PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SISTEMA DE GESTIÓN DE INVENTARIOS EN QUALITYCORE SERVICES SAS

Proyecto de titulación previo a la obtención del título de: Tecnóloga/o Superior En Desarrollo de Software

Autores: Alison Lizbeth Carrion Hidalgo y Alexander Francisco Pavón Álvarez

Tutor: Patricio Omar Alvear Granda

Quito, Ecuador

2025

Dedicatoria

Alison Carrion:

Dedico este trabajo a las dos personas más importantes en mi vida, mis pilares, mis guías y mis mayores motivaciones.

A ustedes, que me enseñaron que no existen imposibles cuando se trabaja con esfuerzo, fe y perseverancia. Gracias por levantarme cada vez que caí, por secar mis lágrimas, por celebrar mis logros como si fueran suyos y por darme fuerzas cuando sentía que ya no podía más.

A mi mamá, por su amor infinito, por abrazarme incluso cuando no entendía mis miedos, por escucharme con paciencia y sin juzgarme, y por repetirme siempre que soy más fuerte de lo que creo.

A mi papá, por ser mi ejemplo de lucha, por enseñarme que el verdadero valor está en levantarse una y otra vez, por sus palabras que, aunque pocas, siempre dicen justo lo que necesito oír. Por creer en mí incluso cuando yo no lo hacía y por impulsarme a soñar sin miedo.

Sin ustedes, este logro simplemente no existiría. Todo lo que he alcanzado y lo que me proponga en el futuro siempre llevará su nombre, porque cada paso que doy está hecho con la fuerza y el amor que ustedes me dieron desde el principio.

A mi tía María Agusta, por estar siempre pendiente de mí, por su cariño y por esos consejos que llegan en el momento justo. Por motivarme a creer que puedo comerme el mundo si me lo propongo, y por recordarme que en cualquier lugar hay personas increíbles, sin importar de dónde vengan, y que no hay ninguna diferencia que nos separe como seres humanos.

A mi gran amigo Alex, que conocí en la universidad y que se convirtió en el mejor dúo para enfrentar cualquier reto. Hemos vivido risas, estrés, anécdotas épicas y muchos logros juntos, y me alegra haber encontrado en ti a alguien tan especial en este camino siempre te voy a llevar en mi corazón.

A mis amigos Alejandro y Michael, por cada momento compartido, por las conversaciones que empezaban como chistes y terminaban en reflexiones profundas. A Alejandro, especialmente, por motivarme tanto, por recordarme mi valor cuando yo misma dudaba, y por demostrarme que con apoyo sincero y fe en uno mismo se puede avanzar incluso en los días más pesados.

A Gabriel, que siempre ha estado presente para alentarme a seguir adelante y recordarme que nunca es tarde para cumplir lo que uno se propone.

Y a mis adoradas mascotas: Coco, Sofía y Wanda. No son solo mis compañeras, son parte de mi familia y de mi corazón. Han estado conmigo en incontables noches de desvelo, acompañándome mientras escribía, estudiaba o simplemente intentaba no rendirme. Con su compañía silenciosa, sus miradas llenas de ternura y esa forma única de dar amor sin pedir nada a cambio, me recordaron que incluso en los momentos más difíciles siempre hay algo que te reconforta y te da fuerzas para seguir.

Este logro es tanto mío como de cada uno de ustedes. Gracias por ser parte de mi historia y de este capítulo tan importante de mi vida.

Alexander Pavón:

Quiero dedicar este trabajo, en primer lugar, a mi familia.

A mis padres, por todo el apoyo incondicional que me han brindado y por el esfuerzo incansable que han hecho para que yo pueda seguir adelante. A mi madre, gracias por cada consejo, por comprenderme y por estar siempre a mi lado en los momentos más difíciles. A mi padre, por su dedicación y sacrificio, que han sido ejemplo e inspiración para mí.
A mi hermana, que ha sido mi ejemplo a seguir y mi inspiración durante toda mi vida, y a mi pequeño sobrinito, que con su ternura me ha motivado a continuar.

También quiero dedicar este logro a mi amiga, compañera de proyecto y mi dúo, Alison. Juntos recorrimos un camino lleno de aprendizajes, retos y satisfacciones, compartiendo momentos de risas, nervios y momentos icónicos. Desde el primer semestre se convirtió en alguien muy importante en mi vida, y gracias a su apoyo y compañía, esta etapa fue más llevadera y enriquecedora.

De igual forma, dedico este trabajo a dos grandes amigos de la universidad, Michael y Alejandro, por cada conversación, cada risa y cada muestra sincera de apoyo que hicieron de este camino una experiencia más especial.

No puedo olvidar a mis fieles compañeros, Pepito y Nala, que me acompañaron en incontables noches de tareas y en momentos de estrés, brindándome su compañía incondicional.

Finalmente, dedico este trabajo a una de las personas más importantes y valiosas en mi vida, Katherine. Quien ha seguido conmigo durante todo este tiempo a pesar de la distancia y aunque nuestros caminos se separaron siempre has estado presente de una forma u otra. Su amistad es una de las cosas más hermosas que me ha pasado y aunque la distancia o las

circunstancias nos separen, sé que siempre estaremos presentes en la vida del otro. Su apoyo ha sido fundamental para que llegara hasta aquí.

Contenido

Lista de tablas

Lista de figuras

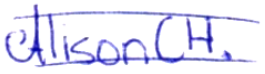
DECLARACIÓN y AUTORIZACIÓN

Yo, ALISON LIZBETH CARRION HIDALGO con C.I. 1750334839 autor(a) del trabajo de titulación intitulado: **"SISTEMA DE GESTIÓN DE INVENTARIOS EN QUALITYCORE SERVICES SAS"**, previa a la obtención del título de Tecnóloga Superior En Desarrollo De Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 11 de mayo de 2025



ALISON LIZBETH CARRION HIDALGO

C.I. 1750334839

DECLARACIÓN y AUTORIZACIÓN

Yo, **ALEXANDER FRANCISCO PAVÓN ÁLVAREZ** con C.I. 1752213023 autor(a) del trabajo de titulación intitulado: **"SISTEMA DE GESTIÓN DE INVENTARIOS EN QUALITYCORE SERVICES SAS"**, previa a la obtención del título de Tecnólogo Superior En Desarrollo De Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 11 de mayo de 2025

ALEXANDER FRANCISCO PÁVON ÁLVAREZ

C.I. 1752213023

Agradecimientos

Alison Carrion:

Agradezco profundamente a mis padres, por ser el cimiento sobre el que he construido cada uno de mis logros. Gracias por su amor que no conoce condiciones, por su apoyo que nunca ha flaqueado y por enseñarme, con el ejemplo, que la constancia y el esfuerzo abren cualquier puerta. A mi mamá, por su paciencia infinita, por saber escuchar incluso en mis silencios y por recordarme que no hay meta demasiado grande si se trabaja con el corazón. A mi papá, por su fortaleza, por mostrarme que los tropiezos son solo pausas en el camino y que levantarse siempre es la mejor respuesta.

A mi tía María Agusta, por su cariño sincero, por sus consejos que siempre llegan a tiempo y por motivarme a mirar el mundo con ambición y sin miedo, reconociendo el valor de cada persona sin importar sus diferencias.

A mi amigo y ahora compañero de tesis, Alex, porque en este camino hemos compartido de todo: risas, nervios, problemas que parecían imposibles, sustos que nos hicieron dudar y logros que nos llenaron de orgullo. Hemos fallado, hemos aprendido y, sobre todo, siempre lo hemos hecho juntos. Desde el principio hasta el final, trabajar a tu lado ha sido una experiencia que me enseñó que cuando hay confianza y apoyo mutuo, cualquier reto se puede superar. Gracias por estar en cada paso, por dar lo mejor de ti y por demostrarme que en equipo siempre llegamos más lejos.

A mis amigos Alejandro, y Michael, por su amistad, apoyo y por compartir tantos momentos que convirtieron este camino en una experiencia única. A Alejandro, especialmente, por ser una fuente constante de motivación y recordarme mi potencial incluso en los días más difíciles.

A Gabriel, por su presencia constante y por alentarme a seguir adelante en todo momento, mostrándome que siempre hay razones para luchar.

A mis docentes, por compartir su conocimiento con dedicación y pasión, contribuyendo a mi formación profesional y personal. De manera especial, agradezco al Ing. Patricio Alvear, mi tutor de tesis, por su guía comprometida, su paciencia inquebrantable y sus valiosas observaciones, que no solo enriquecieron este trabajo, sino que también fortalecieron mi capacidad para enfrentar retos con una visión más crítica y profesional.

A la Pontificia Universidad Católica del Ecuador, por brindarme las herramientas, el conocimiento y un espacio en el que pude crecer y descubrir mi verdadera vocación.

Finalmente, a mis adoradas mascotas: Coco, Sofía y Wanda. Gracias por acompañarme en tantas noches de desvelo, por su amor silencioso, por calmar mis días de estrés y por recordarme, con su lealtad y ternura, que siempre hay motivos para sonreír y seguir adelante.

A todos ustedes, gracias de corazón. Este logro es también suyo.

Alexander Pavón:

Quiero expresar mis más profundos y sinceros agradecimientos a todas las personas que, de una u otra manera, me han acompañado en este camino académico y personal. Cada palabra, gesto o apoyo recibido ha dejado una huella imborrable en mi vida y ha hecho posible que hoy pueda cumplir esta meta.

A mi familia, por ser mi mayor pilar y la razón más grande para seguir adelante. Su amor, comprensión y apoyo incondicional han sido la base sobre la cual he construido este logro. Gracias por creer en mí incluso en los momentos en que yo dudaba de mí mismo, y por enseñarme que el esfuerzo, la paciencia y la fe siempre dan fruto.

A mi compañera de tesis y amiga, Alison Carrión, por su compromiso, responsabilidad y constancia a lo largo de todo este proyecto. Trabajar a su lado no solo fue un verdadero placer, sino también una hermosa experiencia junto con todos los anteriores proyectos que trabajamos juntos donde llegamos a demostrar que somos un gran equipo demostrándome el valor del trabajo en equipo.

A mis amigos Alejandro y Michael, con quienes compartí risas, desvelos, conversaciones y experiencias que marcaron profundamente mi paso por la universidad. Gracias por estar ahí, por su apoyo sincero y por hacer que este viaje fuera más ameno y memorable.

A Katherine, por ser siempre mi refugio y mi compañía, incluso sin compartir directamente este entorno académico. Su presencia ha sido algo importante y valioso para mí, recordándome que no estoy solo. Sus palabras de aliento llegaron siempre en el momento justo, y de alguna u otra manera siempre me hizo sentir cerca, aunque estuviéramos lejos.

A mi profesor de titulación el Ing. Patricio Alvear, por su guía, paciencia y orientación constante durante el desarrollo de este trabajo. Su compromiso, dedicación y confianza en nuestras capacidades fueron clave para alcanzar los objetivos planteados.

A la Pontificia Universidad Católica del Ecuador, por brindarme una formación académica de calidad y ser el espacio en el que crecí no solo como profesional, sino también como persona.

Y finalmente, a todos los docentes que me acompañaron a lo largo de la carrera. Gracias por compartir sus conocimientos, por su vocación inspiradora y por sembrar en mí valores y enseñanzas que llevaré siempre conmigo.

Introducción

El presente proyecto tiene como objetivo desarrollar un Sistema de Gestión de Inventarios para la empresa QualityCore Services SAS, con el fin de mejorar el control de productos, optimizar procesos internos y brindar información precisa que facilite la toma de decisiones estratégicas. Actualmente, la empresa realiza el manejo de inventario de forma manual mediante hojas de cálculo, lo que conlleva a errores frecuentes, desorganización y falta de datos en tiempo real.

La solución propuesta consiste en una aplicación web moderna, compuesta por dos capas principales: Frontend y Backend. Para el desarrollo del frontend se utilizará React.js, una biblioteca de JavaScript que permite construir interfaces interactivas, dinámicas y reutilizables a través de componentes. React será acompañado de HTML5 para estructurar el contenido visual de la aplicación y CSS3 para el diseño y estilo personalizado de la interfaz, asegurando una experiencia de usuario atractiva y adaptable a diferentes dispositivos.

El backend será implementado con Django, un framework web de alto nivel basado en Python, elegido por su facilidad de desarrollo, estructura organizada y herramientas integradas como la gestión de superusuarios y control de roles. A través de Django se desarrollará toda la lógica del sistema, incluyendo la gestión de productos, entradas y salidas de inventario, y generación de reportes. La base de datos utilizada será PostgreSQL, reconocida por su alto rendimiento, seguridad y escalabilidad, lo cual permitirá manejar estructuras relacionales complejas de manera eficiente.

El proyecto se desarrollará utilizando Git y GitHub para el control de versiones, asegurando una colaboración fluida entre los integrantes del equipo y un historial completo de cambios. Para la planificación y seguimiento del desarrollo se aplicará la metodología ágil

SCRUM, utilizando Jira Software para organizar el trabajo por sprints, asignar tareas y monitorear el avance del sistema.

Con la implementación de este sistema, QualityCore Services SAS podrá automatizar su gestión de inventarios, reducir errores operativos, tener visibilidad en tiempo real de sus productos mediante una solución tecnológica moderna, eficiente y alineada a sus necesidades.

1. Capítulo I: Levantamiento de Requisitos y Diseño del Sistema

1.1 Requerimientos Funcionales

1.1.1 Funcionalidad N° 1: Inicio de sesión

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F1	Inicio de sesión	12/05/2025	19/05/2025	Permite a los usuarios ingresar al sistema con credenciales válidas, verificando sus datos contra la base de datos para acceder a las funciones según su rol (administrador o usuario). Garantiza la seguridad del acceso y controla los permisos de uso.	Alta
	Entrada	**/Proceso**		**Salida**	
	Campo de usuario	Validación de credenciales en la base de datos		Si existe, pasa a la siguiente validación.	
	Campo de Contraseña	Comparación de la contraseña ingresada con la almacenada (encriptada).		Si coincide, se permite el ingreso.	
	Botón de inicio de sesión	El sistema envía la solicitud de autenticación y consulta la tabla de usuarios.		Respuesta: acceso al sistema o mensaje de error.	

Tabla 1: Inicio de Sesión

1.1.2 Funcionalidad N° 2: Registro y Gestión de Usuarios

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F2	Registro y gestión de usuarios	20/05/2025	28/05/2025	Permite crear nuevas cuentas de usuario, editar los roles e inactivar las que ya no se usan. Se asignan roles (administrador o usuario) para controlar los permisos de acceso.	Alta
	Entrada	**/Proceso**		**Salida**	
	Formulario de registro con campos: nombre, correo electrónico, teléfono, contraseña, rol	Validación de campos obligatorios: nombre no vacío, correo válido, contraseña.		Si los datos son correctos, se continúa con el registro.	
	Botón de guardar usuario	Encriptación de la contraseña y almacenamiento de datos en la base de datos.		Usuario registrado correctamente.	
	Botón de editar rol	Permite modificar el rol y actualiza en la base de datos.		Rol actualizado con éxito.	
	Botón de inactivar o activar usuario	Inactiva al usuario el cual no puede volver a acceder al sistema a menos que el administrados le active de nuevo.		Usuario inactivado del sistema.	

Tabla 2: Registro y Gestión de Usuarios

1.1.3 Funcionalidad N° 3: Mi perfil

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F3	Visualización del perfil de usuario	29/05/2025	06/06/2025	Permite visualizar el perfil del usuario autenticado, con sus datos personales y credenciales. Incluye botones para editar el perfil y cambiar la contraseña.	Alta
	Entrada	**/Proceso**		**Salida**	
	Datos del usuario logueado	Consulta en la base de datos del usuario autenticado		Perfil del usuario mostrado en pantalla	
	Botón “Editar Perfil”	Carga del formulario con los datos personales del usuario		Cambios reflejados en el sistema	
	Botón “Cambiar Contraseña”	Habilitación de formulario para ingresar nueva contraseña		Cambio exitoso dentro del sistema	

Tabla 3: Perfil de usuario

1.1.4 Funcionalidad N° 4: Proveedor

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F4	Registro, edición y eliminación de proveedores	07/06/2025	14/06/2025	Permite registrar, editar y eliminar proveedores con datos como nombre, RUC, contacto, dirección, etc.	Alta
	Entrada	**/Proceso**		**Salida**	
	Formulario con: nombre, RUC, contacto, dirección	Validación de datos, inserción, edición o eliminación en la base de datos		Proveedor registrado, actualizado o eliminado correctamente	
	Filtro de búsqueda (por	Búsqueda y filtrado de proveedores		Lista filtrada de proveedores según criterios	

	nombre, RUC, estado)		
	Botón de "Guardar", "Editar" o "Eliminar proveedor"	Acciones de gestión sobre proveedores en la base de datos	Cambios reflejados en el sistema

Tabla 4: Proveedor

1.1.5 Funcionalidad N° 5: Registro de Clientes

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F5	Registro, edición y eliminación de clientes	15/06/2025	23/06/2025	Permite registrar clientes con información como nombre, cédula/RUC, teléfono, dirección. Editar y eliminar.	Alta
	Entrada	**/Proceso**		**Salida**	
	Formulario con: nombre, cédula/RUC, teléfono, dirección	Validación de datos, inserción, edición o eliminación en la base de datos		Cliente registrado, actualizado o eliminado correctamente	
	Filtro de búsqueda (por nombre, cédula/RUC)	Búsqueda y filtrado de clientes		Lista filtrada de clientes según criterios	
	Botón de "Guardar", "Editar" o "Eliminar cliente"	Acciones de gestión sobre clientes en la base de datos		Cambios reflejados en el sistema	

Tabla 5: Registro Clientes

1.1.6 Funcionalidad N° 6: Cotización

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F6	Generación de cotizaciones	24/06/2025	05/07/2025	Permite generar cotizaciones personalizadas de productos seleccionados del inventario. Exportar en PDF para su envío al cliente correspondiente.	Alta
	Entrada	**/Proceso**		**Salida**	
	Selección de productos del inventario	Selección de productos, cálculo de precios, subtotal, total		Cotización generada correctamente en PDF.	
	Datos del cliente	Asociación de la cotización al cliente seleccionado		Cotización vinculada al cliente.	
	Botón "Generar cotización"	Generación del archivo PDF.		Documento exportado.	

