fechas de caducidad, cantidades y estados de los lotes. El Administrador tiene acceso total (registro, modificación y actualización de estado), mientras que el Empleado solo puede registrar y consultar lotes.

Precondición:

- El usuario ha iniciado sesión (ADMIN/EMPLEADO).
- Existe al menos un producto previamente registrado.

Flujo principal:

1. El usuario accede a la opción “Gestionar lotes”.
2. El prototipo muestra la lista de lotes existentes por producto.
3. El usuario selecciona la acción que desea realizar:
    - Registrar lote: ingresa número de lote, fecha de caducidad y cantidad inicial.

- Consultar lote: busca lotes por producto, número o estado.
- Actualizar registro de lote: corrige información registrada (solo ADMIN).
- Actualizar estado de lote: cambia el estado (Disponible, Agotado, Vencido o Bloqueado).

4. El prototipo valida los datos ingresados y actualiza la información.
5. El prototipo muestra un mensaje de confirmación de la operación realizada.

Flujos alternos:

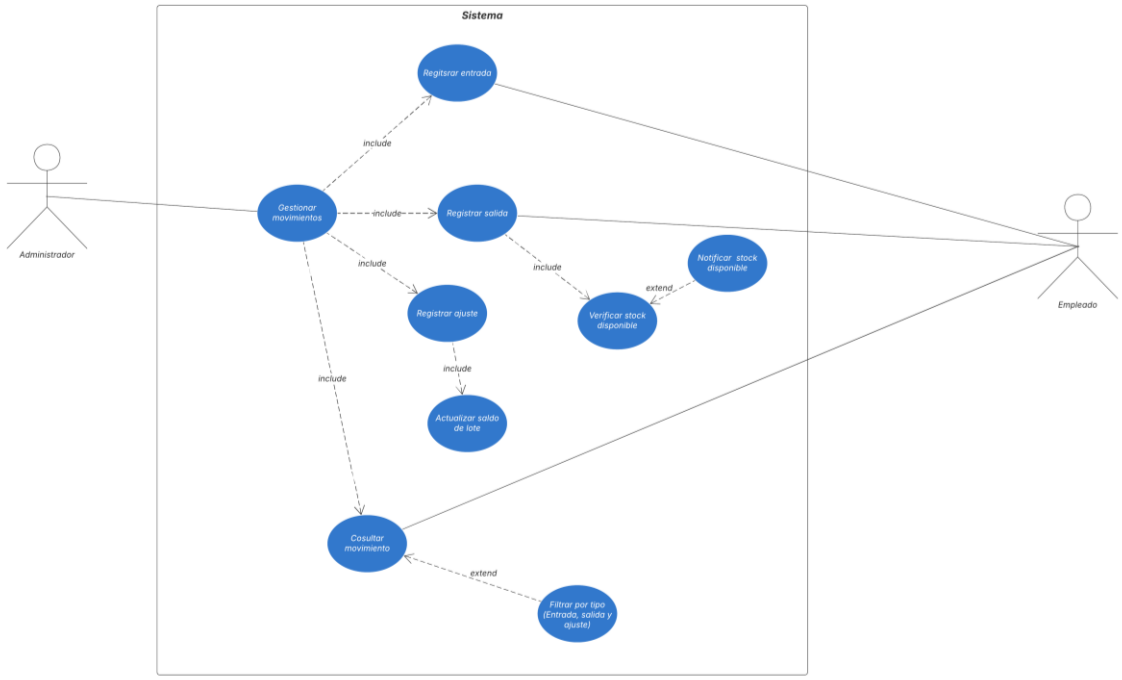
6. Si el número de lote ya existe para el mismo producto, el prototipo muestra el mensaje “El lote ya está registrado para este producto.”
7. Si la fecha de caducidad ingresada es menor a la actual, el prototipo muestra “Fecha de caducidad inválida.”

Postcondición: El prototipo mantiene actualizados los lotes de cada producto con sus respectivas cantidades, fechas de caducidad y estados.

3.3.1.4 Requerimiento F3

Figura 4

Caso de uso requerimiento F3



Nota. Diagrama de casos de uso del proceso de gestión de movimientos en el sistema, mostrando las operaciones de entrada, salida y ajuste, así como las interacciones entre los actores. Elaboración propia, 2026.

Actores: Administrador, Empleado.

Descripción: Permite registrar y consultar los movimientos de inventario (entradas, salidas y ajustes) asociados a los lotes de productos. El módulo controla las variaciones de stock, mantiene la trazabilidad de cada operación e identifica al usuario responsable.

El Administrador puede registrar entradas, salidas y ajustes, además de consultar el historial completo de movimientos.

El Empleado puede registrar entradas y salidas, consultar movimientos y recibir notificaciones automáticas cuando el prototipo detecta que la cantidad solicitada en una salida excede el stock disponible. Si el stock es suficiente, la operación se registra sin notificación adicional.

Precondición:

- El usuario ha iniciado sesión (ADMIN/EMPLEADO).
- Existen productos y lotes registrados con stock disponible para salidas.

Flujo principal:

1. El usuario accede a “Gestionar movimientos”.
2. El prototipo muestra la lista de movimientos por lote y producto.
3. El usuario selecciona la acción:
    - Registrar entrada: lote, motivo (Compra/Devolución), cantidad.
    - Registrar salida: producto y cantidad. El prototipo selecciona automáticamente el lote por FIFO (más antiguo disponible).
    - Registrar ajuste (solo ADMIN): motivo y cantidad corregida.
    - Consultar movimientos: filtra por fechas, tipo, producto o usuario.
4. El prototipo valida datos y actualiza saldos del lote (suma/resta según el tipo).
5. El prototipo registra fecha, usuario, tipo y saldo final.
6. El prototipo muestra confirmación.

Flujos alternos:

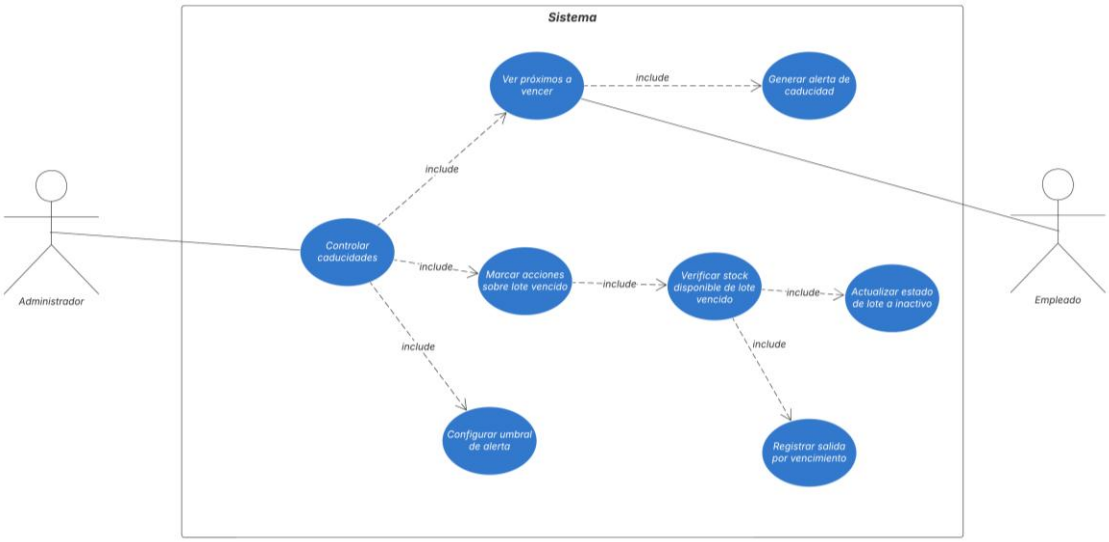
7. Stock insuficiente (salida): muestra “Stock insuficiente para realizar la operación.”
8. Datos inválidos (lote inexistente o cantidad negativa): muestra “Datos inválidos, verifique la información.”

Postcondición: Movimientos registrados con trazabilidad completa (fecha, usuario, tipo, saldo final).

3.3.1.5 Requerimiento F4

Figura 5

Caso de uso requerimiento F4



Nota. Diagrama de casos de uso del proceso de control de caducidades en el prototipo, mostrando las acciones y relaciones entre los actores para la gestión de lotes próximos a vencer. Elaboración propia, 2026.

Actores: Administrador, Empleado.

Descripción: Permite monitorear los productos y lotes próximos a vencer, generar alertas automáticas, verificar el estado y el stock de los lotes vencidos, y registrar las acciones correctivas correspondientes.

El Administrador puede configurar el umbral de alerta (en días), ver los lotes próximos a vencer, generar alertas, marcar acciones sobre los lotes vencidos, verificar su stock y registrar las salidas por vencimiento.

El Empleado únicamente puede visualizar los productos y lotes próximos a vencer para informar al Administrador.

Precondición:

- El usuario ha iniciado sesión (ADMIN/EMPLEADO).
- Existen productos y lotes registrados con fechas de caducidad.

Flujo principal:

1 ADMIN accede a “Control de caducidades”.
2 El prototipo muestra lotes próximos a vencer según el umbral configurado.
3 Acciones:
    - Configurar umbral (por defecto, 90 días).
    - Ver próximos a vencer.
    - Generar alerta de caducidad.
    - Registrar salida por vencimiento: crea el movimiento y actualiza el estado del lote.
4 El prototipo actualiza la información y muestra confirmación.

Flujos alternos:

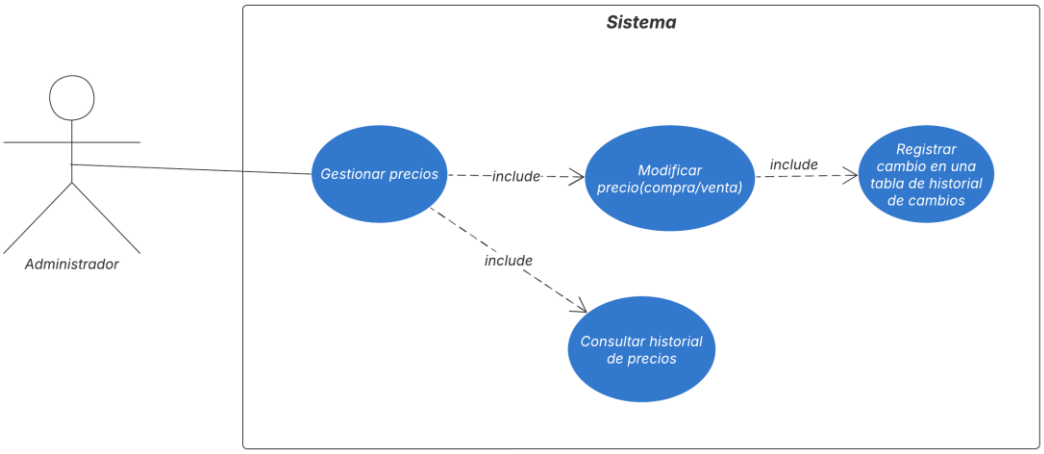
5 Sin lotes próximos a vencer: “No se encontraron lotes dentro del umbral de caducidad.”
6 Umbral < 30 días: “El umbral mínimo permitido es de 30 días.”

Postcondición: Lotes y alertas actualizados, acciones registradas con trazabilidad.

3.3.1.6 Requerimiento F5

Figura 6

Caso de uso requerimiento F5



Nota. Diagrama de casos de uso del proceso de gestión de precios en el prototipo, mostrando las operaciones relacionadas con la modificación y el historial de precios. Elaboración propia, 2026.

Actores: Administrador.

Descripción: Este caso de uso representa el proceso mediante el cual el prototipo registra automáticamente las modificaciones en los precios de compra o venta de los productos.

Cuando el Administrador actualiza los precios desde el módulo “Gestionar productos”, el prototipo almacena los valores anteriores y nuevos, la fecha del cambio y el usuario responsable en la tabla Historial de precios, garantizando la trazabilidad y respaldo de la información sin requerir una acción adicional por parte del usuario.