Tabla 6: Generación de Cotizaciones

1.1.7 Funcionalidad N° 7: Registro de productos

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F7	Registro de productos	06/07/2025	15/07/2025	Permite registrar nuevos productos en el inventario, ingresando información como nombre, proveedor, precio y stock minimo. También permite editar y eliminar productos existentes.	Alta
	Entrada	**/Proceso**		**Salida**	
	Formulario con nombre del producto, categoría, precio, stock mínimo y proveedor	Validación de campos: no vacíos, precios positivos, stock numérico.		Si todo es válido, se habilita el botón de guardar.	

	Botón “Guardar producto”	Inserta los datos en la base de datos.	Producto registrado correctamente y visible en el listado.
	Botón “Editar producto”	Recupera la información del producto, permite modificar y actualiza en BD.	Cambios guardados exitosamente.
	Botón “Eliminar producto”	Confirma con el usuario y borra el producto seleccionado.	Producto eliminado del inventario.

Tabla 7: Registro de Productos

1.1.8 Funcionalidad N° 8: Movimientos

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F8	Movimientos	16/07/2025	23/07/2025	Permite registrar las entradas y salidas de productos, actualizando el inventario y generando comprobantes.	Alta
	Entrada	**Proceso**		**Salida**	
	Añadir entrada	Actualiza el stock al registrar la entrada.		Entrada registrada correctamente.	
	Añadir Salida	Descuenta el stock al registrar la salida.		Salida registrada correctamente.	
	Buscar movimientos	Consulta entradas y salidas en el sistema.		Lista de movimientos mostrada en pantalla.	

Tabla 8: Registro de Ventas

1.1.9 Funcionalidad N° 9: Generación de Reportes

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F9	Generación de reportes	24/07/2025	02/08/2025	El sistema permite generar reportes personalizados de inventario, ventas, productos más vendidos con un filtro de fechas.	Media
	Entrada	**Proceso**		**Salida**	
	Selección del tipo de reporte (productos o ventas)	Consulta en la base de datos con filtros aplicados.		Datos obtenidos y formateados en tabla.	
	Filtro por fechas	Agrupación, conteo o suma de datos según tipo de reporte.		Reporte con totales	
	Botón “Generar reporte”	Se construye el archivo en formato PDF.		Archivo descargable	

Tabla 9: Generación de Reportes

1.1.10 Funcionalidad N° 10: Alertas por bajo stock

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F10	Alertas por bajo stock	03/08/2025	15/08/2025	Genera notificaciones cuando el stock de un producto está por debajo del nivel mínimo definido por el administrador.	Alta
	Entrada	**Proceso**		**Salida**	
	Stock actual y stock mínimo configurado	Comparación automática por el sistema (cada vez que se ingresa o resta producto).		Si el stock menor o igual al mínimo, se genera una alerta visual en la lista.	

	Usuario accede al inventario	El sistema busca productos con alertas activadas.	Visualización de advertencias en una sección especifica de la pantalla.
	Botón “Ocultar”	El usuario hace clic en el botón “Ocultar” ubicado debajo de la alerta visual.	La alerta desaparece de la sección de alertas, pero se mantiene registrada internamente.

Tabla 10: Alertas por Bajo Stock

1.2 Diseño Visual

1.2.1 Mockups

Ilustración 1: Pantalla de Inicio de Sesión

Pantalla de inicio de Sesión. Permite a los usuarios ingresar al sistema mediante correo y contraseña con la opción de recuperación de contraseña.

Ilustración 2: Pantalla de Mi Perfil

Pantalla de Mi Perfil. Permite al usuario consultar su información personal como nombre, correo electrónico, teléfono y rol. Además, ofrece las opciones de editar el perfil y cambiar la contraseña.

Ilustración 3: Pantalla Principal

Pantalla Principal. Muestra un resumen general del sistema con tarjetas de información clave (productos, ventas, movimientos, etc.) y alertas de stock. A la izquierda se encuentra el menú con acceso a las funciones principales como inventario, movimientos, reportes, clientes y usuarios.

Ilustración 4: Pantalla de Productos

Pantalla de Productos. Permite registrar, editar y eliminar los productos del inventario y se visualizan en forma de listado.

Ilustración 5: Pantalla de Movimientos

Pantalla de Movimientos. Gestiona los movimientos de inventario, registrando de forma separada las entradas y salidas para un historial claro y ordenado.

Ilustración 6: Pantalla de Reportes

Pantalla de Reportes. Permite seleccionar el tipo de reporte (movimientos recientes o productos más vendidos) y un rango de fechas para generar un informe en formato PDF.

Ilustración 7: Pantalla de Proveedores

Pantalla de Proveedores. Permite registrar nuevos proveedores y gestionar la información de los existentes, como nombre, correo, teléfono y dirección. Incluye las opciones de editar o eliminar un proveedor.

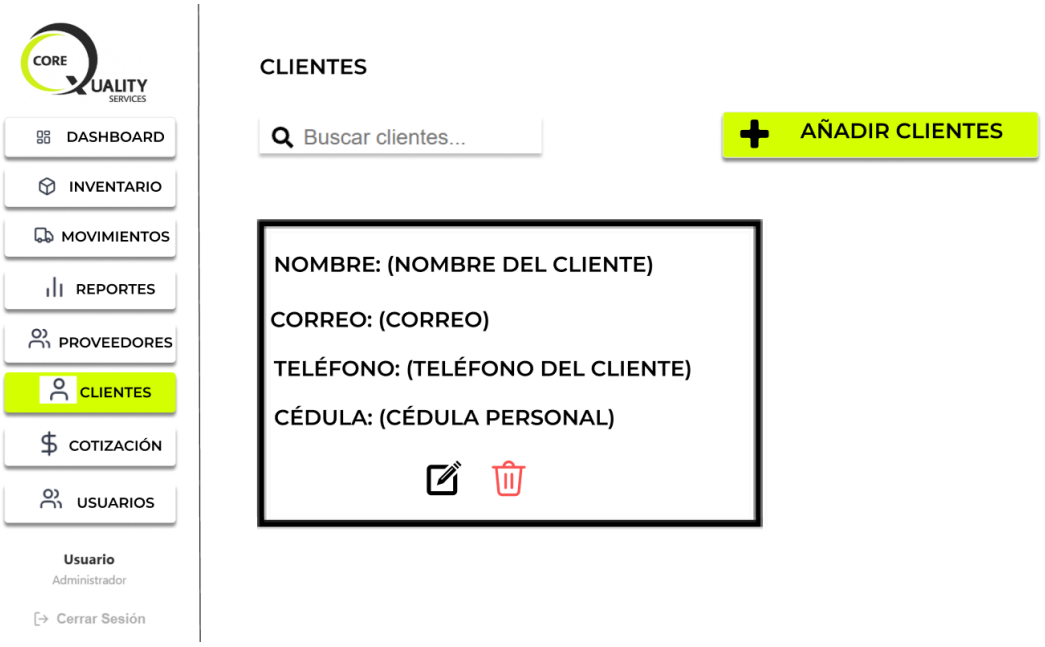


Ilustración 8: Pantalla de Clientes

Pantalla de Clientes. Permite registrar nuevos clientes y gestionar la información de los existentes, como nombre, correo, teléfono y cédula. Incluye las opciones de editar o eliminar un cliente.

Ilustración 9: Pantalla de Cotización

Pantalla de Cotización. Permite generar cotizaciones para clientes, seleccionando productos, cantidades y precios. Calcula automáticamente el subtotal, IVA y total, y ofrece la opción de guardar y generar el archivo PDF.

Ilustración 10: Pantalla de Usuarios

Pantalla de Usuarios. Permite añadir nuevos usuarios con rol de administrador o usuario, y gestionar su acceso activándolos o inactivándolos según sea necesario y enlista a todos los usuarios creados.

1.3 Diagramas de Casos de Uso

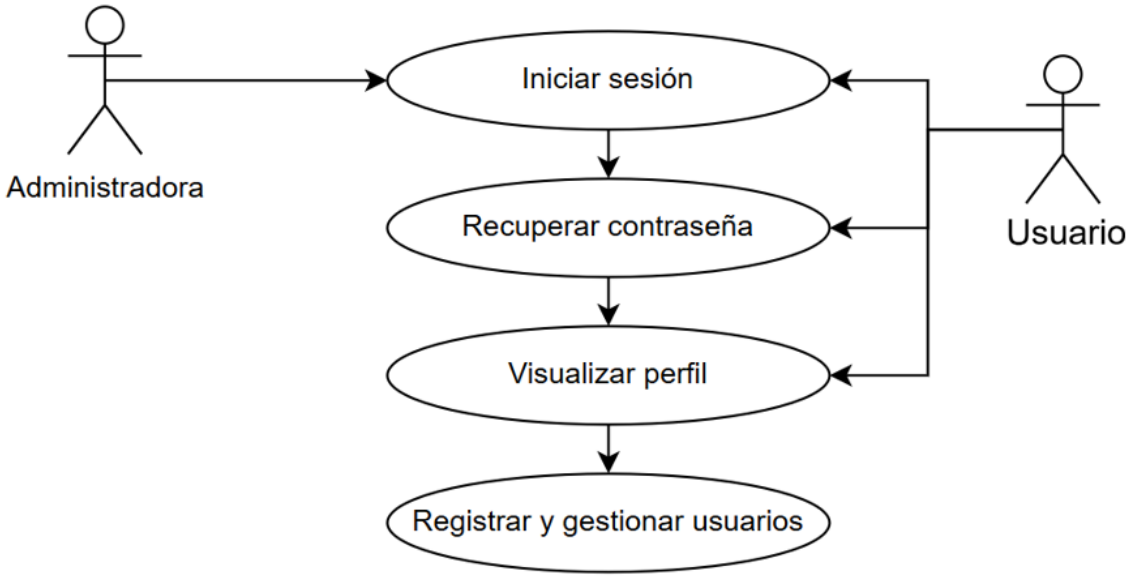


Ilustración 11: Diagrama de Casos de Uso - Gestión de Usuarios

Este diagrama muestra las interacciones entre los actores (Administradora y Usuario) y el sistema para realizar funciones relacionadas con la autenticación de usuarios y la administración de cuentas. Incluye actividades como iniciar sesión, recuperar contraseña, visualizar perfil y registrar o gestionar usuarios. El diagrama define claramente los permisos y acciones que puede realizar cada actor según su rol.

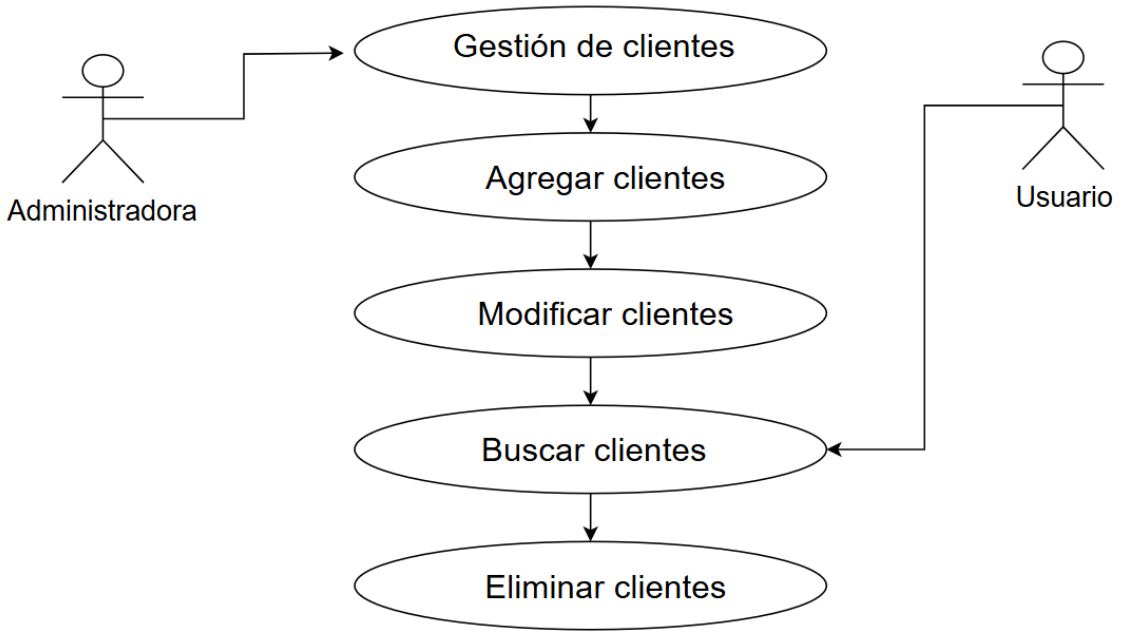


Ilustración 12: Diagrama de Casos de Uso - Gestión de Clientes

Este diagrama muestra las interacciones entre los actores (Administradora y Usuario) y el sistema para gestionar la información de los clientes. Representa las funcionalidades clave como agregar, modificar, buscar y eliminar clientes, así como la visualización general de la lista de clientes registrados. El diagrama detalla las acciones disponibles para cada usuario, estableciendo un control claro sobre la gestión de datos de clientes, lo que permite mantener la información actualizada y precisa dentro del sistema.

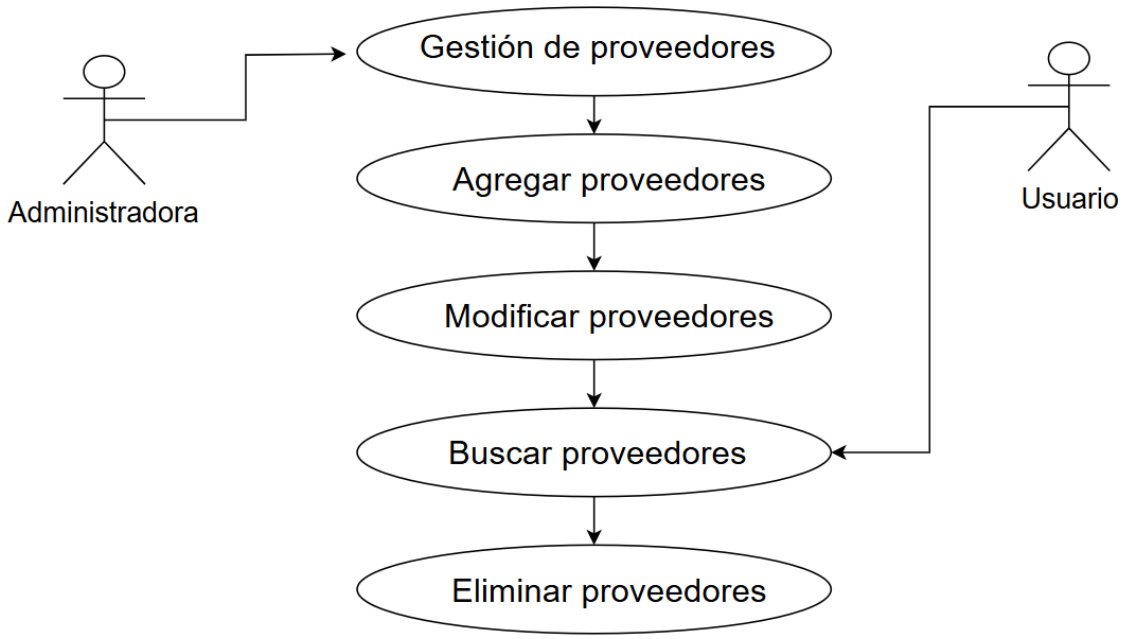


Ilustración 13: Diagrama de Casos de Uso - Gestión de Proveedores

Este diagrama muestra las interacciones entre los actores (Administradora y Usuario) y el sistema para administrar proveedores. Permite agregar, modificar, buscar y eliminar proveedores, asegurando una correcta gestión de la información de contacto y datos comerciales necesarios para el control de inventario.

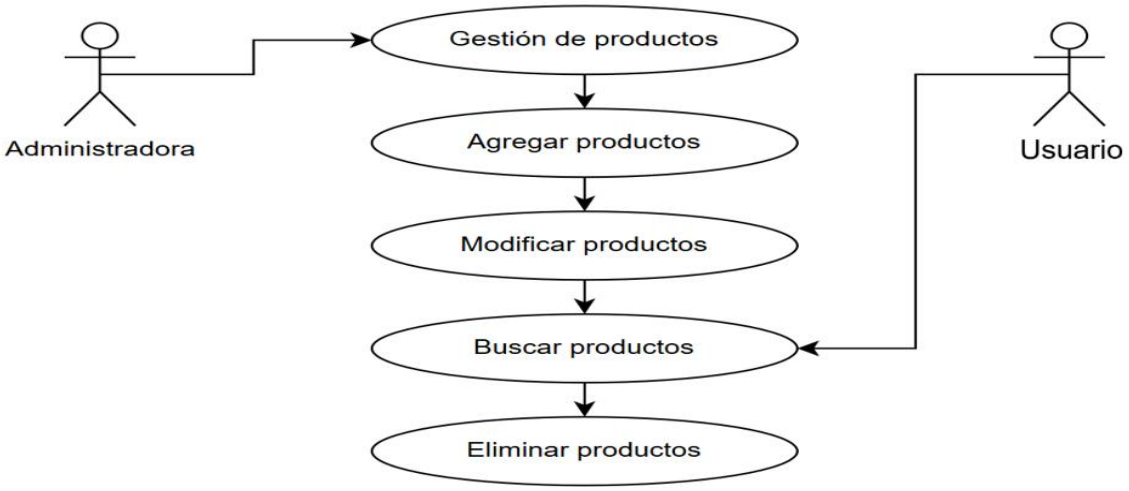


Ilustración 14: Diagrama de Casos de Uso - Gestión de Productos

Este diagrama representa las interacciones entre los actores (Administradora y Usuario) y el sistema para gestionar productos. Permite agregar, modificar, buscar y eliminar productos en el inventario, asegurando que la información de cada artículo esté actualizada y disponible para las operaciones del sistema.

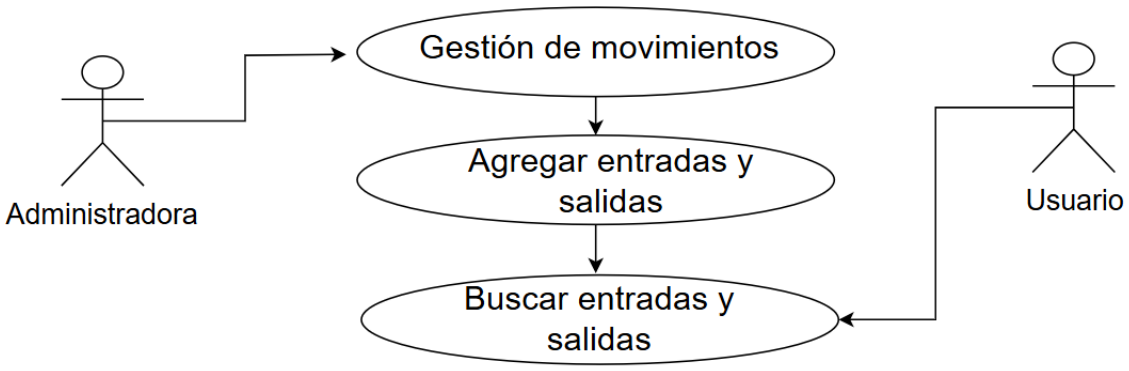


Ilustración 15: Diagrama de Casos de Uso - Gestión de Movimientos

Este diagrama muestra las interacciones entre los actores (Administradora y Usuario) y el sistema para gestionar las operaciones de movimientos. Incluye funcionalidades como agregar y buscar entradas y salidas de productos, lo que permite mantener un control actualizado del inventario y registrar correctamente las transacciones comerciales.

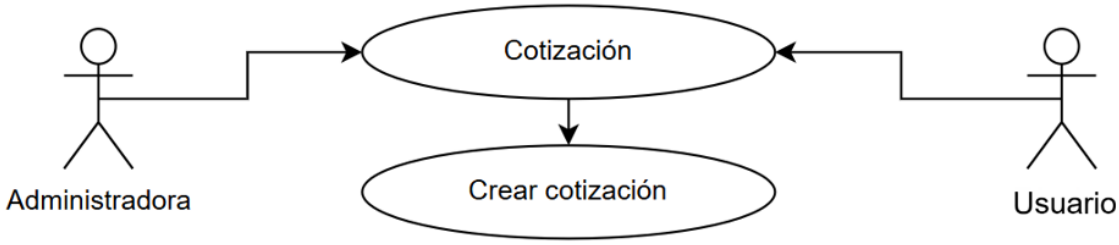


Ilustración 16: Diagrama de Casos de Uso - Cotización

Este diagrama muestra las interacciones entre los actores (Administradora y Usuario) y el sistema para la creación de cotizaciones. Permite generar documentos con los detalles de productos seleccionados, precios, cantidades y datos del cliente, facilitando el proceso de emisión de cotizaciones para los usuarios.

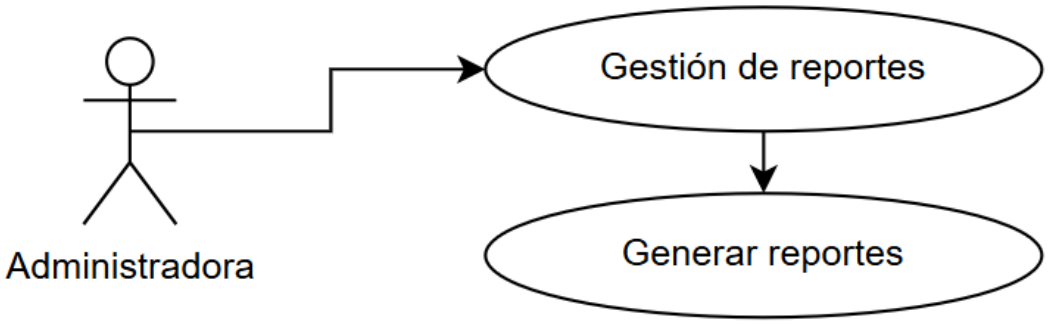


Ilustración 17: Diagrama de Casos de Uso - Gestión de Reportes

Este diagrama muestra la interacción entre la Administradora y el sistema para generar reportes personalizados. Permite obtener informes detallados sobre productos, ventas y alertas, facilitando el análisis y la toma de decisiones estratégicas para la empresa.

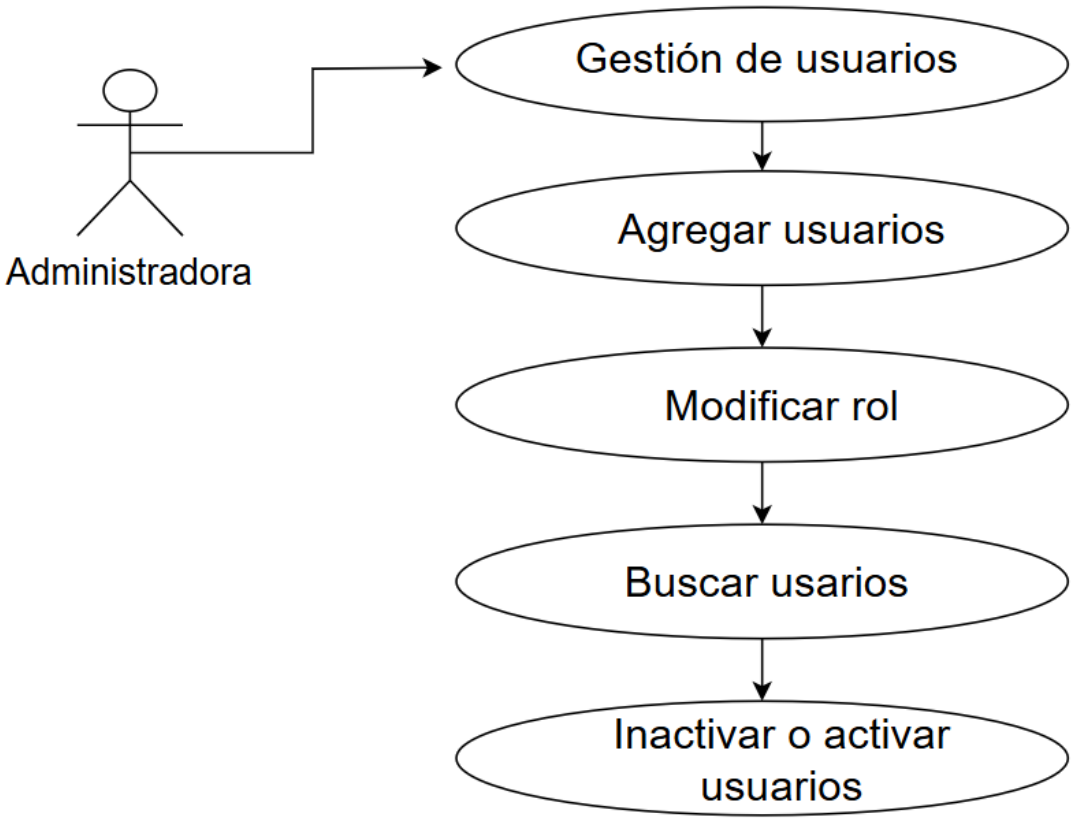


Ilustración 18: Diagrama de Casos de Uso - Gestión de Usuarios

Este diagrama representa las interacciones entre la administradora y el sistema para la gestión de usuarios. Permite realizar acciones como agregar nuevos usuarios, modificar sus roles, buscarlos en el sistema e inactivarlos o activarlos según sea necesario. Estas funciones aseguran un control adecuado del acceso al sistema y la asignación de permisos según el rol de cada usuario.

1.4 Diagrama de Clases

El siguiente diagrama de clases muestra la estructura principal del sistema de gestión de inventarios. Representa las clases principales del sistema, sus atributos, métodos y las relaciones entre ellas. Se incluyen entidades como Usuario, Producto, Cliente, Proveedor, Cotización, Movimiento, ProductoCotizado y Alerta, junto con sus respectivos detalles y cardinalidades.

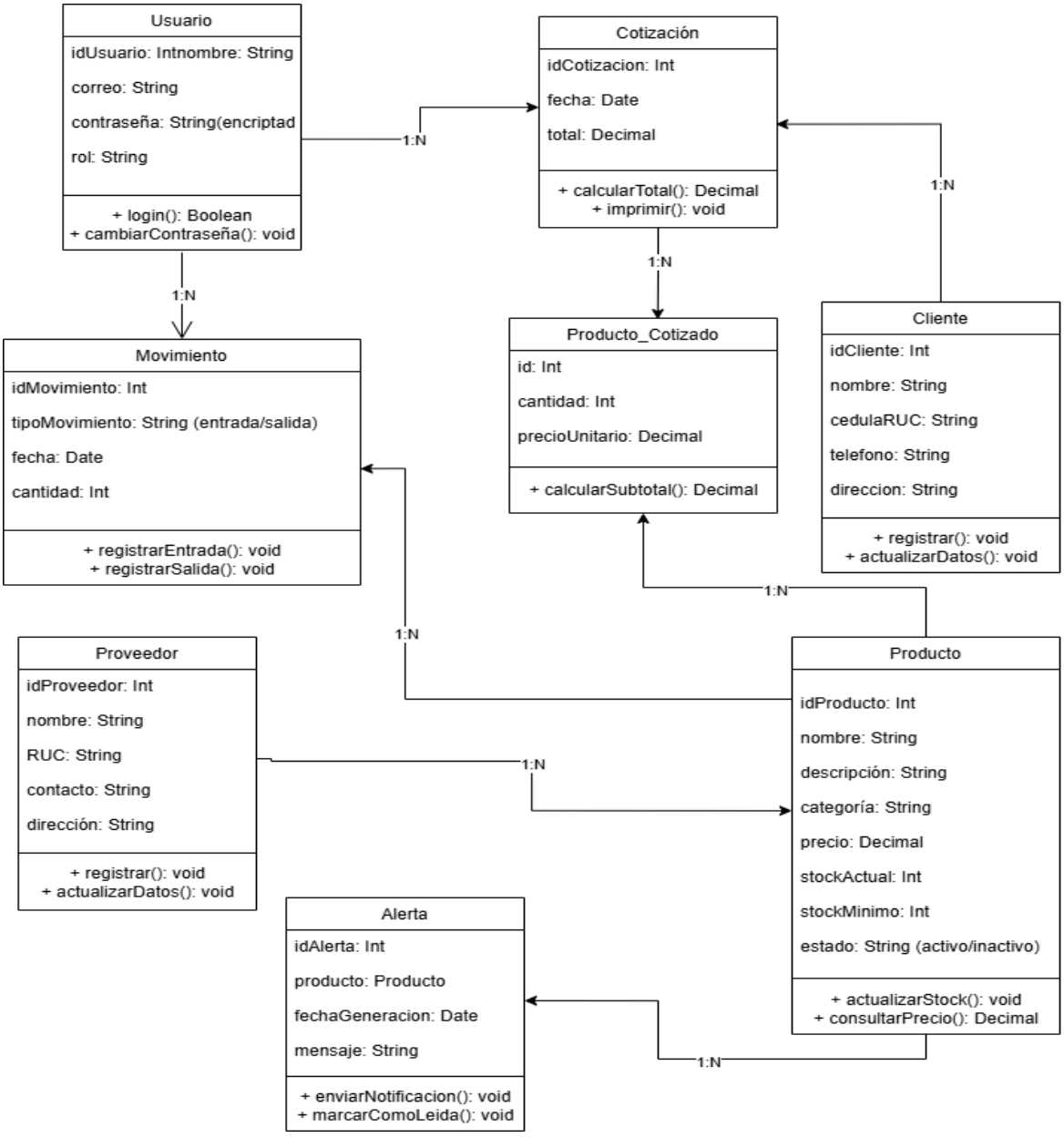


Ilustración 19: Diagrama de Clases

1.5 Diagrama Entidad Relación

Este diagrama Entidad-Relación representa las entidades principales, sus atributos y las relaciones de la base de datos para el sistema de gestión de inventarios.

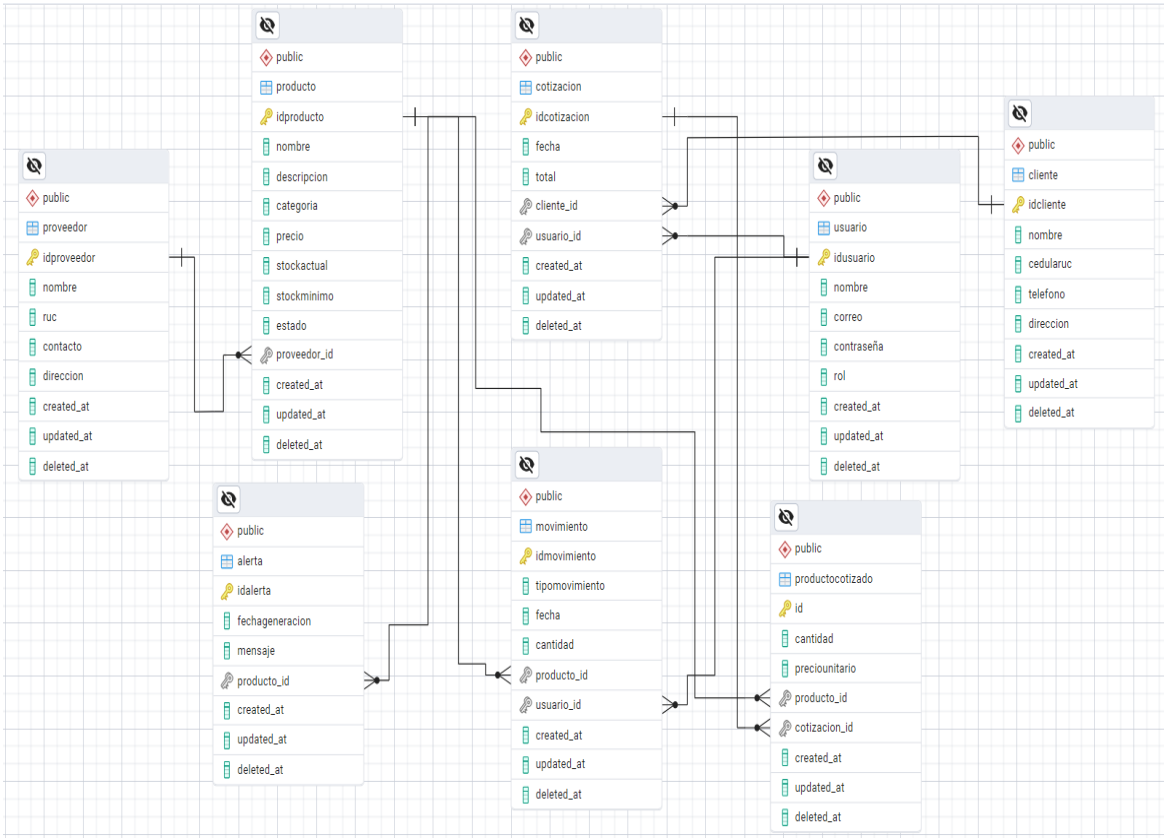


Ilustración 20: Diagrama Entidad Relación

2. Capítulo II

2.1 Construcción del Sistema Frontend

2.1.1 Estándares de construcción

Para el desarrollo del frontend del sistema se utilizó React.js, un framework moderno de JavaScript basado en componentes. Se mantuvieron las siguientes convenciones de desarrollo:

- **Nombres de carpetas**: snake_case como por ejemplo components, pages, services, styles para facilitar su lectura y consistencia.
- **Nombres de clases**: componentes en CamelCase como por ejemplo DashboardPage, UserProfile, AddUserForm, siguiendo las buenas prácticas de React.
- **Archivos JavaScript**: terminan en .js, mientras que los archivos de estilo CSS están separados en la carpeta styles/ para que cada página tenga su propio archivo de diseño.
- Se utilizó la convención de una estructura por capas: components/ para elementos reutilizables, pages/ para cada pantalla, services/ para conexión a API, y assets/ para recursos estáticos como imágenes, en este caso el logo de la empresa QualityCore Service.

2.1.2 Definición de paginas

El sistema cuenta con las siguientes páginas, organizadas dentro de la carpeta pages/, estas son todo lo principal que va a ver el usuario cuando entre al sistema:

2.1.2.1 LoginPage:

La pantalla de inicio de sesión es la puerta de entrada al sistema y permite a los usuarios autenticarse mediante su correo y contraseña junto con una opción para recuperar la contraseña en caso de olvido.

Funcionalidad principal

- **Inicio de sesión**: Muestra un formulario principal (LoginForm) que solicita las credenciales del usuario (correo y contraseña). Si los datos son correctos, se redirige automáticamente al Dashboard.
- **Recuperación de contraseña**: Si el usuario olvida su contraseña, puede hacer clic en el enlace "¿Olvidaste tu contraseña?" para cambiar al formulario de recuperación (ForgotPasswordForm). Allí se solicita un correo para restablecer su acceso.
- **Seguridad**: Al cargar la página, se hace una petición GET al endpoint /login/ para obtener el token CSRF requerido por Django.