Precondición:

- El usuario ha iniciado sesión como Administrador.
- Existen productos registrados en el prototipo.

Flujo principal:

1. El Administrador accede al módulo “Gestionar productos”.
2. El prototipo muestra la lista de productos registrados.
3. El Administrador selecciona un producto existente y actualiza su precio de compra.
4. El prototipo recalcula automáticamente el precio de venta según el parámetro configurado (por defecto, 30%).
5. El prototipo valida que los valores ingresados sean correctos (mayores a cero).
6. El prototipo actualiza la información del producto con los nuevos precios.
7. Automáticamente, el prototipo registra en el Historial de precios los valores anteriores y nuevos, junto con la fecha y el usuario que realizó la modificación.
8. El prototipo muestra un mensaje confirmando que la actualización del producto fue exitosa.

Flujos alternos:

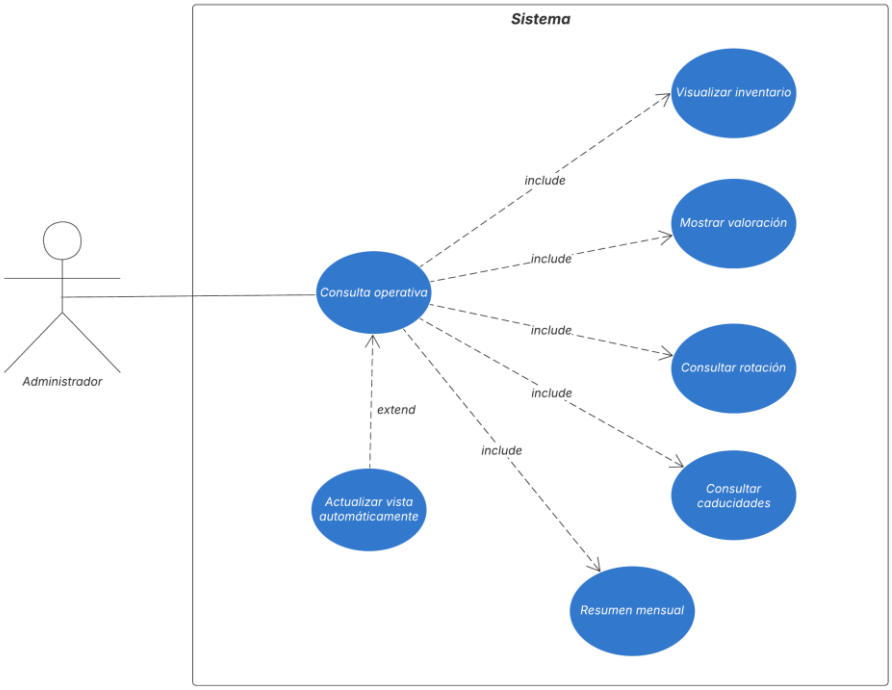
9. Si los valores ingresados son iguales a los precios actuales, el prototipo no genera registro en el historial.
10. Si se ingresa un valor negativo, el prototipo muestra el mensaje: “Precio inválido. Ingrese un valor mayor o igual a cero.”

Postcondición: El prototipo conserva un historial completo de las modificaciones realizadas a los precios de los productos, asegurando trazabilidad, respaldo y auditoría para futuras consultas.

3.3.1.7 Requerimiento F6

Figura 7

Caso de uso requerimiento F6



Nota. Diagrama de casos de uso del proceso de consulta operativa del prototipo, mostrando las acciones disponibles para el Administrador y las vistas analíticas generadas. Elaboración propia, 2026.

Actor: Administrador.

Descripción: Permite al Administrador visualizar información consolidada del inventario, la valoración monetaria de los productos, los niveles de rotación y las caducidades.

El módulo ofrece una visión analítica del estado general del stock para la toma de decisiones gerenciales, y actualiza automáticamente la información al aplicar filtros o al ingresar al módulo.

Precondición:

- El usuario ha iniciado sesión como Administrador.
- Existen productos, lotes y movimientos registrados en el prototipo.

Flujo principal:

1. El Administrador accede al módulo de consulta operativa desde el menú principal.
2. El prototipo muestra un panel con varias vistas:
    - Visualizar inventario: lista los productos con sus lotes, cantidades, totales y estados.
    - Mostrar valoración: presenta el valor monetario por producto y el total general, calculado según el precio de compra.
    - Consultar rotación: muestra los productos más y menos movidos en el período seleccionado.
    - Consultar caducidades: despliega los lotes próximos a vencer o vencidos según el umbral definido.
    - Resumen mensual: presenta el total vendido y la utilidad mensual calculada con base en las salidas por venta.
3. El Administrador puede aplicar filtros de fecha, tipo o marca.
4. El sistema actualiza automáticamente los resultados, recalculando totales, rotación y caducidades en tiempo real.
5. El prototipo muestra los resultados actualizados en pantalla.

Flujos alternos:

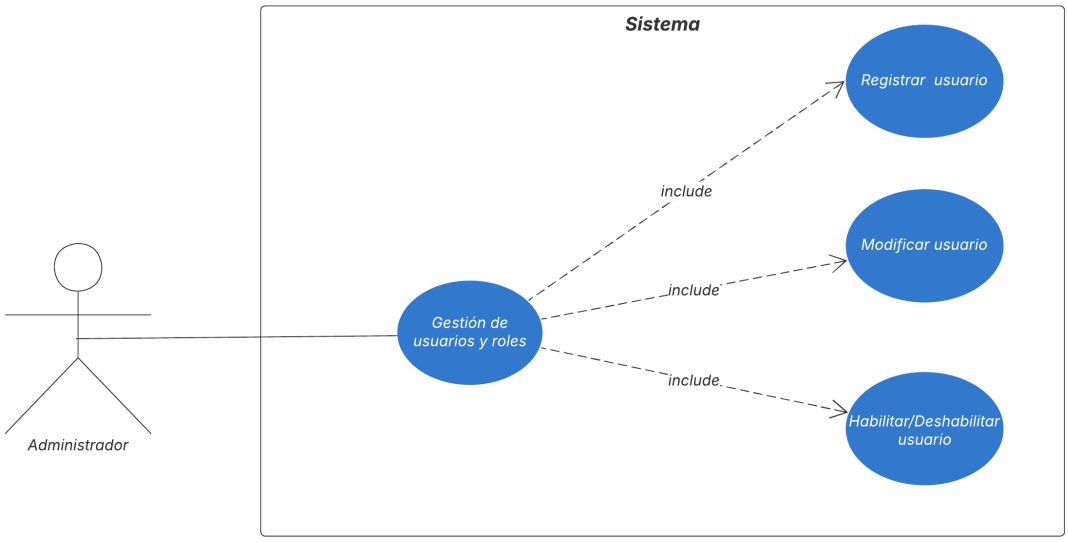
6. Si no existen registros de inventario o movimientos en el período seleccionado, el prototipo muestra el mensaje: “No se encontraron datos disponibles.”
7. Si el Administrador aplica filtros sin coincidencias, el prototipo muestra el mensaje: “No se encontraron productos con los criterios seleccionados.”

Postcondición: El prototipo muestra los indicadores actualizados de inventario, rotación, valoración y caducidad, ofreciendo una visión integral del estado operativo del negocio para la toma de decisiones.

3.3.1.8 Requerimiento F7

Figura 8

Caso de uso requerimiento F7



Nota. Diagrama de casos de uso del proceso de gestión de usuarios y roles del prototipo, mostrando las acciones permitidas al Administrador para el control de acceso. Elaboración propia, 2026.

Actor: Administrador.

Descripción: Permite al Administrador registrar, modificar y habilitar o deshabilitar usuarios dentro del prototipo.

Este módulo garantiza el control de acceso y la correcta asignación de permisos, diferenciando los roles de Administrador y Empleado.

El Administrador es el único autorizado para crear nuevos usuarios, editar su información y gestionar su estado activo o inactivo.

Precondición:

- El usuario ha iniciado sesión como Administrador.
- Existen registros previos de roles definidos (Administrador y Empleado).

Flujo principal:

1. El Administrador accede al módulo “Gestión de usuarios y roles” desde el menú principal.
2. El prototipo muestra la lista de usuarios registrados, indicando su nombre, rol, estado y opciones disponibles.
3. El Administrador puede realizar las siguientes acciones:
   - Registrar usuario: ingresa los datos personales, cédula, correo electrónico, teléfono y asigna el rol (Administrador o Empleado).
   - Modificar usuario: actualiza los datos personales o el rol asignado.
   - Habilitar o deshabilitar usuario: cambia el estado de acceso sin eliminar el registro del prototipo.
4. El prototipo valida la información ingresada, evitando duplicados en cédula o correo electrónico.
5. El prototipo guarda los cambios realizados.
6. El prototipo muestra un mensaje confirmando la operación ejecutada.

Flujos alternos:

7. Si se intenta registrar un usuario con una cédula o correo ya existente, el prototipo muestra el mensaje: “El usuario ya existe en el prototipo.”
8. Si se dejan campos obligatorios vacíos, el prototipo muestra el mensaje: “Complete todos los campos requeridos antes de continuar.”

Postcondición: El prototipo mantiene actualizada la información de los usuarios, garantizando el control de acceso y la correcta asignación de roles dentro de la aplicación.

3.3.2 Requerimientos Funcionales

Para levantar los requerimientos funcionales se realizaron reuniones y entrevistas con el personal de NutriBoost, donde se comprendió de manera general las funcionalidades que debía contemplar el prototipo de gestión de stock. Las funcionalidades principales del prototipo fueron las siguientes:

F0. Ingresar al sistema

F0.1 El prototipo permitió el acceso mediante usuario y contraseña, diferenciando permisos para administrador y empleado.

F1. Gestionar productos

F1.1 Registrar productos con: nombre, sabor, marca, tipo, precios de compra/venta y estado.

F1.2 Modificar datos del producto.

F1.3 Consultar y filtrar productos por nombre, tipo, marca o código.

F1.4 Deshabilitar productos en lugar de eliminarlos, manteniendo el historial.

F2. Gestionar lotes

F2.1 Registrar lotes por producto con número de lote, fecha de caducidad y cantidad inicial.

F2.2 Consultar lotes por producto y estado (Activo e Inactivo).

F2.3 Actualizar el estado del lote según movimientos o caducidad.

F3. Gestionar movimientos

F3.1 Registrar entradas de productos por lote indicando motivo (Compra, Devolución).

F3.2 Registrar salidas de productos por lote indicando motivo (Venta, Devolución, Vencimiento).

F3.3 Registrar ajustes de inventario, guardando observación y documento de referencia.

F3.4 Calcular y almacenar saldo anterior y saldo final por movimiento.

F3.5 Consultar movimientos por rango de fechas, producto, lote y usuario responsable.

F4. Control de caducidades

F4.1 Listar productos y lotes próximos a vencer según umbral definido.

F4.2 Generar alertas automáticas de caducidad para lotes que superaron el umbral.

F4.3 Marcar y documentar acciones tomadas sobre lotes vencidos (bloqueo o salida por vencimiento).

F5. Gestión de precios

F5.1 Registrar cambios de precio (compra/venta) dejando traza del valor anterior y nuevo.

F5.2 Asociar el cambio de precio al usuario y fecha de modificación.

F5.3 Consultar historial de precios por producto y rango de fechas.

F6. Consulta operativa

F6.1 Visualizar el inventario por producto, con desglose por lote, mostrando: cantidad por lote, total del producto (suma de sus lotes) y estado del lote.

F6.2 En la misma pantalla, mostrar la valoración monetaria del inventario (por producto y total general), usando como base el precio de compra.

F6.3 Visualizar la rotación de productos (los más y los menos movidos) según el período seleccionado, con opción de filtrar por tipo o marca.

F6.4 Visualizar los lotes próximos a vencer y vencidos, usando un umbral de al menos 90 días, configurable únicamente por el Administrador.

F6.5 Visualizar un resumen mensual de ventas, mostrando por cada Año y Mes la Utilidad total y el Total vendido, calculados automáticamente a partir de los movimientos tipo Salida-Venta.