Componentes y archivos usados

- LoginForm.js: contiene el formulario de inicio de sesión, validación de campos, envío de datos y manejo de errores.
- ForgotPasswordForm.js: permite enviar un correo de recuperación de contraseña al usuario.
- api.js: define la constante API_URL para conectarse al backend.
- LoginPage.css: hoja de estilos asociada que aplica diseño centrado, tarjetas, imágenes y estilos para mensajes y enlaces.

Diseño visual

- La interfaz se compone de una **tarjeta central** que incluye:
    - El logo de la aplicación.
    - El título del sistema.
    - El formulario correspondiente (login o recuperación).
    - Un enlace que permite alternar entre ambos formularios.
    - Un mensaje visual que muestra confirmaciones o errores.

2.1.2.2 ResetPassword:

La pantalla de restablecimiento de contraseña permite a los usuarios establecer una nueva contraseña luego de haberla solicitado. Esta vista se accede mediante un enlace enviado por correo electrónico, el cual contiene los parámetros de validación necesarios (uid y token).

Funcionalidad principal

- **Validación del enlace**:
    - Se extraen los parámetros uid y token desde la URL utilizando useSearchParams().
    - Si alguno de ellos no está presente (por ejemplo, si el enlace ha expirado), se muestra un mensaje indicando que el enlace no es válido.
- **Formulario de nueva contraseña**:
    - Se solicitan dos campos:

        - Nueva contraseña
        - Confirmación de contraseña
    - Ambos campos se validan localmente para asegurar:
        - Mínimo 6 caracteres
        - Coincidencia entre ambos valores

- **Envío de la nueva contraseña**:
    - Al enviar el formulario, se realiza una solicitud POST al backend (/reset-password/) con el uid, el token y la nueva contraseña.
    - Si la operación es exitosa, se muestra un mensaje de confirmación y el usuario es redirigido automáticamente a la página de login luego de 2 segundos.
    - Si ocurre un error, se muestra un mensaje con el detalle.

Componentes y archivos usados

- lucide-react: se utilizan los íconos Lock para los campos de contraseña y CheckCircle2 para mostrar confirmación visual.
- ResetPassword.css: hoja de estilos que define el diseño del formulario, inputs, botón y mensajes.
- logo.png: imagen de marca del sistema que aparece en la parte superior del formulario.

Diseño visual

- La interfaz está centrada y diseñada como una tarjeta clara y amigable.
- Cada input está acompañado de un ícono representativo.
- Los mensajes se muestran debajo del botón y cambian de color dependiendo del resultado.
- Si la contraseña se cambia correctamente, el sistema muestra un ícono de confirmación y un mensaje en verde.

2.1.2.3 ProfilePage:

La página de perfil permite al usuario visualizar su información personal y realizar acciones relacionadas con su cuenta, como editar sus datos o cambiar su contraseña. Es una vista dedicada al usuario autenticado.

Funcionalidad principal

- **Visualización del perfil**: Se muestra un resumen de los datos del usuario actual (nombre, correo, rol, etc.) a través del componente UserProfile.
- **Edición del perfil**: Al presionar el botón de editar, se abre un modal con el formulario EditProfileForm, que permite modificar los datos personales del usuario.
- **Cambio de contraseña**: El botón de "Cambiar contraseña" abre otro modal con el componente ChangePasswordForm, donde el usuario puede ingresar su contraseña actual y definir una nueva.

Cuando el usuario guarda su perfil hay una sincronización de datos de otras partes de la aplicación que dependan del usuario autenticado y su información.

Componentes y archivos usados

- UserProfile.js: componente principal que muestra los datos del usuario junto a botones de acciones (editar, cambiar contraseña).
- EditProfileForm.js: formulario modal que permite modificar los datos personales.
- ChangePasswordForm.js: formulario para actualizar la contraseña del usuario.
- Modal.js: componente genérico de ventana emergente para encapsular formularios.
- ProfilePage.css: archivo de estilos exclusivo para esta vista.

Diseño visual

- El perfil se muestra en un formato limpio y centrado.
- Los formularios emergentes aparecen en modales con diseño coherente al resto del sistema.
- Se asegura una experiencia de usuario fluida y clara, manteniendo el foco en las acciones personales del usuario.

2.1.2.4 DashboardPage:

La página principal del sistema, denominada **Dashboard**, tiene como objetivo ofrecer una vista general del estado actual del inventario, movimientos, clientes, ventas y alertas de bajo stock. Esta pantalla es la primera que visualiza el usuario después de iniciar sesión.

Funcionalidad principal

- **Resumen general del sistema**: mediante una consulta a la ruta /dashboard/summary/, se obtienen estadísticas claves como:
    - Total de productos
    - Total de clientes registrados
    - Total de movimientos
    - Entradas y salidas
    - Ventas totales
    - Número de alertas por bajo inventario

Estos datos se representan en tarjetas con íconos e indicadores visuales, utilizando la librería lucide-react para íconos.

- **Visualización de alertas**: las alertas de bajo inventario se cargan desde el backend con la función getAlertas() y se muestran como una lista. El usuario puede cerrarlas individualmente con el botón "Ocultar", que ejecuta una petición para desactivarlas (dismissAlerta()).

Componentes y archivos usados

- UserProfile.js: componente que muestra los datos del usuario y botones de acciones.
- EditProfileForm.js, ChangePasswordForm.js, AddUserForm.js: formularios modales para gestionar usuario.
- Modal.js: componente reutilizable para mostrar formularios emergentes.

- api.js: contiene las funciones base para comunicarse con el backend, como apiFetch (manejo unificado de peticiones con cookies y CSRF), apiFetchForm (para formularios multipart), y funciones auxiliares como getAlerts y dismissAlert.

- DashboardPage.css: archivo CSS específico para el diseño visual de esta vista (tarjetas, botones, alertas, etc.).

Diseño visual

- Se usa un diseño de tarjetas alineadas en un grid para representar cada métrica del sistema con un color de fondo distintivo.

- Las alertas se presentan como una lista ordenada con botones para cerrarlas.

- El mensaje de feedback al cerrar una alerta aparece con estilo condicional (verde para éxito, rojo para error), y se elimina automáticamente después de 4 segundos.

2.1.2.5 InventoryPage:

La página de inventario permite al usuario visualizar y gestionar todos los productos registrados en el sistema.

Funcionalidad principal

- **Listado de productos**: Mediante una petición GET a la API, se cargan todos los productos cuyo campo deleted_at sea nulo. Los productos se muestran con su nombre, categoría y proveedor, utilizando tarjetas visuales a través del componente ProductCard.js.

- **Búsqueda inteligente**: El usuario puede filtrar productos en tiempo real escribiendo en el campo de búsqueda. El filtro aplica sobre el nombre del producto, nombre de la categoría o nombre del proveedor.

- **Registro de productos**: Los administradores pueden presionar el botón "AÑADIR PRODUCTO" para abrir un formulario modal (AddProductForm) que permite ingresar datos como nombre, stock, proveedor y categoría.

- **Edición y eliminación**: Cada tarjeta de producto incluye botones de editar y eliminar que son solo visibles para administradores. La edición abre el componente EditProductForm en un modal, y la eliminación se realiza mediante una petición PATCH que actualiza el campo deleted_at del producto.

- **Recarga dinámica**: La página escucha un evento personalizado (recargarInventario) que permite actualizar el listado desde otras vistas. Además, tras cerrar un formulario de registro o edición, se vuelve a cargar automáticamente la lista de productos para reflejar los cambios.

Componentes y archivos usados

- ProductCard.js: componente visual que representa un producto individual con botones de edición y eliminación (si es administrador).

- AddProductForm.js: formulario utilizado para registrar nuevos productos, con campos seleccionables para proveedor y categoría.

- EditProductForm.js: formulario similar al de registro, pero precargado con los datos del producto a modificar.

- Modal.js: componente general que encapsula los formularios en ventanas emergentes.

- api.js: archivo de utilidades que define la URL base del backend y proporciona funciones para realizar peticiones HTTP de forma centralizada, como apiFetch y apiFetchForm, las cuales gestionan automáticamente el token CSRF y las cookies de sesión.

- InventoryPage.css: hoja de estilos específica para esta vista, utilizada para aplicar estilos a las tarjetas, botones y formularios.

Lógica de carga de datos

- **Productos**: Se cargan desde /products/ y se filtran localmente.

- **Proveedores y categorías**: Se obtienen desde /suppliers/ y /categories/ respectivamente, para mostrarlos como opciones en los formularios.

Diseño visual

La vista presenta una interfaz amigable y dinámica. Los productos se muestran en una lista de tarjetas (product-list), cada una con los datos clave. Si no hay productos que coincidan con la búsqueda, se muestra un mensaje informativo.

2.1.2.6 TransactionsPage:

La pantalla de movimientos es una de las vistas centrales del sistema, ya que permite registrar y visualizar tanto **entradas** como **salidas** de productos en el inventario. Esta sección facilita el control histórico de operaciones y la actualización automática del stock.

Funcionalidad principal

- **Visualización de movimientos**:
    - Los movimientos se dividen en dos secciones: **Entradas** (tipo input) y **Salidas** (tipo output).
    - Cada tabla muestra la información clave: fecha, producto, cantidad, proveedor o cliente, stock actual y usuario que registró el movimiento.
    - Los datos se obtienen mediante getMovimientos() desde el backend.
- **Registro de nuevos movimientos**:
    - Solo los usuarios con rol de **Administrador** pueden añadir entradas o salidas.
    - Al presionar **"Añadir Entrada"** o **"Añadir Salida"**, se abre un formulario dentro de un Modal personalizado.
    - El formulario incluye:
        - Fecha (datetime-local)
        - Cantidad
        - Producto
        - Cliente (solo si es una salida)
    - Al guardar, se valida si hay suficiente stock (para salidas) y se envía el movimiento al backend usando postMovimiento().

- **Actualización automática del inventario**:
    - Tras guardar un nuevo movimiento, se actualiza la lista de movimientos y productos.
    - Se dispara el evento recargarInventario para que otras vistas que escuchen este evento se sincronicen (como la página de inventario).
- **Validación de stock**:
    - Antes de guardar una salida, se verifica que la cantidad no exceda el stock actual del producto seleccionado sino lanza una advertencia.

Componentes y archivos usados

- Modal.js: ventana emergente que encapsula el formulario de nuevo movimiento.
- api.js: contiene las funciones getMovimientos, postMovimiento, getProductos y getClientes.
- TransactionsPage.css: hoja de estilos dedicada, que define el diseño de tablas, botones, formularios y alertas visuales.
- lucide-react: íconos usados para mejorar la experiencia visual (entrada, salida, fecha, cliente, producto, éxito/error, etc.).

Diseño visual

- El diseño se organiza en **dos tablas** con títulos: “Entradas” y “Salidas”.

- Cada fila alterna el color de fondo (estilo row-even / row-odd) para facilitar la lectura.
- Las cantidades aparecen con signos + o – y estilos en verde o rojo según el tipo de movimiento.
- El formulario modal es limpio, con íconos junto a cada campo y botones bien diferenciados.

2.1.2.7 ReportsPage:

La página de reportes permite generar documentos PDF con información relevante del sistema, como los movimientos de inventario o los productos más vendidos. Es una herramienta útil para los administradores, ya que ofrece un resumen descargable filtrado por rangos de fechas.

Funcionalidad principal

- **Selección del tipo de reporte**: El usuario puede elegir entre:
  - **Movimientos recientes**: muestra entradas y salidas registradas.
  - **Productos más vendidos**: genera un resumen de ventas ordenado por cantidad.
- **Filtro por fechas**: Se selecciona un **rango de fechas** con campos tipo date que permiten definir el período que se desea consultar.
- **Generación del reporte**:

- Al presionar el botón **“Generar Reporte PDF”**, se envía una solicitud POST al backend con los datos ingresados.
- Si el servidor responde exitosamente, se muestra un mensaje de confirmación y un enlace para descargar el archivo PDF generado.
- Si ocurre un error, se muestra un mensaje con el motivo o un mensaje genérico de error de conexión.

Componentes y archivos usados

- api.js: se utiliza para definir el API_URL base y centralizar las solicitudes al backend mediante funciones como apiFetch y apiFetchForm, que incluyen automáticamente el token CSRF y las cookies de sesión en cada petición.
- lucide-react: se usa el ícono FileText para mostrar visualmente el mensaje de estado.
- ReportsPage.css: hoja de estilos para definir el diseño de la tarjeta de reporte, los campos de entrada, el botón de acción, los mensajes y el enlace al PDF.

Diseño visual

- Se presenta una **tarjeta** centrada en pantalla con todos los campos y botones necesarios.
- El botón de generación muestra un ícono y cambia su mensaje según el resultado.
- El enlace para descargar el PDF se muestra solo si el archivo se generó correctamente.

2.1.2.8 SuppliersPage:

La pantalla de proveedores permite a los usuarios consultar y administrar el listado de proveedores registrados en el sistema. Esta sección es fundamental para mantener organizada la información de las empresas o personas que abastecen los productos del inventario.

Funcionalidad principal

- **Listado de proveedores**: Se realiza una consulta GET a la API para obtener todos los proveedores activos (que no tengan el campo deleted_at), y se muestran como tarjetas en pantalla.

- **Búsqueda en tiempo real**: A través de un campo de texto con ícono de lupa, el usuario puede buscar proveedores por nombre, RUC o cédula, correo o teléfono. El filtro se aplica directamente sobre los datos cargados.

- **Agregar proveedor**: Si el usuario tiene rol de administrador, puede hacer clic en el botón **"AÑADIR PROVEEDOR"**, que abre el formulario AddSupplierForm dentro de un modal. Este formulario permite registrar nuevos proveedores con sus datos básicos.

- **Editar proveedor**: Cada tarjeta de proveedor muestra un botón de edición que carga el componente EditSupplierForm con los datos del proveedor seleccionado.

- **Eliminación lógica**: Al presionar el botón de eliminar, se envía una petición PATCH al backend para establecer el campo deleted_at del proveedor, sin eliminarlo físicamente de la base de datos.

Componentes y archivos usados

- AddSupplierForm.js: formulario para registrar nuevos proveedores.
- EditSupplierForm.js: formulario que permite editar la información de un proveedor existente.
- Modal.js: componente reutilizable que encapsula ambos formularios dentro de una ventana emergente.
- api.js: contiene la URL base del backend y funciones como apiFetch para autenticar y gestionar solicitudes protegidas, incluyendo automáticamente el token CSRF y las cookies de sesión.
- SuppliersPage.css: hoja de estilos para esta vista, que incluye tarjetas, botones y campos de búsqueda.

Diseño visual

- Cada proveedor se muestra como una **tarjeta** con su información clave: nombre, correo, teléfono, RUC o cédula y dirección.
- Los botones de acción (editar y eliminar) solo se muestran a los usuarios con rol de administrador.
- Si no hay resultados para la búsqueda actual, se muestra un mensaje informativo al usuario.

2.1.2.9 CustomersPage:

La página de clientes permite a los usuarios visualizar, buscar, registrar, editar y eliminar información de los clientes registrados en el sistema. Esta funcionalidad está limitada por roles: solo los usuarios con rol de **Administrador** pueden realizar modificaciones.

Funcionalidad principal

- **Listado de clientes**: Se realiza una petición GET a la API para obtener los clientes activos (que no tienen deleted_at).
- **Búsqueda**: A través de un campo de texto con ícono de lupa, se filtran los clientes en tiempo real por nombre, cédula, teléfono o correo.
- **Creación de cliente**: Si el usuario tiene el rol de administrador, se habilita un botón "AÑADIR CLIENTES", el cual muestra un formulario (AddCustomerForm) dentro de un modal reutilizable.
- **Edición**: Cada tarjeta de cliente incluye un botón de edición que carga los datos en un formulario (EditCustomerForm) para modificarlos.
- **Eliminación lógica**: Al presionar el ícono de eliminar, se actualiza el campo deleted_at mediante una petición PATCH, sin borrar definitivamente el cliente.
- **Componente Modal**: Se utiliza un componente de ventana emergente (Modal) para encapsular los formularios de creación y edición, mejorando la experiencia de usuario.
- **Control de acceso**: Mediante la variable isAdmin se limita la visualización de botones de acciones sensibles solo a los administradores.

Componentes y archivos usados

- Modal.js: componente reutilizable para mostrar formularios en ventanas emergentes.
- AddCustomerForm.js: formulario para ingresar un nuevo cliente.
- EditCustomerForm.js: formulario para editar la información de un cliente existente.
- api.js: archivo donde se define la constante API_URL y funciones como apiFetch y apiFetchForm, que gestionan automáticamente el token CSRF y las cookies de sesión requeridas por Django en peticiones protegidas.
- CustomersPage.css: hoja de estilos asociada para aplicar diseño a los elementos como botones, tarjetas y formularios.

Diseño visual

La información de los clientes se presenta en **tarjetas** con campos como nombre, correo, teléfono y cédula. En caso de no existir información, se reemplaza por un guion (-). Si no hay resultados tras la búsqueda, se muestra un mensaje claro al usuario.

2.1.2.10 UsersPage:

La pantalla de usuarios permite a los administradores gestionar las cuentas del sistema. Desde esta vista se pueden **registrar nuevos usuarios**, **modificar sus roles** y **activar o inactinvar** su acceso, lo que permite mantener un control total sobre quién puede ingresar al sistema.