F6.6 La vista se actualiza automáticamente al cambiar los filtros o al abrir el módulo, recalculando los totales, la rotación y el estado de los lotes.

F7. Gestión de usuarios y roles

F7.1 Registrar usuarios con datos básicos y rol (ADMIN/EMPLEADO).

F7.2 Modificar datos de usuarios y habilitar/deshabilitar acceso.

F7.3 Restringir funcionalidades según rol (empleado sin acceso a reportes gerenciales ni gestión de precios).

3.3.3 Requerimientos No Funcionales

Los requerimientos no funcionales definieron las características técnicas y de calidad necesarias para asegurar el correcto funcionamiento del prototipo.

Los principales fueron los siguientes:

NF1. Desempeño y eficiencia

Las operaciones de consulta y registro debían tener tiempos de respuesta menores a 3 segundos.

NF2. Seguridad

El acceso a las funcionalidades estuvo restringido mediante autenticación por usuario y contraseña, con almacenamiento de credenciales mediante hash seguro para evitar guardar contraseñas en texto plano.

NF3. Disponibilidad

El prototipo se desarrolló para funcionar desde cualquier dispositivo con conexión a internet y navegadores modernos (Google Chrome, Mozilla Firefox y Edge).

NF4. Escalabilidad

El prototipo fue diseñado de forma modular, permitiendo integrar a futuro la gestión de ventas y reportes financieros.

NF5. Integridad de datos

Toda la información se almacenó en una base de datos relacional, manteniendo la consistencia y respaldo de la información.

3.3.4 Aprobación de requerimientos por parte del cliente

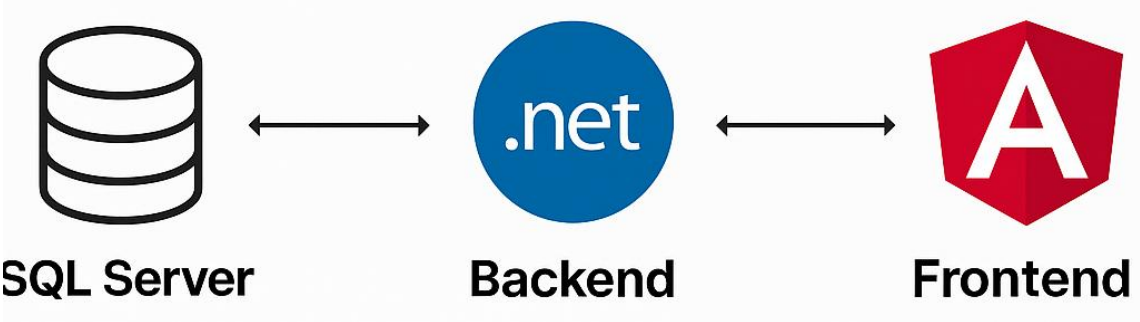
El cliente aprueba: A través de una reunión efectuada, el Product Owner funcional (colaborador de NutriBoost) revisó y aprobó la especificación de requerimientos funcionales (F0–F7) y no funcionales, confirmando su alineación con las operaciones reales del negocio. La aprobación incluye observaciones menores y el compromiso de validar cada incremento durante las revisiones de Sprint.

3.4 Diseño de la Arquitectura del Sistema

3.4.1 Vista de despliegue (infraestructura)

Figura 9

Vista de despliegue del prototipo



Nota. Diagrama de la arquitectura de tres capas del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost, mostrando la interacción entre la base de datos (SQL Server), la capa de lógica del negocio (backend .NET Core) y la interfaz de usuario (Frontend Angular). Elaboración propia, 2026.

El prototipo del sistema web para la gestión del control de inventarios de NutriBoost se estructuró con una arquitectura de tres capas basada en SQL Server, .NET Core y Angular, con lo cual se separaron los datos, la lógica y la interfaz de una manera clara y fácil de mantener, esa división permitió que cada parte del sistema tenga un trabajo definido y que el desarrollo se mantenga ordenado durante la construcción del prototipo, evitando mezclar responsabilidades y facilitando el seguimiento de lo que se iba implementando en cada módulo.

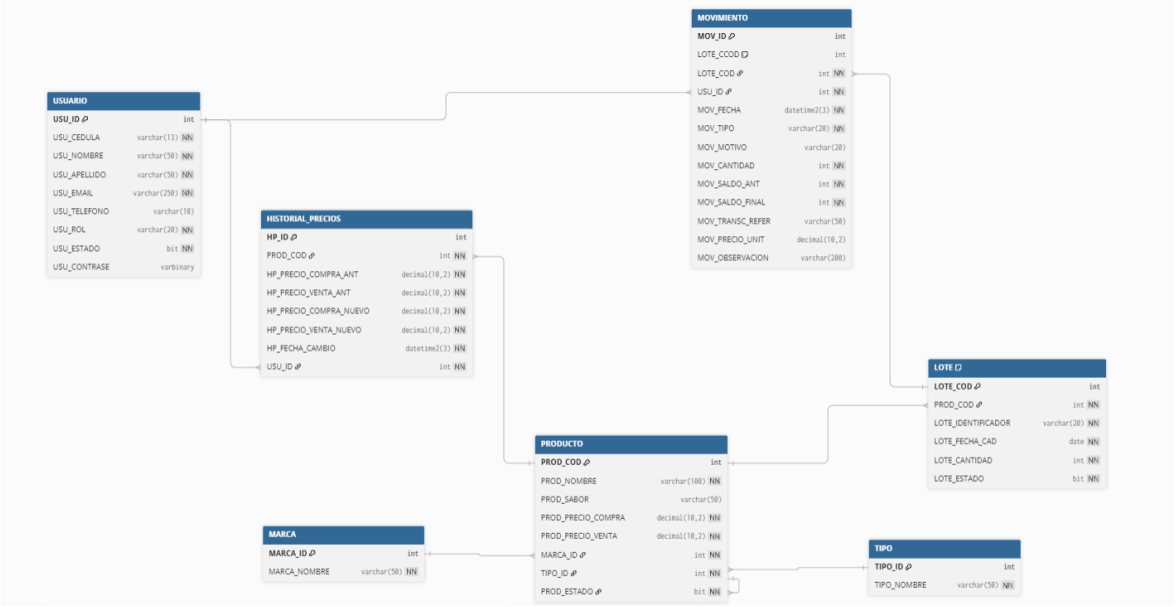
La capa de datos se implementó en SQL Server y almacenó la información de productos, lotes, usuarios y movimientos, manteniendo integridad y consistencia en los registros. El backend en .NET Core gestionó la lógica del negocio y expuso servicios REST para conectar la base de datos con la aplicación. El frontend en Angular brindó una interfaz web para administrar el inventario, consultar información y visualizar reportes, dejando la solución lista para futuras mejoras y ampliaciones.

3.5 Modelado del Sistema

3.5.1 Modelo de datos (ERD)

Figura 10

Modelo entidad–relación



Nota. Diagrama entidad–relación del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost, que muestra la estructura de la base de datos, las entidades principales y sus relaciones. Elaboración propia, 2026.

El modelo entidad–relación del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost define la estructura lógica de la base de datos implementada en SQL Server, estableciendo las relaciones necesarias para garantizar el correcto funcionamiento del control de inventarios.

El diseño está conformado por las entidades principales: USUARIO, PRODUCTO, LOTE, MOVIMIENTO, HISTORIAL_PRECIOS, TIPO y MARCA, vinculadas mediante claves primarias y foráneas que aseguran la integridad referencial.

Cada una cumple una función específica dentro del proceso de gestión:

- USUARIO: administra los datos de acceso, roles y permisos.
- PRODUCTO: almacena la información principal de los artículos junto con su tipo, marca y precios.
- LOTE: registra las fechas de caducidad, cantidad y estado de cada producto.

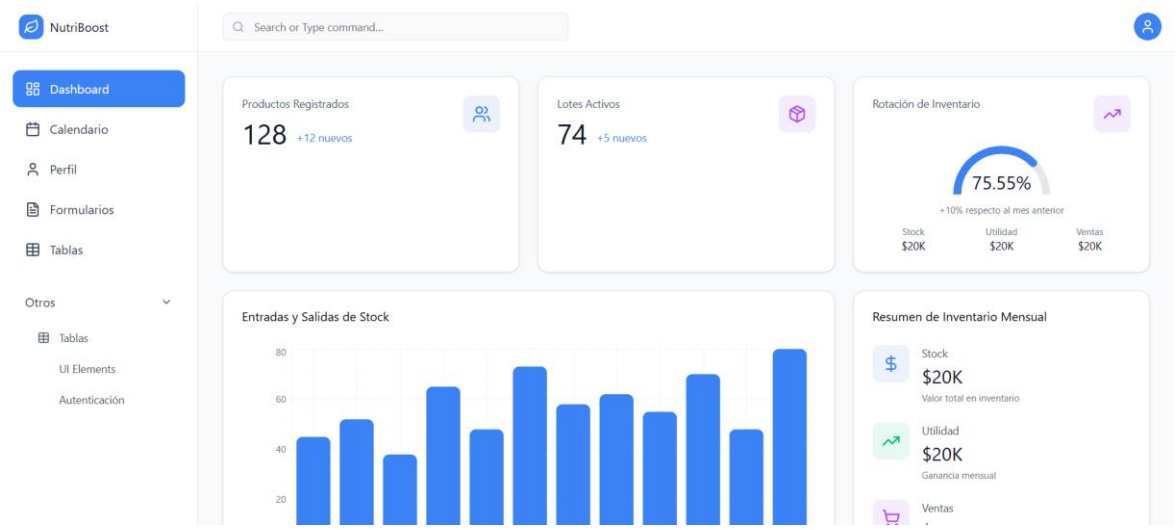
- MOVIMIENTO: controla las entradas, salidas y ajustes de inventario.
- HISTORIAL_PRECIOS: conserva las variaciones de precios con fecha y usuario responsable.
- TIPO y MARCA: permiten clasificar los productos de forma normalizada.

Este modelo relacional garantiza la consistencia, trazabilidad y escalabilidad de los datos, asegurando que todas las operaciones del prototipo se ejecuten de manera integrada y confiable dentro de la arquitectura SQL Server – .NET Core – Angular.

3.6 Diseño de la Interfaz de Usuario

Figura 11

Interfaz principal



Nota. Captura de pantalla de la interfaz principal del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost, donde se visualizan los indicadores generales del sistema, como productos registrados, lotes activos, rotación de inventario y resumen mensual. Elaboración propia, 2026.

Para el desarrollo del prototipo del sistema web para la gestión del control de inventarios de NutriBoost se elaboró una plantilla base que sirvió como referencia para el diseño de todas las pantallas, en esta plantilla se definieron la estructura general, la distribución de elementos y el estilo visual que se mantuvo en todo el sistema, con ello se buscó asegurar uniformidad en la presentación y facilitar que cada módulo conserve la misma lógica de navegación y visualización.

El diseño se trabajó con un estilo limpio y claro, utilizando fondo blanco y una paleta de tonos azules para resaltar elementos importantes como botones, íconos y gráficos, mientras que los textos se mantuvieron en negro y gris para asegurar contraste y legibilidad, dentro de la plantilla se incluyó un menú lateral con secciones principales (Dashboard, Calendario, Perfil, Formularios, Tablas y Autenticación), además de un panel central donde se visualizaron datos relevantes del inventario como productos registrados, lotes activos, rotación e ingresos y salidas de stock, esta plantilla se utilizó como base para el resto de pantallas con el fin de mantener consistencia en colores, posiciones y formato general del prototipo.

Capítulo 4: Desarrollo de la Aplicación

4.1 Preparación del Entorno de Desarrollo

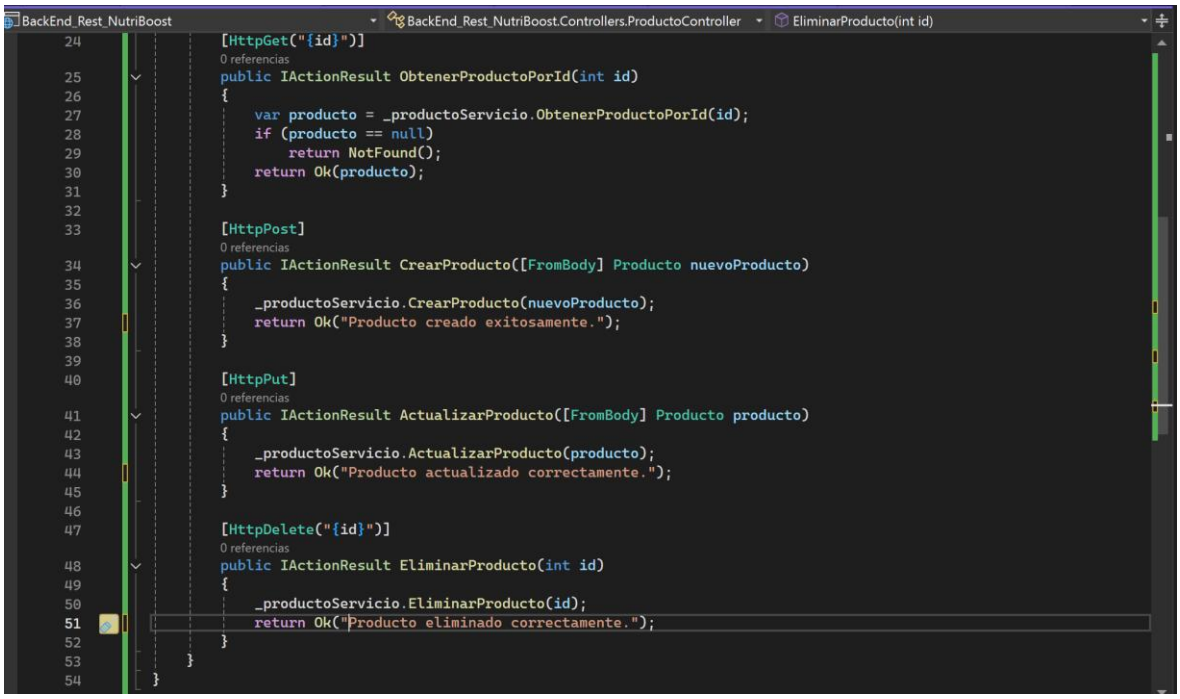
4.1.1 Estándares de codificación

Durante el desarrollo del prototipo del sistema web para la gestión de procesos críticos del control de inventarios de NutriBoost se aplicaron estándares de codificación orientados a mantener coherencia, legibilidad y facilidad de mantenimiento tanto en el backend desarrollado con .NET y C# como en el frontend implementado con Angular y TypeScript, bajo esta pauta en el backend se utilizó PascalCase para clases, métodos y propiedades con nombres como ProductoController y LoteController, mientras que camelCase se reservó para variables locales y parámetros y el guion bajo se mantuvo en campos privados como _productoServicio, adicionalmente el código se organizó en capas diferenciadas para datos, lógica de negocio y presentación mediante API REST, incorporando validación de datos y manejo controlado de excepciones como parte del orden general del proyecto.

Por su parte en el frontend se siguieron las guías oficiales de Angular con una nomenclatura consistente entre componentes, servicios y archivos, empleando camelCase en variables y métodos como cargarProductos() o getProductos(), usando PascalCase en componentes y servicios como ProductoComponent y LoteService, manteniendo kebab-case en nombres de archivos como producto-list.component.ts, con esa estructura el sistema se distribuyó por módulos funcionales y se apoyó en inyección de dependencias y comunicación por servicios para reducir acoplamiento y sostener escalabilidad, finalmente el trabajo se gestionó con control de versiones mediante Git y GitHub registrando avances por Sprint y dejando identificadas las versiones entregadas.

Figura 12

Aplicación de estándares de codificación en el controlador Producto



Nota. Fragmento del código del backend del prototipo NutriBoost que ejemplifica la implementación de operaciones CRUD (GET, POST, PUT y DELETE) en el controlador ProductoController. Elaboración propia, 2026.

4.1.2 Entorno de desarrollo

El entorno de desarrollo se configuró con las siguientes herramientas:

- **Backend:** Visual Studio 2022, .NET 8.0, Entity Framework Core y SQL Server Management Studio.
- **Frontend:** Visual Studio Code, Node.js v20, Angular CLI v17 y Angular Material.
- **Control de versiones:** Git y GitHub para sincronizar cambios entre iteraciones.
- **Pruebas y despliegue:** navegador Google Chrome y Postman para la validación de endpoints REST.

Esta configuración permitió un flujo de trabajo ágil y controlado durante las iteraciones de desarrollo.

4.2 Implementación de la Funcionalidad

4.2.1 Desarrollo iterativo con Scrum

El desarrollo del prototipo se realizó bajo la metodología ágil Scrum, distribuido en dos iteraciones o Sprints de dos semanas cada una.

Cada Sprint permitió entregar incrementos funcionales del prototipo, incorporando la retroalimentación del cliente al finalizar cada ciclo.

4.2.1.1 Sprint 1

El primer Sprint tuvo una duración de dos semanas y se centró en la implementación completa del backend del prototipo NutriBoost, incluyendo la base de datos, la lógica de negocio y los servicios de comunicación.

Este ciclo permitió establecer la arquitectura técnica sobre la que posteriormente se construiría la interfaz web, asegurando la correcta gestión de los procesos de inventario desde el nivel del servidor.

Objetivos del Sprint

- Diseñar y construir la base de datos principal en SQL Server, con todas sus relaciones, llaves foráneas y restricciones de integridad.
- Crear vistas, procedimientos almacenados y triggers que automaticen procesos de control, auditoría y cálculo de precios.
- Desarrollar los servicios de negocio en .NET 8 (API REST), incluyendo las operaciones CRUD y la conexión con la base de datos mediante Entity Framework Core.
- Probar y documentar los endpoints de la API para su futura integración con el frontend Angular.
- Garantizar que todo el backend estuviera funcional y estable antes de pasar al desarrollo de la interfaz.

Alcance comprometido

- **HU1** (Autenticación): inicio de sesión y control básico por rol.
- **HU2** (Productos): registro, edición, consulta y deshabilitación lógica.
- **HU3** (Editar producto): modificaciones de datos del producto con persistencia en BD.

- **HU10** (Gestión de usuarios): registro, edición y deshabilitación de usuarios para control de accesos.

Actividades realizadas

Durante este primer Sprint se ejecutaron las siguientes tareas técnicas:

- Diseño e implementación de la base de datos:
    - Se crearon las tablas principales PRODUCTO, MARCA, TIPO, LOTE, MOVIMIENTO, HISTORIAL_PRECIOS y USUARIO, cumpliendo con la tercera forma normal (3FN) para garantizar consistencia e integridad referencial.
- Creación de procedimientos y disparadores (triggers):
    - Se programaron funciones y triggers que actualizan automáticamente el stock de productos tras cada registro de movimiento, y que registran los cambios de precios en la tabla HISTORIAL_PRECIOS.
- Además, se generaron vistas SQL para resumir la utilidad mensual, el historial de ventas y la rotación de inventario.
- Implementación del backend en .NET:
    - Se desarrollaron los controladores ProductoController, LoteController y MovimientoController, junto con sus respectivos servicios en la capa lógica.
- Cada endpoint fue probado mediante Postman, verificando respuestas correctas en formato JSON y validaciones de error.
- Configuración del contexto de datos:
    - Se implementó la clase NutriBoostContexto que maneja las entidades y relaciones mediante Entity Framework Core, permitiendo operaciones directas sobre la base de datos sin consultas SQL explícitas.

Resultados y validación

- Backend operativo con persistencia y reglas de negocio; CRUD de productos funcionando.
- API REST estable y documentada, conectada a SQL Server.
- Validación del cliente en Sprint Review: operaciones CRUD correctas y cálculos automáticos coherentes.

Evidencias:

Las capturas correspondientes a las evidencias desarrolladas durante este sprint se encuentran en la sección *Evidencia del Sprint 1*.

Métricas del Sprint 1

- Velocidad: 14 puntos (HU1=3, HU2=5, HU3=3, HU10=3).
- Cumplimiento del objetivo de Sprint: 100 %.
- Cumplimiento DoD: 100 % de PBIs comprometidos.
- Retrabajo/incidencias: 1 ajuste menor (validación).
- Satisfacción del cliente: 5/5.

Resultados y validación

Durante la reunión de Sprint Review, el cliente validó el correcto funcionamiento del backend, verificando que las operaciones CRUD respondieran adecuadamente y que los cálculos automáticos del inventario fueran precisos.

Se aprobaron los resultados del Sprint con observaciones mínimas sobre la ampliación de reportes, las cuales se programaron para el siguiente ciclo de desarrollo.

Tabla 7

Resumen de actividades y resultados del Sprint 1

Tarea	Descripción	Estado
Diseño de base de datos	Tablas, relaciones, vistas y triggers implementados	Completada
Desarrollo del backend	Servicios REST con Entity Framework Core	Completada
Pruebas de endpoints	Validación en Postman y Swagger	Completada
Revisión con cliente	Aprobación del funcionamiento general	Aprobada

Nota. Resumen de las actividades ejecutadas durante el Sprint 1 del prototipo NutriBoost, que incluyó la implementación del backend y la verificación de su correcto funcionamiento. Elaboración propia, 2026.

4.2.1.2 Sprint 2

El segundo Sprint tuvo una duración de dos semanas y se centró en la implementación de la interfaz web en Angular y su integración con el backend desarrollado en el Sprint anterior, consolidando así la funcionalidad completa del prototipo NutriBoost.

Este ciclo permitió habilitar el uso del sistema desde el navegador, conectando los módulos de productos, lotes, movimientos y consulta operativa con la API REST, de manera que el cliente pudiera interactuar con el inventario en un entorno gráfico y validarlo de forma práctica.

Objetivos del Sprint

- Implementar la interfaz web en Angular para los módulos de productos, lotes, movimientos, control de caducidades y consulta operativa.
- Conectar la interfaz Angular con la API REST del backend, consumiendo los endpoints existentes para ejecutar las operaciones CRUD.
- Incorporar validaciones visuales, mensajes de retroalimentación y restricciones según rol (ADMIN/EMPLEADO).
- Probar el flujo de información extremo a extremo entre frontend, backend y base de datos, usando datos simulados y casos de prueba.
- Presentar el prototipo funcionando al cliente y recoger su retroalimentación final sobre la usabilidad y el comportamiento del sistema.

Alcance comprometido

- **HU4 (Inventario):** visualización de productos con stock disponible y desglose por lote.
- **HU5 (Movimientos):** registro de entradas, salidas y ajustes de inventario con actualización automática de stock.
- **HU6 (Reportes de inventario):** generación de reportes filtrados por producto o período.
- **HU7 (Lotes):** gestión de lotes (registro, consulta y actualización de estado) desde la interfaz web.
- **HU8 (Caducidades):** visualización de productos y lotes próximos a vencer, con alertas según el umbral configurado.

- **HU9 (Consulta operativa):** panel analítico con inventario, valoración monetaria, rotación y resumen mensual.

Actividades realizadas

Durante este segundo Sprint se realizaron las siguientes actividades:

- Diseño y construcción de la interfaz en Angular
- Se implementó la estructura base del frontend con Angular y Angular Material, definiendo el menú lateral, la barra de navegación y el enrutamiento entre módulos.
- Se desarrollaron los componentes para la gestión de productos, lotes y movimientos, utilizando tablas, formularios y diálogos modales para el registro y edición de información.
- Integración con la API REST del backend.
- Se crearon servicios HTTP en Angular para consumir los endpoints de productos, lotes, movimientos, historial de precios y reportes.
- Se validó el envío y recepción de datos en formato JSON, verificando la correspondencia entre los modelos del frontend y las entidades del backend.
- Implementación de la consulta operativa y control de caducidades.
- Se desarrolló una vista de consulta operativa que muestra el inventario consolidado por producto, la valoración monetaria y los lotes próximos a vencer.
- Se implementaron filtros por rango de fechas, tipo y marca, así como la actualización dinámica de los indicadores al cambiar los filtros.
- Se añadió una vista específica para el control de caducidades, donde el Administrador puede visualizar los lotes próximos a vencer y vencidos.