Funcionalidad principal

- **Listado de usuarios**:
    - Se realiza una consulta GET al backend para obtener todos los usuarios registrados.
    - Se presentan en una tabla con columnas: nombre, correo, teléfono, rol, estado y acciones.
- **Creación de nuevos usuarios**:
    - Mediante el botón **"Añadir Usuario"**, se abre un formulario modal (AddUserForm) que permite registrar una nueva cuenta con sus datos personales y rol asignado.
- **Cambio de rol**:
    - Los administradores pueden cambiar el rol de otros usuarios entre **"Usuario"** y **"Administrador"** mediante un desplegable (select).
    - Este cambio se envía al backend mediante una solicitud PATCH.
- **Activar/Inactivar usuarios**:
    - El estado Activo o Inactivo se representa con un círculo de color (verde o rojo).
    - Se puede activar o inactivar un usuario haciendo clic en el botón correspondiente. Esto actualiza el campo is_active del usuario.
- **Protección de acciones sensibles**:

- El usuario autenticado no puede modificarse a sí mismo ni cambiar su propio rol o estado.
- Si un usuario no tiene el rol de **Administrador**, se redirige automáticamente al Dashboard, impidiendo el acceso no autorizado.

Componentes y archivos usados

- AddUserForm.js: formulario que permite registrar nuevos usuarios.
- Modal.js: ventana emergente donde se presenta el formulario de creación.
- api.js: define la URL base del backend y centraliza las solicitudes protegidas mediante funciones como apiFetch y apiFetchForm, que gestionan automáticamente el token CSRF y las cookies de sesión.
- UsersPage.css: hoja de estilos que define el diseño de la tabla, botones, formulario y elementos visuales como íconos de estado.

Diseño visual

- La página muestra una tabla moderna y clara, con íconos de estado (verde para activo, rojo para inactivo).
- Los botones de acción se muestran solo si el usuario tiene permisos para modificarlos.
- Si no hay usuarios registrados, se muestra un mensaje informativo en lugar de la tabla.

- Las acciones tienen una retroalimentación visual mediante el estado loadingId, que evita múltiples clics o actualizaciones en paralelo.

2.1.2.11 QuotationPage:

La página de cotizaciones permite crear propuestas comerciales formales para clientes seleccionados, detallando los productos, cantidades, precios y observaciones. Además, ofrece la posibilidad de generar automáticamente un documento PDF descargable con la información registrada.

Funcionalidad principal

- **Selección de cliente**: Se carga una lista de clientes disponibles mediante getClientes(), y el usuario puede seleccionar uno desde un menú desplegable (select).
- **Añadir productos a la cotización**: El usuario puede agregar múltiples productos a la cotización. Cada producto tiene los siguientes campos:
    - Producto seleccionado desde la lista (select)
    - Cantidad (input)
    - Precio unitario (input, precargado automáticamente según el producto)
    - Subtotal calculado automáticamente
    - Opción para eliminar el producto de la lista (X)
- **Cálculo de totales**: Cada vez que se modifica un producto, se recalculan automáticamente los siguientes valores:

- **Subtotal**: suma total sin impuestos.
- **IVA (15%)**
- **Total**: suma final con impuestos.

- **Observaciones**: Campo de texto que permite ingresar comentarios adicionales sobre la cotización, como notas de validez o condiciones especiales.

- **Guardar cotización**: Al presionar el botón **"Guardar Cotización"**, se envía una petición POST al backend con los datos recopilados. Si la operación es exitosa:
  - Se limpia el formulario.
  - Se muestra un mensaje de éxito.
  - Se obtiene automáticamente el enlace al PDF generado para visualizar o descargar.

- **Generación y visualización del PDF**: El sistema solicita el PDF desde el backend usando getCotizacionPDF(id), y muestra un enlace directo para abrirlo en una nueva pestaña.

Componentes y archivos usados

- **Iconos de lucide-react**: Para mejorar la interfaz visual y destacar secciones (cliente, productos, resumen, observaciones, guardar, PDF).

- **getClientes() y getProductos()**: funciones del archivo api.js para obtener los datos desde el backend.

- **getCotizacionPDF()**: función que retorna el enlace al documento generado.
- **QuotationPage.css**: hoja de estilos que define la disposición de cada sección, botones, inputs, íconos, mensajes y diseño general.

Diseño visual

- Interfaz dividida en secciones claras:
    1. Información del cliente
    2. Productos cotizados
    3. Resumen de cotización
    4. Observaciones
- Cada sección tiene un ícono representativo y un estilo visual consistente.
- Mensajes de éxito o error aparecen centrados y con colores distintivos.
- El botón para descargar el PDF aparece solo si la cotización fue guardada correctamente.

Cada página importa componentes reutilizables desde components/, como se puede ver en cada una se especifica que componentes se utiliza lo que ayuda a que cada página tenga más funcionalidades que ayudan al funcionamiento de cada uno siendo los componentes algo muy importante para el sistema.

2.2 Construcción del Sistema Backend

2.2.1 Estándares de Construcción

2.2.1.1 Metodología por Capas (Modelo, Servicio, Vista, Serializador)

Para el desarrollo del backend del sistema se utilizó el framework **Django**, junto con **Django REST Framework**, aplicando una **arquitectura por capas** que permite garantizar un sistema limpio, escalable y fácil de mantener.

Esta arquitectura divide el backend en los siguientes componentes:

- **Modelos**: Representan las tablas de la base de datos y definen la estructura de los datos.
- **Servicios**: Contienen la lógica de negocio y se encargan de procesar datos antes o después de ser almacenados.
- **Vistas (ViewSets)**: Gestionan las solicitudes del frontend, recibiendo datos del cliente y respondiendo con los resultados adecuados.
- **Serializadores**: Transforman los datos entre objetos de Python y formato JSON, permitiendo una comunicación clara entre backend y frontend.

2.2.2 Buenas Prácticas Aplicadas

Durante la construcción del sistema se aplicaron diversas prácticas de desarrollo que permitieron mantener la calidad y organización del código. Entre ellas se destacan:

- **Separación de responsabilidades**: Cada archivo y clase tiene una única función específica, siguiendo el *Principio de Responsabilidad Única* (*Single Responsibility Principle*).
- **Modularidad**: Todos los componentes del sistema están organizados por entidad en módulos separados, facilitando su mantenimiento y comprensión.
- **Reutilización de código**: Se reutilizan funciones y estructuras mediante clases de servicio, métodos estáticos y consultas encapsuladas en repositorios.
- **Mantenibilidad**: La estructura clara y ordenada del proyecto permite hacer modificaciones o extensiones con facilidad, sin afectar otras partes del sistema.
- **Claridad y legibilidad del código**: Se utilizaron nombres descriptivos, indentación adecuada y comentarios cuando fue necesario para mejorar la comprensión del código fuente.

Estas decisiones fueron fundamentales para garantizar un desarrollo eficiente y una base sólida sobre la cual se construyeron los diferentes módulos del sistema.

2.2.3 Estándares de Desarrollo y Estructura del Proyecto

Para garantizar una construcción eficiente, escalable y ordenada del sistema backend, se definieron y aplicaron una serie de **estándares de desarrollo** que permiten mantener una estructura coherente y fácil de mantener a lo largo del tiempo.

2.2.4 Estructura del Proyecto

El backend fue desarrollado utilizando el framework **Django** junto con **Django REST Framework**, bajo una estructura de carpetas organizada por capas y entidades, lo cual permite un desarrollo modular y una rápida localización de los componentes.

La organización de carpetas principal del proyecto es la siguiente:

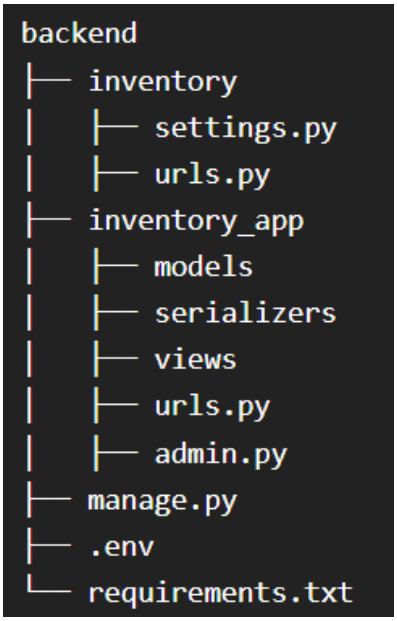


Ilustración 21: Estructura del Proyecto

Esta estructura sigue los principios de la **arquitectura por capas**, donde cada carpeta representa una responsabilidad del sistema, y se divide aún más por entidades (product, movement, user, etc.), manteniendo el código limpio y organizado.

2.2.5 Convenciones de Nomenclatura

Para asegurar uniformidad y facilitar la lectura del código, se adoptaron las siguientes convenciones:

2.2.5.1 Nombres de Archivos

Todos los archivos siguen el estilo snake_case, recomendado por la guía de estilo **PEP 8** de Python. Esto implica utilizar letras minúsculas y guiones bajos para separar palabras.

Ejemplos:

- report_view.py
- aler_serializer.py

2.2.5.2 Nombres de Clases

Las clases están escritas en CamelCase, donde cada palabra inicia con mayúscula y no se usan guiones bajos.

Ejemplos:

- CustomerListCreateView
- MovementSerializer
- Product

2.2.5.3 Nombres de Métodos y Variables

Tanto los métodos como las variables usan sanake_case, tal como es estándar en Python.

Ejemplos:

- get_queryset(self)
- current_stock, user_role
- perfom_create(self, serializer)
- get_query

2.2.5.4 Organización de Importaciones

Las importaciones se organizan de forma jerárquica y ordenada:

1. Librerías estándar de Python.
2. Librerías externas como Django o DRF.
3. Módulos internos del proyecto (models, serializers, etc.).

2.2.6 Otras Buenas Prácticas Aplicadas

- Se utilizaron nombres descriptivos para todas las clases, funciones y variables.
- Se evitaron comentarios innecesarios y, en su lugar, se prefirió un código autodescriptivo.
- Se aplicaron decoradores como @staticmethod en métodos reutilizables que no dependen de instancias.
- Se utilizó paginación, filtros y validaciones dentro de las vistas para mejorar la eficiencia.
- Se evitó repetir lógica, reutilizando métodos en servicios y validaciones comunes.

2.2.7 Definición y Codificación de Modelos

Los modelos representan la estructura de datos del sistema. Cada modelo equivale a una tabla en la base de datos PostgreSQL y fue implementado utilizando el **ORM (Object-Relational Mapping)** de Django, lo que facilita la creación, modificación y consulta de datos sin necesidad de escribir SQL directamente.

Los modelos están organizados en archivos separados dentro de la carpeta inventory_app/models/, lo que permite una arquitectura ordenada y por entidad. Todos incluyen campos de auditoría (created_at, updated_at, deleted_at) para registrar fechas de creación, actualización y eliminación lógica de los registros.

A continuación, se describe la función y codificación de cada modelo:

2.2.7.1 Modelo User

Este modelo representa a los usuarios del sistema, permitiendo autenticación y control de accesos según roles.

Atributos:

- email: campo principal de autenticación, único.
- name: nombre del usuario.
- role: puede ser Administrator o User.
- phone: número de teléfono con validación de 10 dígitos.
- is_active, is_staff, is_superuser: campos propios de Django para permisos.
- created_at, updated_at, deleted_at: control de auditoría.

Además, se personalizó el UserManager para crear usuarios y superusuarios sin necesidad de username, utilizando el correo electrónico como identificador principal.

2.2.7.2 Modelo Product

Define los productos gestionados en el inventario.

Atributos:

- name, description: nombre y descripción del producto.
- category: relación con el modelo Category.
- price: precio unitario.
- current_stock: cantidad actual disponible.
- minimum_stock: mínimo permitido antes de emitir alertas.
- status: estado del producto.
- supplier: proveedor asociado.
- image: imagen opcional del producto.
- created_at, updated_at, deleted_at: campos de auditoría.

Este modelo permite visualizar el inventario, controlar entradas y salidas, y emitir alertas automáticas.

2.2.7.3 Modelo Category

Permite agrupar productos por tipo o categoría.

Atributos:

- name: nombre único de la categoría.
- created_at, updated_at, deleted_at.

2.2.7.4 Modelo Supplier

Registra a los proveedores que suministran productos.

Atributos:

- name, email, tax_id, phone, address: información de contacto.
- Validación de teléfono con exactamente 10 dígitos.
- created_at, updated_at, deleted_at.

2.2.7.5 Modelo Customer

Representa a los clientes que reciben productos (salidas).

Atributos:

- name, email, document, phone, address: datos básicos del cliente.
- Teléfono validado con 10 dígitos.
- created_at, updated_at, deleted_at.

2.2.7.6 Modelo Movement

Registra cada movimiento de inventario, ya sea entrada (IN) o salida (OUT).

Atributos:

- movement_type: tipo de movimiento.
- date: fecha y hora del movimiento.
- quantity: cantidad de productos.
- product: relación con el modelo Product.
- user: usuario que registra el movimiento.
- customer: cliente (solo para salidas).
- stock_in_movement: stock del producto al momento del movimiento.

- created_at, updated_at, deleted_at.

El stock del producto se actualiza automáticamente al registrar el movimiento, y se valida si hay suficiente stock para las salidas.

2.2.7.7 Modelo Alert

Este modelo se activa automáticamente cuando un producto tiene bajo stock.

Atributos:

- type: tipo de alerta (low_stock, one_unit, out_of_stock).
- message: descripción de la alerta.
- product: producto relacionado.
- created_at, updated_at, deleted_at.

Este modelo permite al sistema emitir advertencias visuales y mantener el inventario controlado.

2.2.7.8 Modelo Quotation

Permite registrar cotizaciones realizadas a clientes.

Atributos:

- date: fecha automática de generación.
- subtotal, tax, total: valores financieros de la cotización.
- notes: observaciones.
- customer: cliente al que va dirigida la cotización.

- user: usuario que la generó.
- created_at, updated_at, deleted_at.

2.2.7.9 Modelo QuotedProduct

Es un modelo intermedio que guarda el detalle de cada producto cotizado dentro de una cotización.

Atributos:

- product: producto cotizado.
- quotation: cotización a la que pertenece.
- quantity: cantidad cotizada.
- unit_price: precio unitario.
- subtotal: se calcula automáticamente multiplicando cantidad * precio.
- created_at, updated_at, deleted_at.

Incluye una sobrescritura del método save() para calcular automáticamente el subtotal antes de guardar.

2.2.7.10 Modelo Report

Este modelo almacena un historial de los reportes generados en PDF.

Atributos:

- file: archivo PDF generado.
- user: usuario que lo generó.

- generated_at: fecha y hora de creación.

En conclusión, los modelos desarrollados representan fielmente la lógica de negocio y las entidades del sistema. La correcta definición de relaciones, validaciones y campos de auditoría permite mantener una base de datos coherente, organizada y funcional, facilitando tanto la gestión del inventario como el registro de cotizaciones, movimientos y reportes.

2.2.8 Implementación de Serializers

En Django REST Framework, los serializadores cumplen una función clave dentro de la arquitectura del sistema, ya que son responsables de **transformar los datos** entre los modelos de Django y el formato JSON que se utiliza para la comunicación con el frontend.

Cada entidad del sistema tiene su propio serializador, el cual se encarga de:

- Convertir objetos Python (como instancias de modelos) a JSON para enviarlo al cliente.
- Validar y transformar datos JSON recibidos del frontend para convertirlos en instancias de modelo antes de guardarlas.
- Personalizar los campos que se exponen, ocultar información sensible, renombrar atributos y crear campos calculados.

A continuación, se describen los serializadores implementados para las principales entidades del sistema:

2.2.8.1 AlertSerializer

Este serializador transforma los datos del modelo Alert, mostrando información del producto relacionado y el tipo de alerta en formato legible para el usuario.

Campos personalizados:

- product_name: extraído desde product.name.
- type_display: muestra el valor legible del tipo de alerta (Stock bajo, Sin stock, etc.).

Esto permite al frontend mostrar mensajes amigables y claros al usuario final.

2.2.8.2 CategorySerializer, CustomerSerializer, SupplierSerializer, ProductSerializer

Estos serializadores siguen una estructura directa utilizando fields = '__all__', lo que significa que exponen todos los campos del modelo correspondiente.

Esta configuración es útil para CRUD simples donde no se necesita ocultar ni transformar campos.

2.2.8.3 MovementSerializer

Este serializador es uno de los más complejos, ya que transforma el modelo Movement y además **enriquece** los datos con campos de lectura adicionales para el frontend.

Campos personalizados:

- product_name: nombre del producto.
- product_stock: stock del producto al momento del movimiento.
- supplier_name: proveedor del producto (si existe).
- customer_name: nombre del cliente (en salidas).
- user_name: nombre del usuario que registró el movimiento.

Gracias a estos campos calculados, el frontend puede mostrar datos clave sin realizar consultas adicionales.

2.2.8.4 UserSerializer

Este serializador transforma la información del modelo de usuarios. Contiene validaciones especiales para ocultar el campo password en las respuestas y manejarlo de forma segura durante la creación o actualización.

Características especiales:

- password es **escritura solamente** (write_only=True).
- Se sobrescriben los métodos create y update para cifrar correctamente la contraseña utilizando set_password().

Esto garantiza la seguridad del sistema en el manejo de credenciales.

2.2.8.5 QuotationSerializer

Serializador avanzado que maneja la cotización completa, incluyendo su relación con los productos cotizados.

Características destacadas:

- Campo anidado: quoted_products, que contiene una lista de productos cotizados usando QuotedProductSerializer.
- Renombramiento de campos para el frontend:
  - tax se presenta como vat.
  - notes se presenta como observations.

2.2.8.6 Método personalizado:

- create(): sobrescrito para crear primero la cotización y luego los productos cotizados asociados, asegurando la relación correcta.

Este enfoque facilita el envío de toda la cotización desde el frontend en una sola solicitud.

2.2.8.7 QuotedProductSerializer

Transforma los productos cotizados dentro de una cotización. Calcula automáticamente el subtotal (cantidad × precio) como campo solo de lectura (read_only=True).