Validaciones y experiencia de usuario

- Se incorporaron validaciones de campos obligatorios, formatos numéricos y rangos de fechas en los formularios de productos, lotes y movimientos.
- Se habilitó/ocultó funcionalidad según el rol del usuario (ADMIN/EMPLEADO), restringiendo el acceso a reportes y gestión de precios para el Empleado.
- Se revisó la coherencia visual de la interfaz (colores, tipografía, espaciados) con la plantilla base definida en el diseño de UI.

Resultados y validación

- Interfaz web en Angular operativa para productos, lotes, movimientos, caducidades y consulta operativa.
- Flujo extremo a extremo funcionando: el usuario registra/consulta información desde el navegador y los cambios se reflejan en la base de datos SQL Server.
- Reglas de negocio (FIFO, caducidades, trazabilidad de movimientos y precios) visibles y comprobables a través de la interfaz.
- Validación del cliente en Sprint Review: prototipo considerado funcional para demostrar el control de inventario, con comentarios favorables sobre la claridad de la interfaz y la organización de la información.

Evidencias:

Las capturas correspondientes a las evidencias desarrolladas durante este sprint se encuentran en la sección *Evidencia del Sprint 2*.

Métricas del Sprint 2

- **Velocidad:** 31 puntos (HU4=5, HU5=8, HU6=5, HU7=5, HU8=3, HU9=5).
- **Cumplimiento del objetivo de Sprint:** 100 %.
- **Cumplimiento DoD:** 100 % de PBIs comprometidos.
- **Retrabajo/incidencias:** 1 ajuste menor (mensajes de error).
- **Satisfacción del cliente:** 5/5.

Resultados y validación

En la Sprint Review del segundo Sprint, el cliente validó el funcionamiento de la interfaz web y el flujo completo de uso del prototipo, comprobando que las operaciones de registro, consulta y control de inventario se ejecutaran de forma coherente con los procesos reales de la tienda.

Se aprobaron los resultados del Sprint 2, indicando que el prototipo cumple con las funcionalidades críticas del control de inventario (registro de productos, gestión de lotes, movimientos, caducidades y reportes básicos) y puede utilizarse como base para futuras ampliaciones del sistema.

Tabla 8

Resumen de actividades y resultados del Sprint 2

Tarea	Descripción	Estado
Implementación del frontend	Desarrollo de componentes Angular para productos, lotes, movimientos y consulta operativa	Completada
Integración frontend–backend	Consumo de la API REST, envío/recepción de datos JSON y manejo de errores	Completada
Validaciones y control de acceso	Validaciones en formularios y restricción de funcionalidades según rol ADMIN/EMPLEADO	Completada
Revisión con cliente	Demostración del prototipo integrado y aprobación de la funcionalidad general	Aprobada

Nota. Resumen de las actividades del Sprint 2 del prototipo NutriBoost, con implementación de la interfaz en Angular, integración con el backend mediante API REST y validación del flujo operativo del sistema. Elaboración propia, 2026.

4.3 Plan de Pruebas

4.3.1 Tipos de Pruebas

4.3.1.1 Pruebas unitarias

Las pruebas unitarias se realizaron para comprobar el comportamiento correcto de funcionalidades puntuales del sistema antes de la validación final con el cliente; no se incorporó un framework de automatización como xUnit, NUnit o Jasmine, así que la comprobación se ejecutó de forma manual, caso por caso, con datos controlados y rastreables; el foco estuvo en operaciones CRUD y validaciones de negocio en los módulos de productos, lotes y movimientos, revisando respuestas esperadas, restricciones de campos y manejo de entradas no válidas; cada ejecución se contrastó entre lo mostrado en la interfaz y lo registrado en la base de datos, cuidando que los resultados se mantengan coherentes cuando se repite el mismo escenario.

Las capturas y resultados de estas pruebas se presentan en *6.5 Pruebas* **unitarias**.

4.3.1.2 Pruebas de aceptación

Las pruebas de aceptación se realizaron al finalizar la integración del prototipo, con participación del cliente, con el objetivo de confirmar que el sistema cumple con lo requerido para el control operativo del inventario de NutriBoost. Para esta validación se ejecutaron procesos completos representativos del uso real, como registro de productos, gestión de lotes, movimientos de entrada/salida y revisión de información operativa. Con base en los resultados observados, el cliente confirmó la conformidad del sistema respecto a las necesidades definidas.

Las evidencias de aceptación y la validación final del cliente se incluyen en*6.6 Pruebas de* **aceptación**.

4.3.2 Criterios de Aceptación

Una funcionalidad se consideró aceptada cuando, al ser ejecutada, produjo el resultado esperado según el requerimiento, mantuvo coherencia entre lo visualizado en la aplicación y lo registrado en la base de datos, y permitió completar el proceso sin inconsistencias. La aceptación final se confirmó con la revisión del cliente sobre los módulos principales y los resultados obtenidos en los procesos probados.

4.3.3 Evidencias de pruebas

Las evidencias generadas durante la ejecución de las pruebas se registraron y se anexaron como soporte del proceso de validación del prototipo. En esta sección de *6.5 Pruebas* **unitarias** se incluyen las evidencias correspondientes a las pruebas unitarias, tales como respuestas de endpoints, validaciones realizadas y resultados obtenidos en cada operación evaluada. En el apartado de*6.6 Pruebas de* **aceptación** se presenta la evidencia correspondiente a la revisión final, donde se documenta la validación del cliente sobre los procesos principales del sistema y los resultados confirmados durante la revisión final.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

1. Durante el desarrollo del prototipo del sistema web para el control de inventarios de NutriBoost se consolidó en una sola plataforma el manejo de productos, lotes, movimientos y fechas de caducidad trabajando sobre una base de datos única, además se definieron reglas claras para el control y la actualización del stock, la verificación del funcionamiento se realizó mediante pruebas en el backend con Swagger y Postman junto con validaciones directas en SQL Server y revisiones desde la interfaz web en Angular, con estos controles se comprobó que las operaciones principales de registro, consulta y modificación se ejecutaron de forma coherente y que la información se mantuvo centralizada y consistente en todo el sistema.
2. La arquitectura implementada se basó en SQL Server para la capa de datos, una API REST en .NET 8 para la lógica de negocio y una interfaz web en Angular para la interacción del usuario, esta separación por capas se ajustó a los procesos críticos del inventario definidos en el alcance y facilitó el desarrollo ordenado de los módulos, a partir de esta estructura se implementaron funciones clave como el control por lotes, la actualización automática del stock y el registro del historial de cambios de precios, manteniendo integridad y consistencia en los datos conforme se integraban los componentes del prototipo.
3. Aunque el sistema construido corresponde a un prototipo y no a un producto final desplegado en producción se validó su funcionamiento frente a los requerimientos funcionales definidos mediante pruebas de registro, consulta y actualización en los módulos desarrollados, con ello se obtuvo evidencia de que la solución es técnicamente viable y operativa para el contexto del negocio, en consecuencia el prototipo quedó como una base sólida para futuras mejoras y como referencia para un posible despliegue en un entorno real si NutriBoost decide continuar con la implementación del sistema.

5.2 Recomendaciones

1. Se recomienda que la tienda NutriBoost utilice el prototipo como referencia para formalizar sus procesos de inventario aun cuando no se lo implemente de forma directa en producción, la estructura de datos, las reglas de negocio y las pantallas diseñadas pueden servir como guía para ordenar el registro actual de productos, lotes y movimientos, con ello se reduce la dependencia de controles informales y se mejora la disciplina en el manejo de la información, esta referencia también permite estandarizar criterios de actualización y consulta para que el inventario mantenga consistencia en el tiempo.
2. Resulta recomendable documentar y conservar de manera organizada los artefactos generados durante el proyecto como el modelo de datos, los diagramas, el código fuente, los casos de uso y la descripción de sprints, este respaldo facilita que el prototipo pueda integrarse o extenderse en el futuro como parte de una arquitectura más amplia si el negocio requiere nuevos módulos o funcionalidades, al mantener esta documentación actualizada la tienda puede conectar el sistema con otros servicios conforme crezca su operación sin perder coherencia en los procesos y sosteniendo la trazabilidad de la información registrada.
3. También es conveniente aprovechar la experiencia obtenida con la aplicación de Scrum en el desarrollo del prototipo como referencia para proyectos posteriores de titulación o iniciativas internas de NutriBoost, la planificación por sprints, el manejo de un Product Backlog alineado a requerimientos funcionales y el uso de métricas básicas permitieron controlar el avance del trabajo y mantener validación constante con el usuario, replicar estas prácticas en otros contextos puede contribuir a organizar mejor tareas y prioridades y a mantener una comunicación más clara con las personas involucradas en futuros desarrollos.

Bibliografía

Amazon Web Services . (s. f.). *AWS*. Obtenido de Diferencia entre SOAP y REST: https://aws.amazon.com/es/compare/the-difference-between-soap-rest/

Amazon Web Services. (s.f.). *Amazon Web Services*. Obtenido de https://aws.amazon.com/es/what-is/web-application/

Amazon Web Services. (s.f.). *Amazon Web Services, Inc.* Obtenido de https://aws.amazon.com/es/what-is/ide/

Amazon Web Services, Inc. (18 de marzo de 2024). *AWS*. Obtenido de https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/

Angular Team. (2025). *Angular*. Obtenido de https://docs.angular.lat/overview

García Sandoval, M. G., Ariza Torrado, H. D., Pinzón, M. L., & Flórez Fuentes, A. S. (2016). Buenas prácticas aplicadas a la implementación colaborativa de aplicativos web. *Revista Mundo FESC*, 23–30. Obtenido de file:///C:/Users/asusv/Downloads/Dialnet-BuenasPracticasAplicadasALaImplementacionColaborat-5351802.pdf

Gil Vera, V. D., Gomes Da Silva, C. R., Gil Vera, J. C., & Teutsch, J. (2018). Frameworks para el desarrollo de prototipos web: Un caso de aplicación. *Lámpsakos*, 40-53. doi:https://doi.org/10.21501/21454086.2065

GoDaddy. (16 de Abril de 2024). *GoDaddy*. Obtenido de https://www.godaddy.com/resources/latam/tecnologia/metodologia-xp-extreme-programming-que-es

González, J. F. (s.f.). *Introducción a las metodologías ágiles.* Barcelona: Fundación para la Universitat Oberta de Catalunya (FUOC). Obtenido de https://gc.scalahed.com/recursos/files/r161r/w25941w/S02R02.pdf

González-Longatt, F. M. (2007). *Introducción a los sistemas de información: Fundamentos.* Universidad Experimental Politécnica de la Fuerza Armada. Obtenido de https://www.uv.mx/personal/artulopez/files/2012/08/fundamentossistemasinformacion.pdf

Hernández, J. A., & Serrano, L. F. (2019). Metodologías ágiles para la gestión de proyectos de software: Un análisis comparativo. *Revista Ibérica de Sistemas y Tecnologías de Información*, E19. Obtenido de https://www.redalyc.org/pdf/4962/496250736004.pdf

Herrera Infante, D. C. (20 de Febrero de 2025). *Hostinger Tutorials*. Obtenido de https://www.hostinger.com/es/tutoriales/seguridad-en-aplicaciones-web

International Organization for Standardization. (2018). *ISO 9241-11:2018 Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts.* ISO. Obtenido de https://www.iso.org/obp/ui/en/#iso:std:iso:9241:-11:ed-2:v1:en

Kanban University. (Febrero de 2021). *Kanban University*, 1. Obtenido de https://resources.kanban.university/wp-content/uploads/2021/08/The-Official-Kanban-Guide_Spanish_A4.pdf

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software.* Tesis de licenciatura, Universidad Nacional del Centro de la Provincia de Buenos Aires. Obtenido de https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Marqués, M. (2011). *Bases de datos.* Castellón de la Plana: Publicacions de la Universitat Jaume I. Obtenido de https://bdigital.uvhm.edu.mx/wp-content/uploads/2020/05/Bases-de-Datos.pdf

Microsoft. (12 de septiembre de 2023). *Microsoft Learn*. Obtenido de https://learn.microsoft.com/es-es/devops/develop/git/what-is-version-control

Microsoft. (8 de Septiembre de 2025). *Microsoft Learn*. Obtenido de https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver17

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (Novena ed.). McGraw-Hill Education.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game.* Scrum.org. Scrum.org. Obtenido de

https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2014). *Fundamentos de bases de datos* (Quinta ed.). McGraw-Hill Interamericana de España, S.L.