Al mantener el cálculo en el backend, se garantiza la consistencia de los datos financieros.

2.2.8.8 ReportSerializer

Este serializador transforma el modelo Report, el cual contiene el archivo PDF generado y el usuario que lo creó.

Características:

- El campo user es de solo lectura (read_only_fields = ['user']), ya que se asigna automáticamente en la vista al generar el reporte.

Los serializadores en este sistema están diseñados no solo para transformar datos, sino también para:

- Mejorar la experiencia del usuario mostrando información enriquecida.
- Proteger datos sensibles como contraseñas.

- Validar correctamente la información enviada por el frontend.
- Gestionar estructuras anidadas complejas como cotizaciones con múltiples productos.

Esta capa del sistema actúa como un puente vital entre la base de datos y la interfaz gráfica, asegurando una comunicación eficiente, segura y amigable.

2.2.9 Implementación de Controladores (Vistas)

Las vistas del sistema fueron implementadas utilizando el framework Django REST Framework, lo que permitió definir clases que gestionan solicitudes HTTP (GET, POST, PATCH, DELETE) de forma estructurada. Estas vistas se organizaron por entidad en archivos separados dentro de la carpeta inventory_app/views/, favoreciendo una arquitectura clara y mantenible. Cada vista se encarga de aplicar permisos, procesar datos y delegar operaciones al modelo correspondiente.

A continuación, se describen las vistas implementadas, organizadas por funcionalidad:

2.2.9.1 Usuarios y Autenticación

Se implementaron diversas vistas que permiten gestionar tanto el proceso de autenticación como el manejo completo de usuarios:

- **LoginView**: permite autenticar a un usuario mediante email y contraseña. Si el usuario está inactivo, retorna un mensaje de advertencia.
- **ForgotPasswordView**: genera un token y envía un correo electrónico con el enlace de recuperación de contraseña.

- **ResetPasswordView**: valida el token y permite establecer una nueva contraseña.

- **ChangePasswordView**: permite a usuarios autenticados cambiar su contraseña actual.

- **UserListCreateView y UserDetailView**: permiten a usuarios con rol de "Administrador" crear, listar, editar y consultar usuarios.

Para restringir el acceso, se implementó la clase personalizada **IsAdmin**, que verifica que el usuario autenticado tenga el rol "Administrador".

2.2.9.2 Clientes

- **CustomerListCreateView**: permite a administradores listar y crear nuevos clientes. Verifica que el usuario tenga permisos adecuados antes de guardar.

- **CustomerDetailView**: permite consultar o editar los datos de un cliente existente, también con restricción al rol "Administrador".

2.2.9.3 Proveedores

- **SupplierListCreateView y SupplierDetailView**: Permiten gestionar proveedores del sistema. La creación y edición solo puede ser realizada por administradores. También se filtran los proveedores eliminados lógicamente.

2.2.9.4 Categorías

- **CategoryListCreateView y CategoryDetailView**: permiten crear, listar, consultar, actualizar y eliminar categorías. No requieren permisos especiales, por lo que cualquier usuario autenticado puede utilizarlas.

2.2.9.5 Productos

- **ProductListCreateView**: permite listar productos activos (no eliminados) y crear nuevos registros. La creación está restringida a administradores.

- **ProductDetailView**: habilita la consulta y actualización de productos. Solo administradores pueden realizar actualizaciones.

2.2.9.6 Movimientos de Inventario

- **MovementListCreateView**: permite listar movimientos y registrar entradas o salidas de inventario. Valida que:

    - Solo usuarios con rol "User" o "Administrator" puedan registrar movimientos.

    - Las salidas no excedan el stock actual del producto.

    - El stock del producto se actualice correctamente.

Además, esta vista genera automáticamente alertas si el producto cae por debajo del stock mínimo, queda una sola unidad o se agota por completo.

2.2.9.7 Alertas de Bajo Inventario

- **AlertListView**: lista todas las alertas activas (no eliminadas) ordenadas por fecha de creación.

- **AlertUpdateView**: permite "descartar" una alerta marcándola como eliminada mediante un campo deleted_at.

2.2.9.8 Cotizaciones

- **QuotationCreateView**: permite crear una cotización completa junto con sus productos cotizados. Calcula automáticamente el subtotal, IVA (15%) y total, y permite registrar todo en una sola solicitud.

- **QuotationListView**: lista las cotizaciones, filtrando por usuario en caso de usuarios normales, y mostrando todas para los administradores.

- **QuotationDetailView**: permite consultar los detalles de una cotización específica.

- **QuotationPDFView**: genera un archivo PDF con los detalles de la cotización, incluyendo logo, cliente, productos, totales e información adicional. El archivo se guarda y se registra en la base de datos mediante el modelo Report.

2.2.9.9 Reportes

- **ReportListView**: permite a cada usuario consultar los reportes PDF generados previamente.

- **ReportGeneratePDFView**: genera un reporte en PDF según el tipo seleccionado:

    - **"top_vendidos"**: muestra los 10 productos más vendidos en un rango de fechas.

    - **"movimientos"**: muestra los últimos 50 movimientos de inventario, incluyendo tipo, fecha, producto, cantidad, cliente o proveedor y usuario.

El archivo generado se guarda en la carpeta correspondiente y se registra en la base de datos.

2.2.9.10 Dashboard

- **DashboardSummaryView**: ofrece un resumen general con los siguientes indicadores:
    - Total, de productos, clientes y movimientos
    - Total, de entradas y salidas
    - Número de alertas activas
    - Cantidad total de productos vendidos

Esta información se utiliza para alimentar los gráficos e indicadores en el frontend del dashboard.

2.2.10 Definición de Rutas

Las rutas del sistema fueron definidas en el archivo urls.py dentro del módulo inventory_app. Estas rutas permiten exponer los distintos endpoints de la API REST y conectar las vistas (views) con el frontend del sistema. Cada ruta está asociada a una clase controladora (vista) que procesa la lógica correspondiente según la entidad.

Las rutas se organizaron en bloques temáticos, agrupando funcionalidades similares para mantener un orden claro y comprensible.

A continuación, se detallan las principales rutas implementadas:

1. Autenticación

- POST /login/ – Permite a los usuarios iniciar sesión.
- POST /forgot-password/ – Envía un correo para recuperar la contraseña.
- POST /reset-password/ – Restablece la contraseña con token.
- POST /change-password/ – Cambia la contraseña autenticada.

2. Gestión de Usuarios

- GET /users/ – Lista todos los usuarios del sistema.
- POST /users/ – Crea un nuevo usuario.
- GET /users/<id>/ – Muestra los detalles de un usuario específico.
- PATCH /users/<id>/ – Actualiza datos o rol del usuario.

3. Clientes (Customers)

- GET /customers/ – Lista todos los clientes.
- POST /customers/ – Registra un nuevo cliente.
- GET /customers/<id>/ – Muestra detalles de un cliente.
- PUT /customers/<id>/ – Edita la información del cliente.

4. Proveedores (Suppliers)

- GET /suppliers/ – Lista todos los proveedores.
- POST /suppliers/ – Agrega un nuevo proveedor.

- GET /suppliers/<id>/ – Detalles del proveedor.
- PUT /suppliers/<id>/ – Edita los datos del proveedor.

5. Productos (Products)

- GET /products/ – Lista los productos activos.
- POST /products/ – Crea un nuevo producto.
- GET /products/<id>/ – Detalles de un producto.
- PUT /products/<id>/ – Modifica un producto existente.

6. Categorías

- GET /categories/ – Lista de categorías.
- POST /categories/ – Crea una nueva categoría.
- GET /categories/<id>/ – Detalles de una categoría.
- PUT /categories/<id>/ – Modifica la categoría.

7. Movimientos de Inventario

- GET /movements/ – Lista de movimientos (entradas y salidas).
- POST /movements/ – Registra un nuevo movimiento (entrada o salida).

8. Alertas de Bajo Inventario

- GET /alerts/ – Lista de alertas activas.
- PATCH /alerts/<id>/dismiss/ – Descarta una alerta (eliminación lógica).

9. Cotizaciones

- GET /quotations/ – Lista cotizaciones existentes.
- POST /quotations/create/ – Crea una nueva cotización.
- GET /quotations/<id>/ – Consulta los detalles de una cotización.
- GET /quotations/pdf/<id>/ – Genera el archivo PDF de la cotización.

10. Reportes

- GET /reports/ – Muestra los reportes generados por el usuario.
- POST /reports/generate/ – Genera un nuevo reporte PDF según el tipo.

11. Dashboard

- GET /dashboard/summary/ – Muestra un resumen general del sistema: total de productos, clientes, ventas, entradas, salidas y alertas.

2.2.11 Configuración de la Base de Datos y Conexión

Para garantizar un almacenamiento seguro y estructurado de la información, el sistema utiliza como motor de base de datos a **PostgreSQL**, el cual se integró con Django a través del backend postgresql_psycopg2.

Con el objetivo de proteger la información sensible como credenciales de acceso, se implementó el uso de variables de entorno mediante la librería django-environ. Estas variables se almacenan en un archivo .env, el cual no forma parte del repositorio ni de la documentación, cumpliendo con las buenas prácticas de desarrollo seguro.