Sommerville, I. (2011). *Ingeniería de software.* Ciudad de México: Pearson Educación de México. Obtenido de https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

Vue.js Core Team. (2025). *Vue.js*. Obtenido de https://vuejs.org/guide/introduction.html

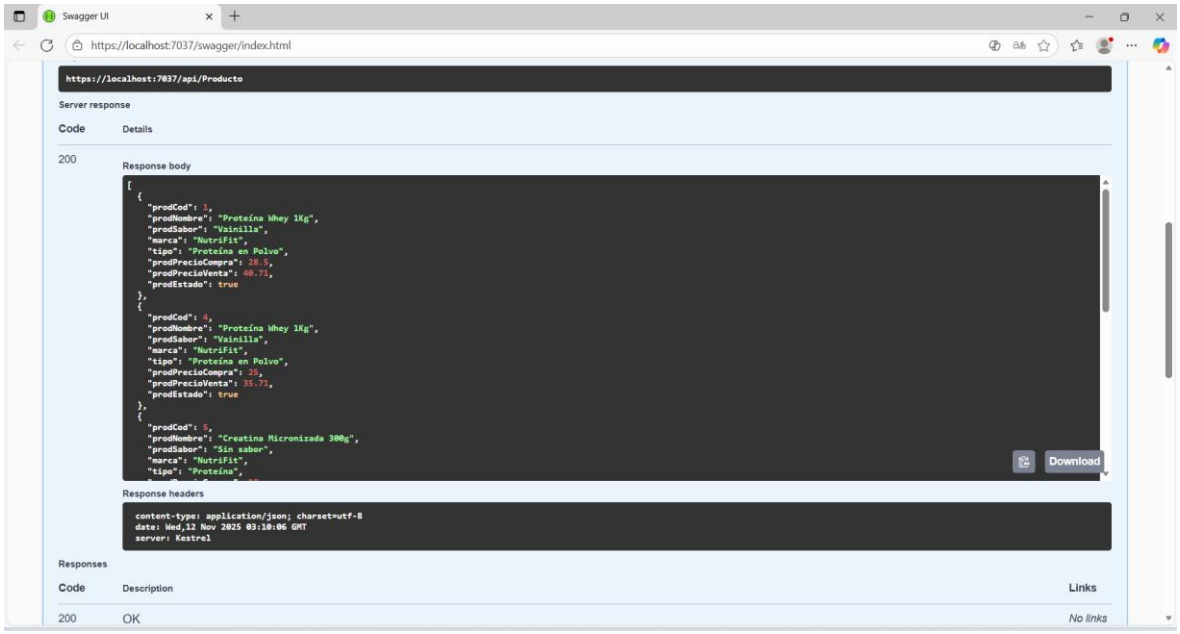
Zumba Gamboa, J. P., & León Arreaga, C. A. (2018). Evolución de las metodologías y modelos utilizados en el desarrollo de software. *INNOVA Research Journal*, 20-33. Obtenido de file:///C:/Users/ASUS/Downloads/Dialnet-EvolucionDeLasMetodologiasYModelosUtilizadosEnElDe-6777227.pdf

Anexos

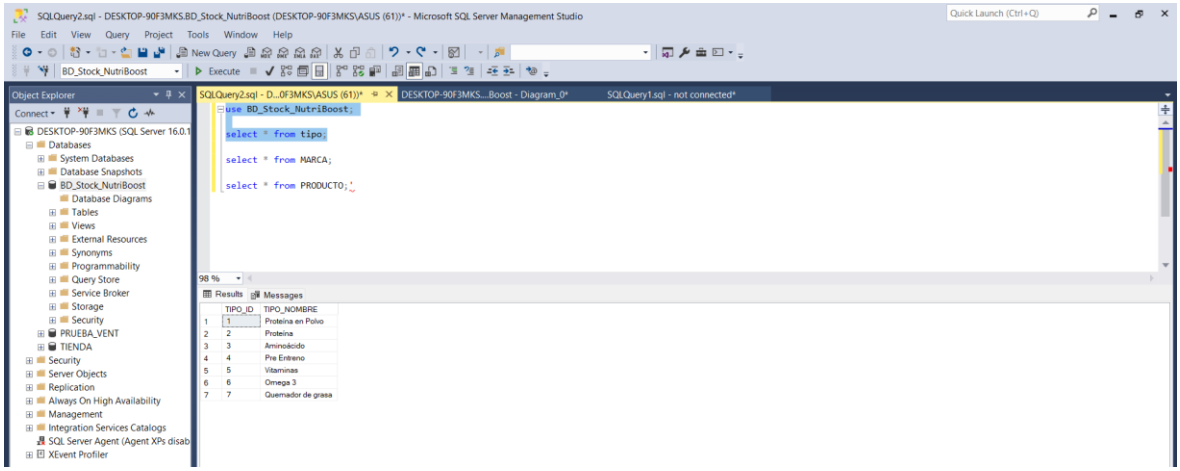
6.1 Evidencia del Sprint 1

La visualización en Swagger UI del endpoint GET /api/Producto permitió confirmar que el backend del prototipo de inventario de NutriBoost genera correctamente las respuestas JSON al consultar los productos registrados en la base de datos. De esta forma, la ejecución de consultas en Microsoft SQL Server verificó la inserción adecuada de datos en la tabla TIPO, evidenciando la integridad y consistencia del modelo relacional después de las operaciones realizadas., la solicitud GET /api/Tipo en Postman mostró una respuesta JSON coherente con la información almacenada, lo que confirma la correcta comunicación entre la API REST y la base de datos SQL Server durante las pruebas del backend.

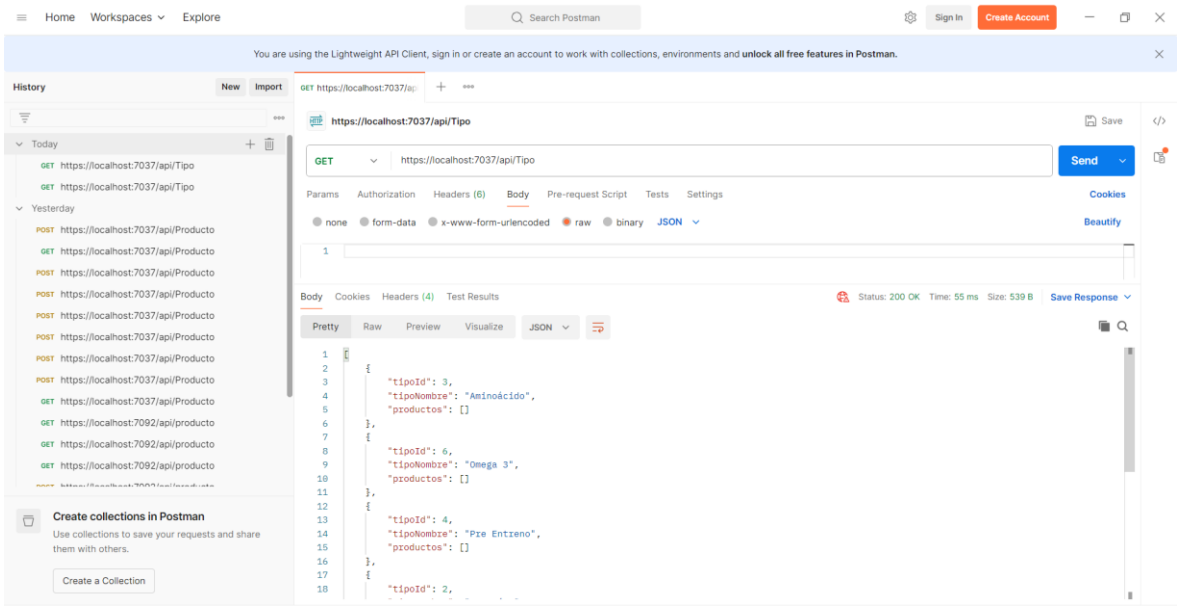
Verificación de endpoints en Swagger para el módulo de productos



Validación de registros en la base de datos SQL Server



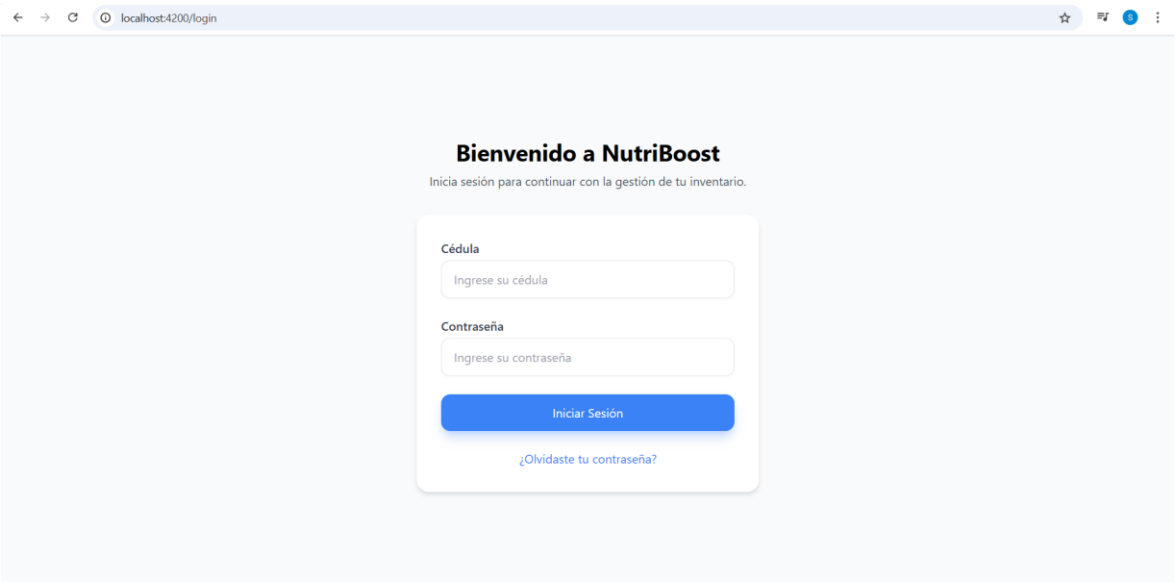
Respuesta JSON del endpoint en Postman para el módulo Tipo



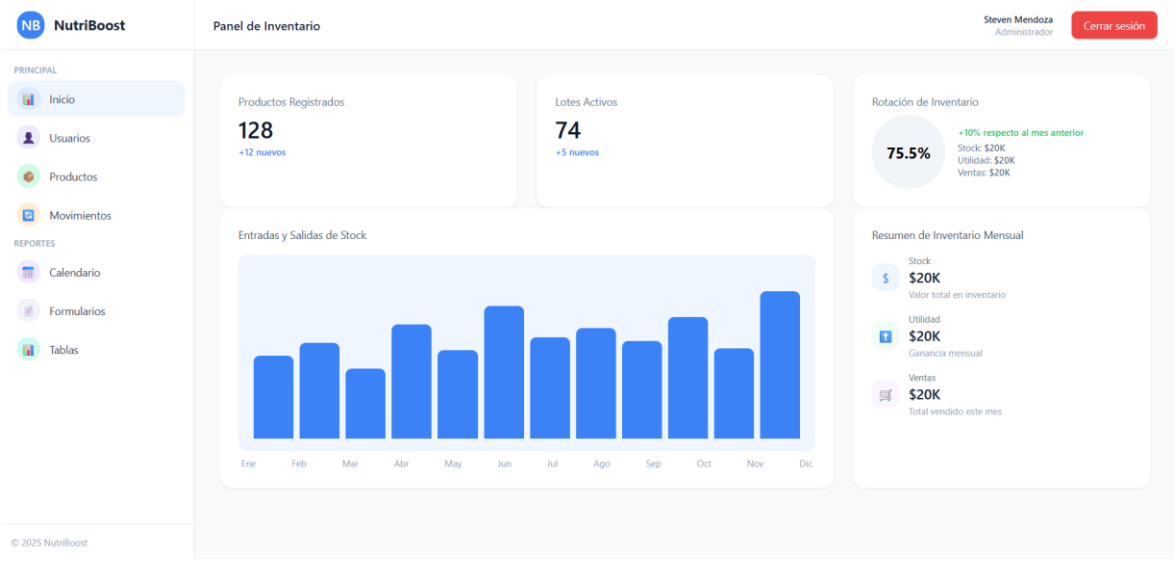
6.2 Evidencia del Sprint 2

Durante el Sprint 2 se validó el funcionamiento de la interfaz web del prototipo NutriBoost, comprobando la correcta visualización de indicadores del inventario, la gestión de productos y el proceso de autenticación del usuario. Las siguientes capturas de pantalla muestran la interacción del usuario con el sistema desde el navegador, verificando la integración completa del frontend en Angular con la API REST desarrollada en el Sprint 1.

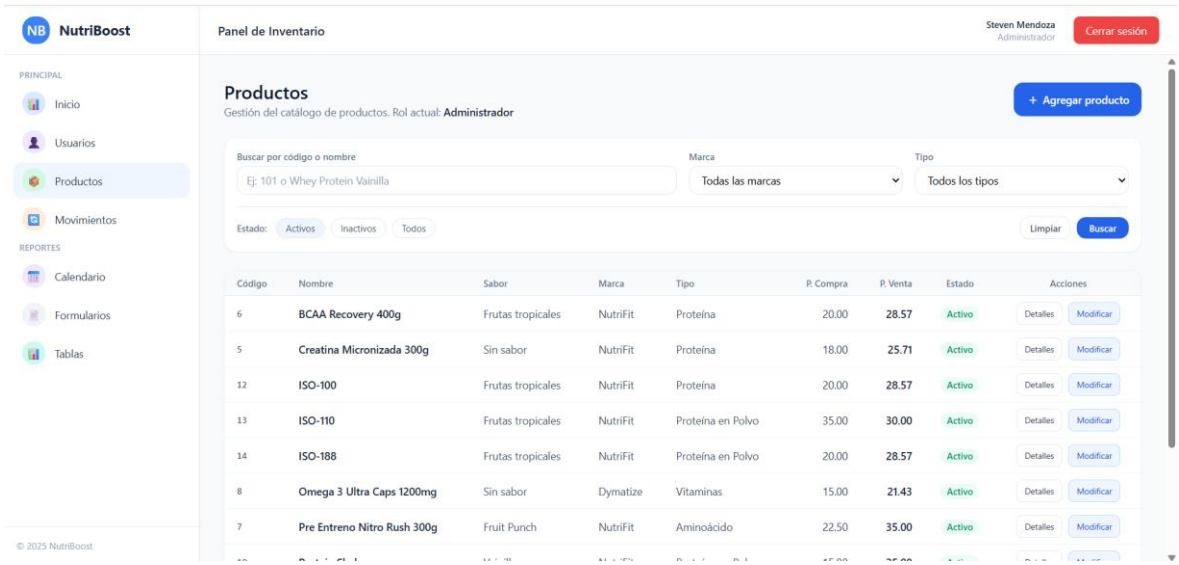
Pantalla de inicio de sesión del sistema



Panel principal de inventario y métricas del sistema



Panel de gestión de productos



6.3 Acta del Sprint 1

Acta de sprint review – Sprint 1

Proyecto: Prototipo de sistema web para la gestión de procesos críticos del control de inventarios – NutriBoost

Sprint: 1

Duración: 2 semanas

Fecha de revisión: 10-11-2025

Participantes:

Product Owner funcional: Daniela Yamileth Chávez Haro

Desarrollador / Scrum Master: Steven Alexis Mendoza Reza

1. Objetivo del Sprint

Implementar el backend del prototipo NutriBoost (base de datos, lógica de negocio y API REST), dejando una arquitectura estable para su posterior integración con la interfaz web.

2. Alcance del Sprint

Historias de usuario trabajadas:

HU1 – Autenticación (F0)

HU2 – Gestión de productos (F1)

HU3 – Edición de productos (F1)

HU10 – Gestión de usuarios (F7)

3. Resultados y evidencias

BD en SQL Server operativa (tablas, relaciones, vistas y triggers).

API REST en .NET 8 funcional para autenticación, productos y usuarios.

Endpoints probados con Swagger y Postman, verificando persistencia en la Base de Datos.

El Product Owner comprobó el correcto funcionamiento del backend y la coherencia de los cálculos de inventario.

Conclusión: El incremento del Sprint 1 se aprueba, con la observación de ampliar reportes en el siguiente Sprint.

4. Métricas del Sprint

Velocidad: 14 puntos.

Cumplimiento del objetivo: 100 %.

Cumplimiento de DoD: 100 % de PBIs comprometidos.

Retrabajo / incidencias: 1 ajuste menor.

Satisfacción del Product Owner: 5/5.

Firmas

Daniela Chávez Product Owner funcional	Steven Mendoza Desarrollador / Scrum Master

6.4 Acta del Sprint 2

Acta de sprint review – Sprint 2

Proyecto: Prototipo de sistema web para la gestión de procesos críticos del control de inventarios – NutriBoost

Sprint: 2

Duración: 2 semanas

Fecha de revisión: 05-12-2025

Participantes:

Product Owner funcional: Daniela Yamileth Chávez Haro

Desarrollador / Scrum Master: Steven Alexis Mendoza Reza

1. Objetivo del Sprint

Implementar la interfaz web en Angular e integrar el frontend con la API REST del backend, permitiendo gestionar productos, lotes, movimientos, caducidades y consulta operativa desde el navegador.

2. Alcance del Sprint

Historias de usuario trabajadas:

HU4 – Inventario / visualización de stock (F3)

HU5 – Movimientos de inventario (F3)

HU6 – Reportes de inventario (F5)

HU7 – Gestión de lotes (F2)

HU8 – Control de caducidades (F4)

HU9 – Consulta operativa (F6)

3. Resultados y evidencias

Interfaz web en Angular operativa para productos, lotes, movimientos, caducidades y consulta operativa.

Flujo extremo a extremo funcionando (frontend → API REST → base de datos SQL Server).

Reglas de negocio (stock, FIFO, caducidades, trazabilidad) visibles a través de la interfaz.

El Product Owner validó el uso del prototipo y confirmó que refleja los procesos reales de la tienda.

Conclusión: El incremento del Sprint 2 se aprueba y se considera que el prototipo cumple las funcionalidades críticas de control de inventarios.

4. Métricas del Sprint

Velocidad: 31 puntos (HU4=5, HU5=8, HU6=5, HU7=5, HU8=3, HU9=5).

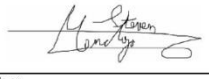
Cumplimiento del objetivo: 100 %.

Cumplimiento de DoD: 100 % de PBIs comprometidos.

Retrabajo / incidencias: 1 ajustes menores (mensaje de error).

Satisfacción del Product Owner: 5/5.

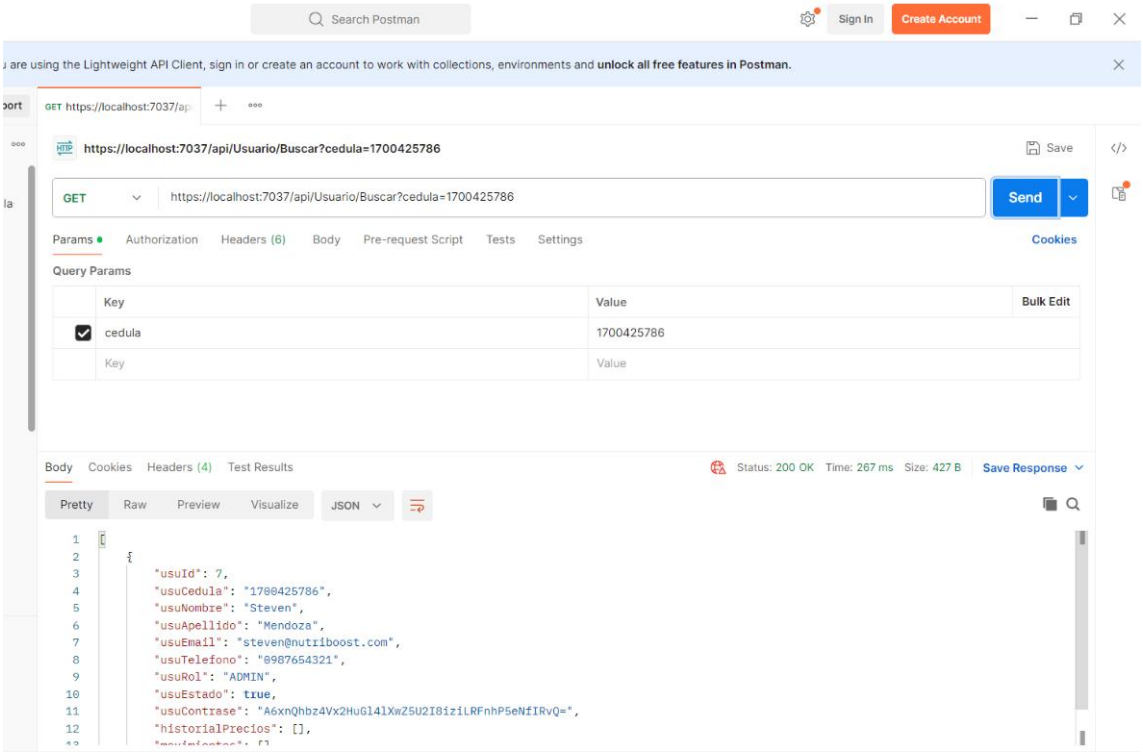
Firmas

	
Daniela Chávez Product Owner funcional	Steven Mendoza Desarrollador / Scrum Master

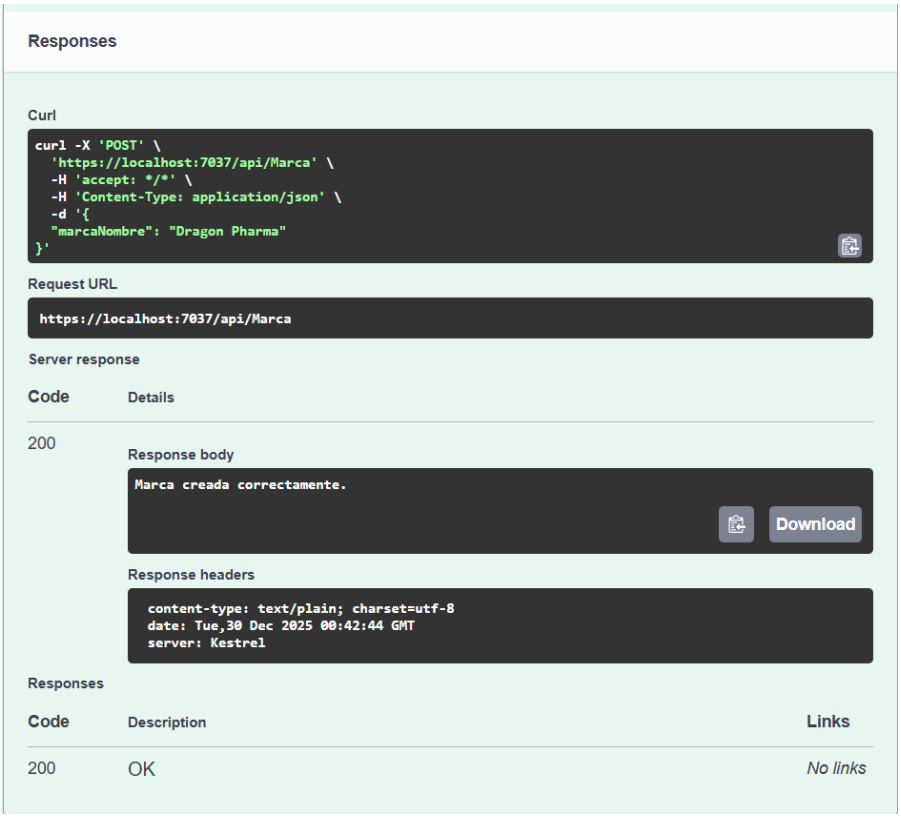
6.5 Pruebas unitarias

Las pruebas unitarias se realizaron para verificar que los endpoints del backend respondieran correctamente de forma individual, antes de la validación final con el cliente. Para este caso, se ejecutaron peticiones específicas sobre los módulos principales del sistema, comprobando la obtención de datos, la creación de registros y la respuesta del servidor en formato JSON; estas pruebas se apoyaron en Swagger UI y Postman, permitiendo confirmar que la API REST procesaba correctamente las solicitudes y devolvía resultados consistentes para el control de inventarios de NutriBoost.