A continuación, se presenta un ejemplo de cómo se configuró la conexión a la base de datos dentro del archivo settings.py, sin exponer valores reales:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': env('DB_NAME'),
        'USER': env('DB_USER'),
        'PASSWORD': env('DB_PASSWORD'),
        'HOST': env('DB_HOST'),
        'PORT': env('DB_PORT'),
    }
}
```

Este bloque de configuración utiliza variables externas para definir el nombre de la base de datos, el usuario, la contraseña, el host y el puerto. Gracias a esto, es posible cambiar de entorno (por ejemplo, de desarrollo a producción) sin modificar el código fuente, simplemente editando el archivo. env. Además, esta arquitectura facilita el mantenimiento, mejora la seguridad y permite que el proyecto pueda ser desplegado fácilmente en distintos entornos de forma profesional.

2.3 Anexos

https://github.com/AlexanderPavon/qualitycore-inventory-backend

https://github.com/alcarrion/qualitycore-inventory-frontend

Capítulo III

3.1 Pruebas y Estabilización Frontend

3.1.1 Autenticación de Usuario

3.1.1.1 Registro de Usuario desde interfaz

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el formulario de registro permita crear un nuevo usuario desde la interfaz, validando los campos requeridos y mostrando un mensaje de confirmación si el registro es exitoso.	
Precondiciones	• El sistema debe estar encendido y funcionando correctamente. • El usuario actual debe tener el rol de administrador. • Debe estar disponible la conexión con la base de datos. • Debe estar disponible el botón para agregar nuevos usuarios en la interfaz.
Pasos para seguir	1. Iniciar sesión como administrador. 2.Ir a la sección "Usuarios" desde el menú lateral. 3.Presionar el botón "Agregar nuevo usuario". 4.Complete el formulario con los datos solicitados: • Nombre • Correo electrónico • Rol del usuario • Contraseña 5.Presionar el botón "Guardar". 6.Verifique que aparezca el nuevo usuario en la tabla. 7.Confirmar que se muestre un mensaje indicando que el registro fue exitoso.
Datos de Entrada	• Nombre: Camila Torres • Correo: camila.torres@quality.com • Rol: Usuario • Contraseña: Camila123
Resultados Esperados	• Todos los campos ingresados se validan correctamente. • La contraseña cumple con los requisitos mínimos de seguridad. • El sistema guarda el nuevo usuario y lo muestra en la tabla. • Se visualiza un mensaje confirmando que el registro fue exitoso.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Todos los campos se validan correctamente. • La contraseña cumple con los requisitos de seguridad. • El registro se completó exitosamente.

	• Se muestra el mensaje de éxito. • El nuevo usuario se visualiza en la tabla de usuarios. **RECHAZO:** • Algún campo está vacío. • El correo electrónico tiene un formato incorrecto. • La contraseña no cumple con los requisitos mínimos. • El sistema muestra un error. • No se muestra el mensaje de confirmación.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 11: Registro de Usuarios

3.1.1.2 Inicio de Sesión

DESCRIPCIÓN DEL CASO DE PRUEBA Comprobar que el formulario de inicio de sesión permita acceder al sistema con credenciales válidas y muestre mensajes de error si los datos son incorrectos o el usuario está inactivo.	
Precondiciones	• El sistema debe estar funcionando correctamente. • Debe existir al menos un usuario activo registrado en el sistema. • Debe estar disponible la pantalla de inicio de sesión.
Pasos para seguir	1. Ingresar al sistema desde el navegador. 2.Visualizar el formulario de inicio de sesión. 3. Ingresar el correo electrónico del usuario. 4.Ingresar la contraseña correspondiente. 5.Presionar el botón “Iniciar sesión”. 6. Verifique que se muestre la pantalla del tablero si los datos son válidos. 7. Verifique que se muestre un mensaje de error si los datos son incorrectos.
Datos de Entrada	• Correo: juan.perez@quality.com • Contraseña: Juan1234
Resultados Esperados	• El sistema valida las credenciales ingresadas. • Si los datos son correctos, el usuario accede al sistema y se redirige al panel principal (dashboard). • Si los datos son incorrectos o el usuario está inactivo, se muestra un mensaje de error indicando que no se pudo iniciar sesión.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Las credenciales se validan correctamente.

	• El usuario inicia sesión y accede al panel. • Se almacena la sesión del usuario. • El sistema responde rápidamente sin errores. **RECHAZO:** • Algún campo está vacío. • El correo o la contraseña son incorrectos. • El usuario está inactivo. • El sistema muestra un error inesperado.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 12: Inicio de Sesión

3.1.1.3 Recuperar Contraseña

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que el sistema envíe un enlace de recuperación al correo electrónico ingresado, mostrando mensajes claros tanto en caso de éxito como de error.	
Precondiciones	• El sistema debe estar activo y funcionando correctamente. • El usuario debe estar previamente registrado con un correo válido. • El formulario de recuperación debe estar disponible desde la pantalla de inicio de sesión.
Pasos para seguir	1. Acceda al sistema y haga clic en "¿Olvidaste tu contraseña?". 2. Visualizar el formulario de recuperación. 3. Ingresar el correo electrónico registrado. 4. Presionar el botón "Recuperar contraseña". 5. Verifique que el sistema muestre un mensaje indicando que se envió el correo. 6. Comprobar que el usuario reciba un correo con un enlace para restablecer su contraseña.
Datos de Entrada	• Correo electrónico: laura.mendoza@quality.com
Resultados Esperados	• El sistema valida que el correo esté registrado. • Se genera un token y se envía un correo con el enlace de recuperación. • Se muestra un mensaje en pantalla indicando que el correo fue enviado correctamente.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El correo ingresado está registrado. • El sistema genera correctamente el enlace de recuperación. • El mensaje de confirmación se muestra en pantalla. • El correo llega al buzón del usuario.

	RECHAZO: • El campo de correo está vacío. • El correo no está registrado en el sistema. • El formato del correo es inválido. • El sistema muestra un error inesperado. • No se genera ni se envía el enlace de recuperación.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 13: Recuperar Contraseña

3.1.4 Dashboard

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el tablero principal del sistema muestre correctamente la información general del inventario, incluyendo el número total de productos, movimientos y clientes, así como las gráficas e indicadores visuales, y que los datos estén actualizados y se carguen sin errores.	
Precondiciones	• El sistema debe estar en funcionamiento. • El usuario debe haber iniciado sesión correctamente. • Debe existir información registrada en el sistema (productos, clientes, movimientos, etc.).
Pasos para seguir	1. Iniciar sesión con un usuario válido. 2. Ser redirigido automáticamente al tablero del sistema. 3. Verificar la visualización de los indicadores (total de productos, clientes y movimientos). 5.Comparar los datos del tablero con la información real registrada.
Datos de Entrada	(No aplica – los datos se cargan automáticamente desde la base de datos)
Resultados Esperados	• Los indicadores muestran el número real de productos, clientes y movimientos. • El usuario puede visualizar toda la información sin errores de carga.

Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El usuario visualiza correctamente los totales de cada sección. • Los datos mostrados coinciden con lo registrado en el sistema. • La página carga de forma rápida y completa. **RECHAZO:** • Algún valor aparece vacío o incorrecto. • La información no está actualizada. • Se presenta un error visual o técnico al cargar el tablero.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 14: Dashboard

3.1.5 Gestión de usuarios (crear nuevo usuario)

3.1.5.1 Crear nuevo usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el formulario modal de "Añadir nuevo usuario" permita registrar correctamente un usuario con todos los campos obligatorios, y que se muestre en la tabla sin recargar la vista.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El usuario actual debe tener rol de administrador. • La opción "Agregar nuevo usuario" debe estar visible en la interfaz.
Pasos para seguir	1. Iniciar sesión como administrador. 2.Acceda a la sección "Usuarios" desde el menú lateral. 3. Presionar el botón "Agregar nuevo usuario". 4. Llenar el formulario con la siguiente información: • Nombre • Correo electrónico • Rol (usuario o administrador) • Contraseña 5. Presionar el botón "Guardar". 6.Verifique que el usuario aparezca en la tabla. 7. Confirmar que se mostrará un mensaje de éxito.
Datos de Entrada	• Nombre: David Herrera • Correo electrónico: david.herrera@quality.com • Rol: Administrador • Contraseña: David123

Resultados Esperados	• Todos los campos son validados correctamente. • El sistema muestra un mensaje indicando que el usuario fue registrado con éxito. • El nuevo usuario aparece en la tabla de usuarios sin necesidad de recargar la página.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El formulario valida correctamente los datos. • El usuario se registra exitosamente. • El mensaje de éxito se muestra. • El usuario nuevo se ve en la tabla de usuarios. **RECHAZO:** • Algún campo está vacío o incompleto. • El correo ya existe en el sistema. • La contraseña no cumple los requisitos mínimos. • El sistema muestra un mensaje de error o no responde.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 15: Crear nuevo usuario

3.1.5.2 Editar rol del usuario desde la tabla

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el rol del usuario pueda modificarse directamente desde el campo desplegable de la tabla, y que el nuevo rol se guarde correctamente.	
Precondiciones	• El usuario debe estar registrado. • La tabla debe mostrar el campo desplegable de rol.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ir a la sección “Usuarios”. 3. Localizar el usuario que se desea editar. 4. Cambiar el valor del campo “Rol” usando el select desplegable. 5. Confirmar visualmente que el rol se actualizó.
Datos de Entrada	• Nombre: David Herrera • Correo electrónico: david.herrera@quality.com • Rol: Usuario • Contraseña: David123
Resultados Esperados	• El nuevo rol se guarda automáticamente o tras confirmación. • El cambio se refleja en la tabla.

Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El select permite cambiar el rol. • El nuevo rol se actualiza correctamente. • El cambio es visible de inmediato. **RECHAZO:** • El select no responde. • El rol vuelve al valor anterior. • El sistema muestra un error.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 16: Editar rol del usuario desde la tabla

3.1.5.3 Inactivar usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el botón "Inactivar" en la tabla de usuarios cambie el estado del usuario a inactivo, impidiendo su ingreso al sistema.	
Precondiciones	• El usuario debe estar registrado. • El botón "Inactivar" debe estar visible.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ir a la sección "Usuarios". 3. Ubicar el usuario activo. 4. Presionar el botón "Inactivar". Confirmar que el estado cambie a "Inactivo".
Datos de Entrada	• Nombre: David Herrera • Acción: Inactivar usuario
Resultados Esperados	• El estado cambia a "Inactivo" con indicador visual (color y texto). • El usuario ya no puede iniciar sesión.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El estado cambia a "Inactivo" con indicador visual (color o texto). • El usuario ya no puede iniciar sesión. • Se muestra mensaje de confirmación (opcional). **RECHAZO:** • El estado no cambia. • El usuario sigue iniciando sesión. • No hay retroalimentación visual.

Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 17: Inactivar usuario

3.1.6 Gestión de Productos

3.1.6.1 Registro de producto en inventario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita registrar correctamente un nuevo producto en el inventario, validando todos los campos requeridos y mostrando un mensaje de confirmación. El producto debe aparecer en la tabla sin errores.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El usuario debe haber iniciado sesión y tener permisos para gestionar productos. • Deben existir al menos una categoría y un proveedor previamente registrados.
Pasos para seguir	1.Iniciar sesión como usuario con permisos de inventario. 2. Ingresar a la sección “Inventario” desde el menú lateral. 3.Presionar el botón “Agregar nuevo producto”. 4.Completar los siguientes campos en el formulario: • Nombre del producto • Categoría • Proveedor • Stock mínimo • Stock actual 6.Presionar el botón “Guardar”. 7.Verifique que el producto se muestre en la tabla de productos. 8.Confirmar que el sistema muestre un mensaje de éxito.
Datos de Entrada	• Nombre del producto: ejemplo • Categoría: Tecnología • Proveedor: InforTech • Stock mínimo: 5 • Stock actual: 12
Resultados Esperados	• El sistema valida todos los campos. • Se guarda correctamente el producto en la base de datos. • Se muestra el producto en la tabla de inventario. • Se presenta un mensaje indicando que el producto fue registrado exitosamente.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Todos los campos son válidos. • El producto se registra correctamente.

	• El mensaje de éxito se muestra. • El producto aparece en la tabla. **RECHAZO:** • Faltan campos para completar. • El nombre del producto ya existe. • Se presenta un error del sistema al guardar. • No aparece el producto en la tabla.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 18: Registro de producto en inventario

3.1.6.2 Editar producto

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que se pueda modificar la información de un producto existente desde la tabla de inventario, y que los cambios se guarden correctamente y se reflejen en la interfaz.	
Precondiciones	• Debe existir al menos un producto registrado. • El usuario debe tener permisos para editar productos.
Pasos para seguir	1. Iniciar sesión como usuario autorizado. 2. Ingresar a la sección “Inventario”. 3. Localizar el producto a editar. 4. Modificar uno o varios campos del formulario. 5. Presionar el botón “Guardar cambios” (si existe) o confirmar la edición. 6. Verificar que los cambios se reflejen en la tabla. 7. Confirmar que se muestre un mensaje de éxito.
Datos de Entrada	• Nombre del producto: ejemplo editado • Categoría: Tecnología • Proveedor: InforTech • Stock mínimo: 100 • Stock actual: 80
Resultados Esperados	• El sistema valida todos los campos. • El sistema guarda los cambios. • El producto se actualiza en la tabla. • Se muestra un mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Los datos se modifican correctamente. • Se actualiza la información en la tabla. • Se muestra un mensaje de éxito. **RECHAZO:**

	• El sistema no guarda los cambios. • La tabla no se actualiza. • Se muestra un mensaje de error.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 19: Editar producto

3.1.6.3 Eliminar producto

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que se pueda eliminar un producto del inventario de forma lógica, marcándolo como eliminado sin borrarlo físicamente, y que deje de mostrarse en la tabla.	
Precondiciones	• Debe existir al menos un producto registrado. • El usuario debe tener permisos de eliminación.
Pasos para seguir	1. Iniciar sesión como usuario autorizado. 2. Ingresar a la sección “Inventario”. 3. Localizar el producto a eliminar. 4. Presionar el botón de eliminar (si aplica) o activar la opción “Eliminar”. 5. Confirmar la acción si se solicita. 6. Verificar que el producto ya no se visualice en la tabla. 7. Confirmar mensaje de eliminación exitosa.
Datos de Entrada	• Nombre del producto: ejemplo
Resultados Esperados	• El producto se marca como eliminado. • Ya no aparece en la tabla de inventario. • Se muestra un mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El producto se oculta correctamente. • Se confirma la eliminación con mensaje. • No se borra de la base de datos permanentemente. **RECHAZO:** • El producto sigue apareciendo. • Ocurre un error visual o de backend. • No se muestra mensaje de confirmación.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 20: Eliminar producto

3.1.7 Movimientos de Inventario

3.1.7.1 Registrar entrada

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita registrar una entrada (ingreso) de productos al inventario desde la interfaz, actualizando el stock del producto seleccionado y mostrando un mensaje de confirmación.	
Precondiciones	• El sistema debe estar funcionando correctamente. • Debe existir al menos un producto registrado. • Debe haber un proveedor asociado. • El usuario debe tener permisos para registrar movimientos.
Pasos para seguir	1. Iniciar sesión como usuario autorizado. 2. Ingresar a la sección "Movimientos". 3. Seleccionar tipo de movimiento: Entrada. 4. Seleccionar el producto desde el desplegable. 5. Ingresar la cantidad a ingresar. 6. Seleccionar el proveedor asociado. 7. Presionar el botón "Guardar" o "Registrar entrada". 8. Verificar que se muestre mensaje de éxito. 9. Confirmar que el stock del producto aumentó correctamente.
Datos de Entrada	• Tipo de movimiento: Entrada • Producto: Alcohol • Cantidad: 10 • Proveedor: InforTech
Resultados Esperados	• El movimiento se registra correctamente como entrada. • El stock del producto aumenta en la cantidad indicada. • Se muestra un mensaje de éxito. • El nuevo movimiento aparece en la tabla.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El sistema permite registrar el ingreso. • El stock se actualiza correctamente. • El movimiento aparece listado. • Se muestra mensaje de confirmación. **RECHAZO:** • El formulario no se valida correctamente. • El producto no se encuentra. • El stock no se actualiza. • No se muestra mensaje de éxito.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 21: Registrar entrada

3.1.7.2 Registrar salida

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita registrar una salida de productos desde el inventario, asociando un cliente y disminuyendo el stock del producto, siempre que exista suficiente cantidad disponible.	
Precondiciones	• Debe haber productos registrados con stock disponible. • Debe existir al menos un cliente registrado. • El usuario debe tener permisos para registrar movimientos
Pasos para seguir	1. Iniciar sesión como usuario autorizado. 2. Ingresar a la sección "Movimientos". 3. Seleccionar tipo de movimiento: Salida. 4. Elegir el producto desde el desplegable. 5. Ingresar la cantidad a retirar. 6. Seleccionar el cliente asociado a la salida. 7. Presionar el botón "Guardar" o "Registrar salida". 8. Verificar que se muestre mensaje de éxito. 9. Confirmar que el stock del producto se haya reducido.
Datos de Entrada	• Tipo de movimiento: Salida • Producto: Alcohol • Cantidad: 5 • Proveedor: InforTech
Resultados Esperados	• El movimiento se registra correctamente como salida. • El stock del producto disminuye. • Se muestra un mensaje de éxito. • El nuevo movimiento aparece en la tabla.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Se registra correctamente la salida. • El stock disminuye en la cantidad indicada. • El cliente queda asociado al movimiento. • Se muestra un mensaje de confirmación. **RECHAZO:** • La cantidad supera el stock disponible. • Faltan datos en el formulario. • El sistema no actualiza el stock. • El movimiento no se registra o da error.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 22: Registrar salida

3.1.8 Gestión de Proveedores
3.1.8.1 Crear proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita registrar un nuevo proveedor desde la interfaz, validando los campos obligatorios, y mostrando un mensaje de confirmación si la creación es exitosa.	
Precondiciones	• El sistema debe estar en funcionamiento. • El usuario debe haber iniciado sesión con permisos de administración. • El botón de "Añadir proveedor" debe estar disponible.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ingresar a la sección “Proveedores”. 3. Presionar el botón “Añadir proveedor”. 4. Llenar el formulario con los siguientes campos: a. Nombre b. Correo electrónico c. Cédula/RUC d. Teléfono e. Dirección 5. Presionar el botón “Guardar”. 6. Verificar que el proveedor aparezca en la tabla. 7. Confirmar que se muestre mensaje de éxito.
Datos de Entrada	• Nombre: InforTech • Correo electrónico: contacto@infortech.com • Cédula/RUC: 1790123456001 • Teléfono: 0999999999 • Dirección: Av. 10 de Agosto y Bogotá
Resultados Esperados	• Todos los campos se validan correctamente. • El proveedor se registra y aparece en la tabla. • Se muestra un mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • Todos los campos son válidos. • El proveedor se registra exitosamente. • Se muestra mensaje de éxito. • El nuevo proveedor se visualiza en la tabla. **RECHAZO:** • Todos los campos son válidos. • El proveedor se registra exitosamente. • Se muestra mensaje de éxito. • El nuevo proveedor se visualiza en la tabla.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 23: Crear proveedor

3.1.8.2 Editar proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que el sistema permita editar la información de un proveedor existente desde la interfaz, mostrando un mensaje de confirmación y actualizando los datos en la tabla.	
Precondiciones	• El proveedor debe estar previamente registrado. • El usuario debe tener permisos de edición.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ingresar a la sección "Proveedores". 3. Buscar al proveedor que se desea editar. 4. Presionar el botón de edición. 5. Modificar los campos necesarios en el formulario. 6. Guardar los cambios. 7. Verificar que la tabla se actualice con la nueva información. 8. Confirmar que se muestre mensaje de éxito.
Datos de Entrada	• Nombre: InforTech Ecuador • Correo electrónico: soporte@infortech.com • Cédula/RUC: 1790123456001 • Teléfono: 0987654321 • Dirección: Av. América
Resultados Esperados	• El sistema permite editar los campos. • Los cambios se guardan y se reflejan en la tabla. • Se muestra mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • **Los campos se editan correctamente.** • **La tabla refleja los cambios.** • **El mensaje de éxito aparece.** **RECHAZO:** • Los campos se editan correctamente. • La tabla refleja los cambios. • El mensaje de éxito aparece.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 24: Editar proveedor

3.1.8.3 Eliminar proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita eliminar lógicamente un proveedor desde la tabla, ocultándolo de la vista sin borrarlo físicamente de la base de datos.	
Precondiciones	• El proveedor debe estar registrado. • El usuario debe tener permisos de eliminación.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ir a la sección "Proveedores". 3. Buscar al proveedor que se desea eliminar. 4. Presionar el botón de eliminar o inactivar. 5. Confirmar la acción si se solicita. 6. Verificar que el proveedor desaparezca de la tabla. 7. Confirmar que se muestre mensaje de confirmación.
Datos de Entrada	• Proveedor: InforTech
Resultados Esperados	• El proveedor se marca como eliminado. • Ya no se muestra en la tabla. • Se muestra un mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El proveedor se oculta correctamente. • Se muestra mensaje de éxito. • No se elimina de la base de datos físicamente. **RECHAZO:** • El proveedor sigue visible. • No se muestra mensaje. • Ocurre un error visual o técnico.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 25: Eliminar proveedor

3.1.9 Gestión de Clientes