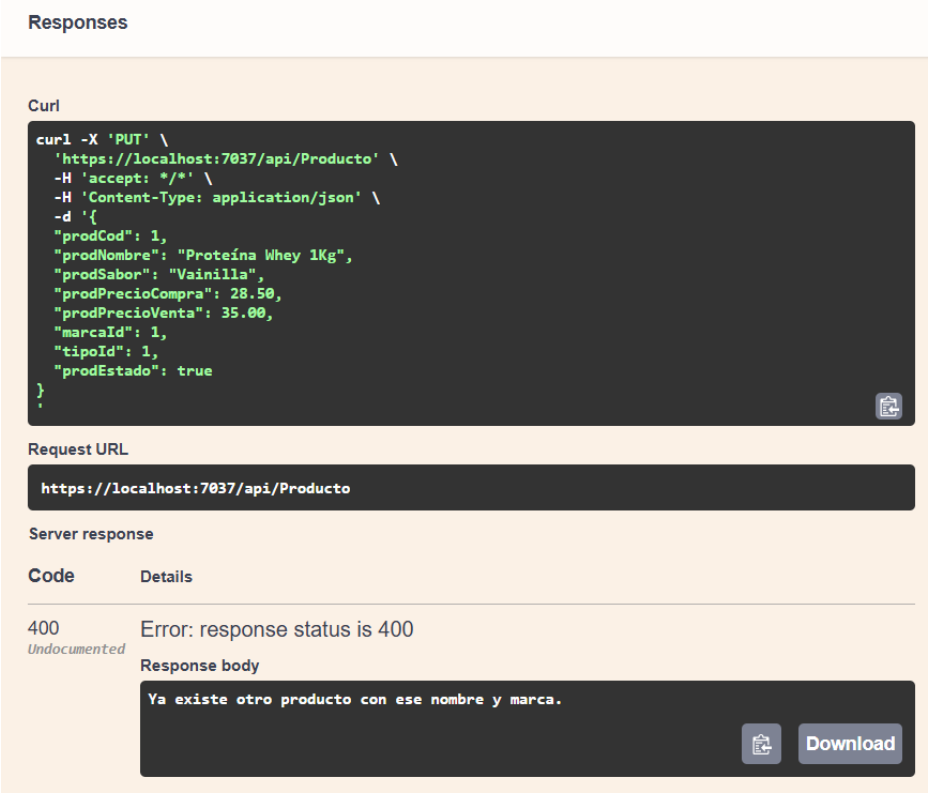
Postman – Buscar Usuario por cédula



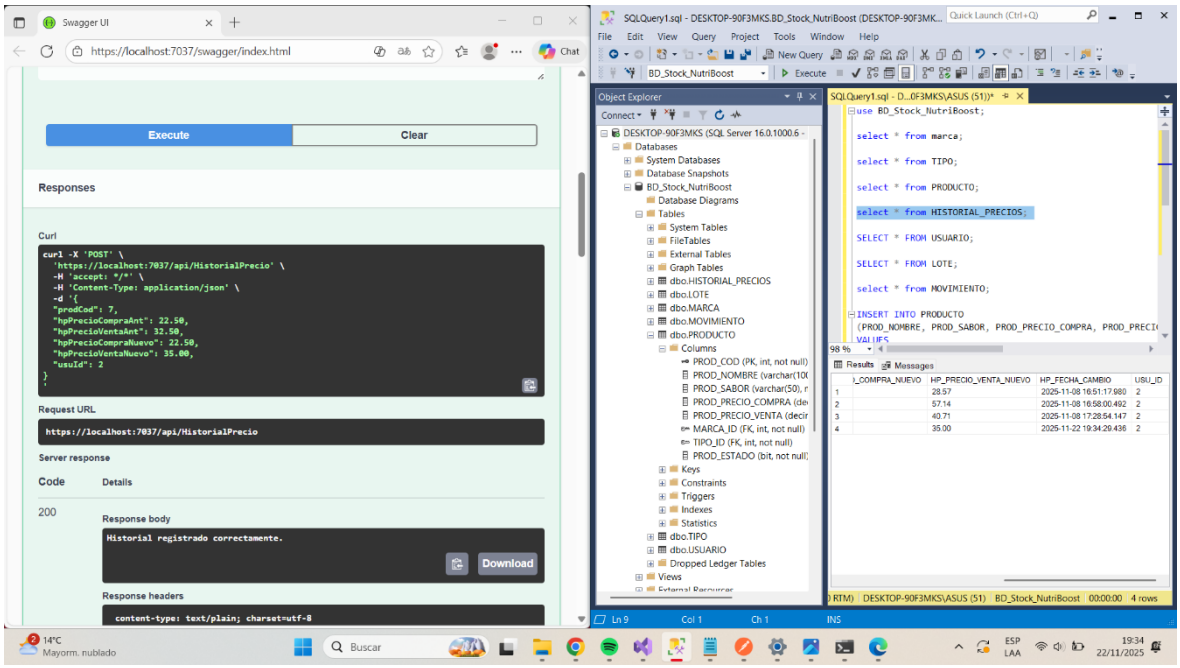
Swagger – POST Marca



Swagger – PUT Producto con validación de duplicidad



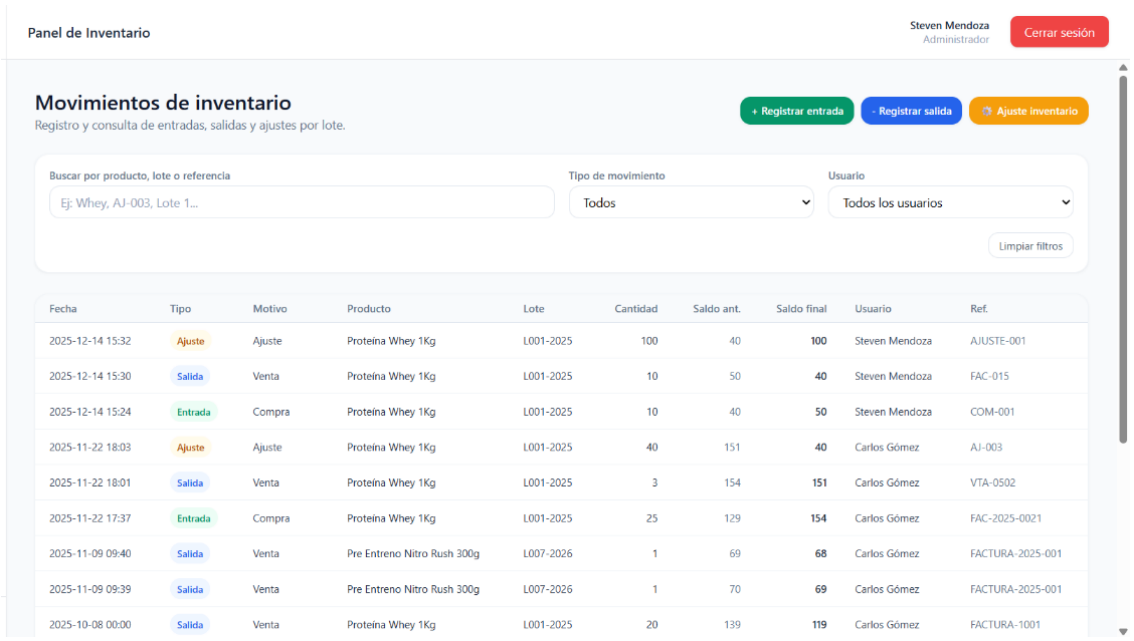
Swagger – POST HistorialPrecio y verificación en SQL Server



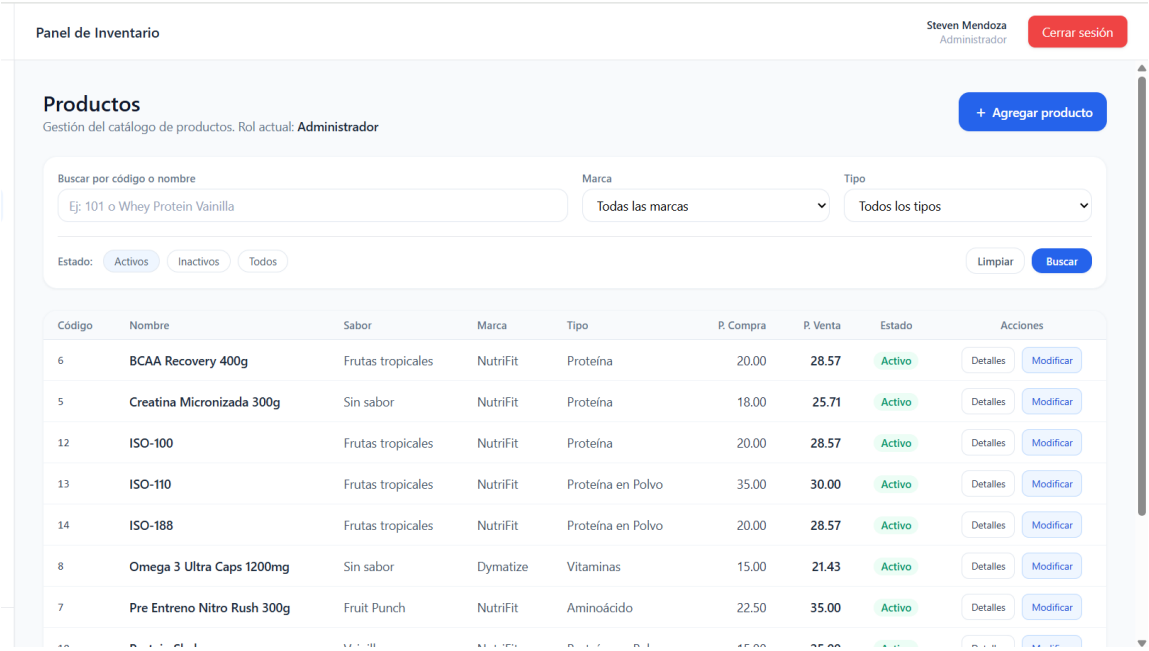
6.6 Pruebas de aceptación

Las pruebas de aceptación se realizaron con el objetivo de confirmar con el cliente que el prototipo cumplía con el flujo real de control de inventarios de NutriBoost. Durante esta validación se revisaron las funcionalidades principales desde la interfaz web, verificando el acceso al sistema con rol administrador, la gestión de los productos y con ello el registro/consulta de movimientos de inventario. Con base en los resultados observados, el cliente confirmó la conformidad del sistema para su uso como prototipo.

Módulo Movimientos de inventario



Módulo de Productos



Módulo Inventario de Productos