3.1.9.1 Crear cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita registrar un nuevo cliente desde el formulario, validando los campos obligatorios y mostrando un mensaje de confirmación al guardarlo. El cliente debe aparecer en la tabla sin necesidad de recargar la página.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El usuario debe haber iniciado sesión con permisos de administración. • Debe estar visible el botón para agregar cliente.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ingresar a la sección “Clientes”. 3. Presionar el botón para añadir cliente. 4. Completar los campos del formulario: nombre, correo, cédula o RUC, teléfono y dirección. 5. Presionar el botón para guardar. 6. Verificar que el cliente aparezca en la tabla. 7. Confirmar que se muestre un mensaje de éxito.
Datos de Entrada	• Nombre: Laura Pérez • Correo: laura.perez@quality.com • Cédula/RUC: 1752201234 • Teléfono: 0988776655 • Dirección: Av. Amazonas y Rumipamba
Resultados Esperados	• El cliente se registra correctamente. • Los datos se validan y se muestra un mensaje de éxito. • El nuevo cliente aparece en la tabla.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El cliente se registra con éxito. • El mensaje de confirmación se muestra. • La información es visible en la tabla. **RECHAZO:** • Algún campo obligatorio no fue llenado. • El número de cédula no cumple con el formato. • El sistema muestra un error o el cliente no aparece.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 26: Crear cliente

3.1.9.2 Editar cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que el sistema permita modificar los datos de un cliente registrado, y que los cambios se actualicen correctamente en la tabla con un mensaje de confirmación.	
Precondiciones	• Debe existir al menos un cliente registrado. • El usuario debe tener permisos de edición.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ingresar a la sección "Clientes". 3. Buscar al cliente que se desea editar. 4. Presionar el botón de editar. 5. Modificar uno o más campos del formulario. 6. Guardar los cambios. 7. Verificar que los datos se actualicen en la tabla. 8. Confirmar que se muestre mensaje de éxito.
Datos de Entrada	• Nombre: Laura Pérez Mora • Correo: laura.mora@quality.com • Cédula/RUC: 1752201234 • Teléfono: 0998877665 • Dirección: Av. 10 de Agosto
Resultados Esperados	• Los cambios se guardan correctamente. • El cliente se actualiza en la tabla. • El sistema muestra mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El formulario permite editar los campos. • Los datos se actualizan correctamente. • Se muestra el mensaje de éxito. **RECHAZO:** • Los cambios no se guardan. • El sistema muestra un error. • La tabla no se actualiza.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 27: Editar cliente

3.1.9.3 Eliminar cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita eliminar lógicamente un cliente desde la tabla, ocultándolo de la vista sin borrarlo definitivamente de la base de datos.	
Precondiciones	• Debe existir al menos un cliente registrado. • El usuario debe tener permisos para eliminar registros.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Ingresar a la sección “Clientes”. 3. Buscar al cliente que se desea eliminar. 4. Presionar el botón de eliminar. 5. Confirmar la acción si se solicita. 6. Verificar que el cliente ya no se muestre en la tabla. 7. Confirmar que se muestre mensaje de éxito.
Datos de Entrada	• Cliente: Laura Pérez Mora
Resultados Esperados	• El cliente se marca como eliminado. • Ya no se muestra en la tabla. • Se presenta un mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El cliente se elimina correctamente. • El mensaje de éxito aparece. • El cliente desaparece de la tabla. **RECHAZO:** • El cliente sigue visible después de eliminarlo. • No se muestra ningún mensaje. • Ocurre un error visual o técnico.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 28: Eliminar cliente

3.1.10 Gestión de Alertas

3.1.10.1 Descartar alerta visualmente

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el usuario pueda descartar una alerta de bajo stock desde la interfaz, eliminándola de la vista sin que afecte la base de datos histórica.	
Precondiciones	• Debe existir al menos una alerta activa. • El usuario debe haber iniciado sesión.
Pasos para seguir	1. Iniciar sesión en el sistema. 2. Ir a la sección de alertas o dashboard. 3. Localizar una alerta de stock bajo. 4. Presionar el botón de “Descartar” o “Ocultar”.

	5. Confirmar que la alerta desaparezca de la vista.
Datos de Entrada	• No aplica. (La alerta ya debe existir en pantalla).
Resultados Esperados	• La alerta desaparece inmediatamente de la interfaz. • El sistema muestra un mensaje de confirmación (si está implementado).
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • La alerta se oculta correctamente de la vista. • La página no requiere recarga para actualizar. • Se muestra un mensaje de confirmación si aplica. **RECHAZO:** • La alerta sigue visible después de descartarla. • El sistema muestra error o no responde.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 29: Descartar alerta visualmente

3.1.11 Gestión de Cotizaciones

3.1.11.1 Crear cotización desde formulario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema permita crear una cotización desde el formulario, seleccionando cliente y productos, calculando valores automáticamente y mostrando un mensaje de confirmación.	
Precondiciones	• Debe haber clientes y productos registrados. • El usuario debe tener permisos para generar cotizaciones.
Pasos para seguir	1. Iniciar sesión en el sistema. 2. Ir a la sección “Cotizaciones”. 3. Presionar “Nueva Cotización”. 4. Seleccionar un cliente de la lista. 5. Agregar uno o más productos con sus cantidades. 6. Revisar el cálculo automático de subtotal, IVA e importe total. 7. Presionar “Guardar”. 8. Verificar que la cotización aparezca en la tabla. 9. Confirmar que se muestre mensaje de éxito.
Datos de Entrada	• Cliente: Alexandra Jiménez Producto 1: UTP– 10 unidades – $2.00 c/u Producto 2: Cable Coaxial – 50 unidades – $2.00 c/u Observaciones: Cotización válida por 30 días.
Resultados Esperados	• La alerta desaparece inmediatamente de la interfaz. • El sistema muestra un mensaje de confirmación (si está implementado).

Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • La cotización se registra sin errores. • Los totales calculados son correctos. • La tabla muestra el registro. **RECHAZO:** • Faltan datos obligatorios. • Los cálculos no se muestran o son incorrectos. • El sistema genera error o no muestra mensaje.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 30: Crear cotización desde formulario

3.1.12 Reportes PDF

3.1.12.1 Generar reporte de movimientos

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que el sistema genere correctamente un archivo PDF de los movimientos (entradas y salidas) desde la interfaz, permitiendo su descarga.	
Precondiciones	• Debe haber movimientos registrados. • El usuario debe tener permisos para generar reportes.
Pasos para seguir	1. Iniciar sesión en el sistema. 2. Ir a la sección “Reportes”. 3. Seleccionar el rango de fechas o tipo de reporte. 4. Presionar el botón “Generar PDF”. 5. Verificar que el sistema descargue el archivo PDF.
Datos de Entrada	• Rango de fechas: 01/07/2025 al 31/07/2025
Resultados Esperados	• El sistema genera un PDF con la información de los movimientos seleccionados. • El archivo se descarga sin errores. Los datos en el PDF corresponden al rango elegido.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: • El PDF se genera y descarga correctamente. • La información es precisa y actualizada. **RECHAZO:** • El archivo no se genera. • El PDF tiene datos incompletos o erróneos. • El sistema muestra un error al intentar exportar.
Desarrollador Asignado	Alison Lizbeth Carrion Hidalgo

Tabla 31: Generar reporte de movimientos

3.2 Pruebas y Estabilización Backend

3.2.1 Autenticación de Usuario

3.2.1.1 Login

DESCRIPCIÓN DEL CASO DE PRUEBA Probar si un usuario registrado puede iniciar sesión correctamente con sus credenciales.	
Precondiciones	• El usuario debe estar registrado y activo en el sistema.
Pasos para seguir	1. Enviar una solicitud POST a /login/ 2. Ingresar un email y contraseña válidos.

Datos de Entrada	{
	"email": "admin@example.com",
	"password": "1234"
	}

Resultados Esperados	• Se recibe los datos del usuario.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** El sistema devuelve status 200. **RECHAZO** El sistema devuelve status 401 si los datos son inválidos.
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 32: Login

3.2.1.2 Recuperación de contraseña

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que se envía un correo con enlace de recuperación al solicitarlo desde el sistema.	
Precondiciones	• El usuario debe estar registrado con un email válido.
Pasos para seguir	1. Enviar una solicitud POST a /forgot-password/ 2. Ingresar el email registrado.

Datos de Entrada	{
	"email": "admin@example.com"
	}

Resultados Esperados	• Se genera un token de recuperación y se envía un correo con el enlace.

Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Se envía el correo correctamente con status 200. **RECHAZO** Se recibe error si el email no está registrado.
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 33: Recuperación de contraseña

3.2.2 Gestión de usuarios

3.2.2.1 Crear nuevo usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que se puede registrar un nuevo usuario con rol asignado.	
Precondiciones	• El usuario actual debe tener permisos de administrador.
Pasos para seguir	1. Enviar una solicitud POST a /users/ 2. Ingresar datos requeridos para la creación.

Datos de Entrada	{
	"email": "usuario1@gmail.com",
	"name": "Usuario 1",
	"role": "Administrator",
	"phone": "1234567890",
	"is_active": true
	"password": "usuario1"
	}

Resultados Esperados	• Se crea el usuario y se retorna status 201 con los datos registrados.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** El usuario se crea exitosamente. **RECHAZO** Error 400 si faltan campos o email duplicado.
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 34: Crear nuevo usuario

3.2.2.2 Editar datos de usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Validar que se puedan actualizar los datos de un usuario existente.	
Precondiciones	• El usuario debe estar registrado y activo.

Pasos para seguir	1. Enviar una solicitud PATCH o PUT a /users/{id}/ 2. Modificar uno o más campos del usuario.

Datos de Entrada	{
	"email": "usuario1.modificado@gmail.com",
	"name": "Usuario 1 Modificado",
	"role": "Administrator",
	"phone": "0987654321",
	"is_active": true
	}

Resultados Esperados	• Se actualiza el nombre del usuario y se retorna status 200.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Se modifica correctamente el registro. **RECHAZO** Error 404 si el usuario no existe.
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 35: Editar datos de usuario

3.2.2.3 Cambiar contraseña

DESCRIPCIÓN DEL CASO DE PRUEBA Probar que un usuario autenticado puede cambiar su contraseña.	
Precondiciones	• El usuario debe estar autenticado con sesión activa.
Pasos para seguir	1. Enviar POST a /change-password/ 2. Ingresar contraseña actual y nueva.

Datos de Entrada	{
	"old_password": "usuario1",
	"new_password": "usuario1modificado"
	}

Resultados Esperados	• Se cambia la contraseña y se devuelve mensaje de confirmación.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Contraseña cambiada correctamente. **RECHAZO** Error si la contraseña actual es incorrecta.

Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 36: Cambiar contraseña

3.2.2.4 Inactivar Usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Verificar que un administrador puede desactivar un usuario.	
Precondiciones	• Debe existir un usuario activo y tener permisos de administrador.
Pasos para seguir	1. Enviar PATCH a /users/{id}/ 2. Establecer el campo is_active en false.

Datos de Entrada	{
	"is_active": false
	}

Resultados Esperados	• El usuario se marca como inactivo y no podrá iniciar sesión.
Criterios de Aceptación / Rechazo	**ACEPTACIÓN**: Status 200 y confirmación de desactivación. **RECHAZO**: Error 403 si no tiene permisos.
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 37: Inactivar Usuario

3.2.3 Gestión de productos

3.2.3.1 Registrar producto

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se pueda registrar un nuevo producto con su categoría y proveedor.	
Precondiciones	• Debe existir al menos una categoría y un proveedor.
Pasos para seguir	1. Enviar POST a /products/ 2. Ingresar datos requeridos para la creación.
Datos de Entrada	{ "name": "Producto 1", "description": "Descripcion de Producto 1", "price": "1200.00", "minimum_stock": 5, "status": "Activo", "image": null, "category": 1, "supplier": 1

	}
Resultados Esperados	• El producto se registra y devuelve status 201
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Producto registrado correctamente **RECHAZO** Faltan campos requeridos
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 38: Registrar producto

3.2.3.2 Editar producto

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se pueda modificar la información de un producto.	
Precondiciones	• El producto debe existir.
Pasos para seguir	1. Enviar PATCH o PUT a /products/{id}/ 2. Cambiar algún campo.

Datos de Entrada	{
	"name": "Producto 1 Modificado",
	"description": "Descripcion de Producto 1 Modificado",
	"price": "1400.00",
	"minimum_stock": 3,
	"status": "Activo",
	"image": null,
	"category": 2,
	"supplier": 2
	}

Resultados Esperados	• El producto se actualiza correctamente.
Criterios de Aceptación / Rechazo	**Aceptación** Status 200 y cambio visible **Rechazo** Producto no existe o datos inválidos
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 39: Editar producto

3.2.3.3 Eliminar producto

DESCRIPCIÓN DEL CASO DE PRUEBA Elimina lógicamente el producto (soft delete).	
Precondiciones	• El producto debe estar activo.
Pasos para seguir	1. Enviar PATCH a /products/{id}/ 2. Confirmar que ya no aparece

Datos de Entrada	{
	"deleted_at": "2025-07-17T23:55:00-05:00"
	}

Resultados Esperados	• El campo deleted_at se actualiza.
Criterios de Aceptación / Rechazo	**Aceptación** Producto ocultado **Rechazo** ID no encontrado
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 40: Eliminar producto

3.2.4 Gestión de movimientos

3.2.4.1 Registrar entrada de producto

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se registre una entrada y se actualice el stock del producto.	
Precondiciones	• Debe existir al menos un producto registrado.
Pasos para seguir	1. Enviar POST a /movements/ 2. Ingresar tipo entrada, cantidad, producto

Datos de Entrada	{
	"movement_type": "input",
	"date": "2025-07-17T07:52:00-05:00",
	"quantity": 10,
	"product": 1,
	"product_name": "Producto 1",
	"product_stock": 10,
	"supplier_name": "Proveedor 1",
	"customer": null,
	"customer_name": "",
	"user_name": "Usuario 1"
	}

Resultados Esperados	• Se registra la entrada y se incrementa el stock
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Entrada registrada y stock actualizado **RECHAZO** Producto inexistente
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 41: Registrar entrada de producto

3.2.4.2 Registrar salida de producto

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se registre una salida si hay suficiente stock.	
Precondiciones	• Debe haber stock suficiente del producto.
Pasos para seguir	1. Enviar POST a /movements/ 2. Ingresar tipo salida, cantidad, producto y cliente

Datos de Entrada	{
	"movement_type": "output",
	"date": "2025-07-17T07:52:00-05:00",
	"quantity": 1,
	"product": 1,
	"product_name": "Producto 1",
	"product_stock": 9,
	"supplier_name": "Proveedor 1",
	"customer": 1,
	"customer_name": "Cliente 1",
	"user_name": "Usuario 1"
	}

Resultados Esperados	• Se registra la salida y disminuye el stock
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Salida registrada correctamente **RECHAZO** Stock insuficiente
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 42: Registrar salida de producto

3.2.5 Gestión de proveedores

3.2.5.1 Crear proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se pueda registrar un nuevo proveedor.	
Precondiciones	• Debe estar autenticado como administrador.
Pasos para seguir	1. Enviar POST a /suppliers/ 2. Ingresar datos requeridos para la creación.

Datos de Entrada	{
	"name": "Proveedor 1",
	"email": "proveedor1@gmail.com",
	"tax_id": "1752213023001",
	"phone": "1234567890",
	"address": "El Ejido"
	}

Resultados Esperados	• Proveedor registrado correctamente con status 201
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Proveedor creado **RECHAZO** Nombre duplicado o datos faltantes
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 43: Crear proveedor

3.2.5.2 Editar proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Permite modificar el nombre o contacto del proveedor.	
Precondiciones	• El proveedor debe estar registrado.
Pasos para seguir	1. Enviar PATCH o PUT a /suppliers/{id}/ 2. Modificar campos.

Datos de Entrada	{
	"name": "Proveedor 1 Modificado",
	"email": "proveedor1.modificado@gmail.com",
	"tax_id": "3203122571001",
	"phone": "0987654321",
	"address": "El Ejido"
	}

Resultados Esperados	• El proveedor se actualiza
Criterios de Aceptación / Rechazo	**Aceptación**: status 200 y datos modificados **Rechazo**: proveedor no existe
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 44: Editar proveedor

3.2.5.3 Eliminar proveedor

DESCRIPCIÓN DEL CASO DE PRUEBA Elimina lógicamente un proveedor (soft delete).	
Precondiciones	• Proveedor existente
Pasos para seguir	1. Enviar PATCH a /suppliers/{id}/ 2. Confirmar que ya no aparece

Datos de Entrada	{
	"deleted_at": "2025-07-17T23:55:00-05:00"
	}

Resultados Esperados	• El campo deleted_at se actualiza.

Criterios de Aceptación / Rechazo	**Aceptación** Proveedor ocultado **Rechazo** ID no encontrado
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 45: Eliminar proveedor

3.2.6 Gestión de Clientes

3.2.6.1 Crear cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Verifica que se pueda registrar un nuevo cliente.	
Precondiciones	• Debe estar autenticado.
Pasos para seguir	1. Enviar POST a /customers/ 2. Ingresar datos requeridos para la creación.

Datos de Entrada	{
	"name": "Cliente 1",
	"email": "cliente1@gmail.com",
	"document": "1752213023",
	"phone": "1234567890",
	"address": "El Ejido"
	}

Resultados Esperados	• Se registra el cliente con status 201
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Cliente registrado **RECHAZO** Datos incompletos
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 46: Crear cliente

3.2.6.1 Editar cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Edita el nombre o contacto de un cliente	
Precondiciones	• Cliente registrado
Pasos para seguir	1. Enviar PATCH a /customers/{id}/ 2. Modificar campos
Datos de Entrada	{ "name": "Cliente 1 Modificado", "email": "cliente1.modificado@gmail.com", "document": "3203122571", "phone": "0987654321",

	"address": "El Ejido" }
Resultados Esperados	• Cliente modificado
Criterios de Aceptación / Rechazo	**Aceptación** Datos actualizados correctamente **Rechazo** Error de validación o cliente no existe
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 47: Editar cliente

3.2.6.1 Eliminar cliente

DESCRIPCIÓN DEL CASO DE PRUEBA Elimina lógicamente el cliente (soft delete).	
Precondiciones	• Cliente existente
Pasos para seguir	1. Enviar PATCH a /customers/{id}/ 2. Confirmar que ya no aparece

Datos de Entrada	{
	"deleted_at": "2025-07-17T23:55:00-05:00"
	}

Resultados Esperados	• Cliente ya no visible
Criterios de Aceptación / Rechazo	**Aceptación** Cliente ocultado **Rechazo** ID no encontrado
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 48: Eliminar cliente

3.2.7 Gestión de categorías

3.2.7.1 Registrar categoría

DESCRIPCIÓN DEL CASO DE PRUEBA Permite registrar una nueva categoría de productos.	
Precondiciones	• Debe estar autenticado.
Pasos para seguir	1. Enviar POST a /categories/ 2. Ingresar nombre de categoría

Datos de Entrada	{
	"name": "Categoría 1"
	}

Resultados Esperados	• Se registra la categoría con éxito
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Categoría creada **RECHAZO** Nombre duplicado
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 49: Registrar categoría

3.2.8 Gestión de alertas

3.2.8.1 Descartar alerta de stock

DESCRIPCIÓN DEL CASO DE PRUEBA Permite ocultar una alerta de bajo stock al usuario.	
Precondiciones	• Debe existir una alerta activa.
Pasos para seguir	1. Enviar PATCH a /alerts/{id}/dismiss/ 2. Confirmar que se oculta
Datos de Entrada	(No se requiere body, solo método PATCH)
Resultados Esperados	• La alerta se marca como eliminada lógicamente
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Status 200 y mensaje de éxito **RECHAZO** Status 404 si la alerta no existe
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 50: Descartar alerta de stock

3.2.9 Gestión de cotización

3.2.9.1 Crear cotización

DESCRIPCIÓN DEL CASO DE PRUEBA Generar una cotización con productos, cantidades, precio y notas.	
Precondiciones	• Debe existir cliente y productos registrados.
Pasos para seguir	1. Enviar POST a /quotations/create/ 2. Ingresar productos, precios y cliente
Datos de Entrada	{ "date": "2025-07-17", "subtotal": "1200.00", "vat": "180.00", "total": "1380.00", "observations": "Envio en 24 horas.",

	"customer": 1,
	"user": 1,
	"quoted_products": [
	{
	"product": 1,
	"quantity": 1,
	"unit_price": "1200.00",
	"subtotal": "1200.00"
	}
	]
	}

Resultados Esperados	• La cotización se guarda correctamente y se genera PDF
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Cotización registrada **RECHAZO** Error si faltan datos o IDs inválidos
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 51: Crear cotización

3.2.10 Reportes PDF

3.2.10.1 Generar reporte de movimientos por fecha

DESCRIPCIÓN DEL CASO DE PRUEBA Genera un PDF con los movimientos filtrados por rango de fechas.	
Precondiciones	• Debe haber movimientos registrados.
Pasos para seguir	1. Enviar POST a /reports/generate/ 2. Ingresar tipo de reporte y rango de fechas
Datos de Entrada	(No se requiere body, solo método POST)
Resultados Esperados	• Se genera el archivo PDF y se guarda en historial
Criterios de Aceptación / Rechazo	**ACEPTACIÓN** Se genera y devuelve el PDF **RECHAZO** No hay movimientos en el rango
Desarrollador Asignado	Alexander Francisco Pavón Álvarez

Tabla 52: Generar reporte de movimientos por fecha

3.3 Despliegue Base de datos

Instalación en entorno local

Durante el desarrollo del sistema, se utilizó PostgreSQL como sistema gestor de base de datos. Para ejecutar la aplicación en un entorno local, es necesario instalar PostgreSQL y configurar una base con los mismos parámetros usados en producción.

Pasos para entorno local:

1. Instalar PostgreSQL.
2. Crear una base de datos con el nombre qualitycore_inventory.
3. Crear un usuario con permisos y contraseña segura.
4. Configurar el archivo .env en el backend con los datos de conexión:
    - DB_NAME=qualitycore_inventory
    - DB_USER=postgres
    - DB_PASSWORD=admin
    - DB_HOST=localhost
    - DB_PORT=5432

Despliegue en producción

La base de datos fue desplegada en la nube utilizando el servicio Amazon RDS (Relational Database Service), con el motor PostgreSQL. La base se encuentra alojada en la región us-east-2 (Ohio) y fue configurada para conectarse directamente con el backend desplegado en Railway.

Detalles técnicos:

- Nombre del identificador de la base: inventorydb

- Motor utilizado: PostgreSQL
- Región: us-east-2 (Estados Unidos - Ohio)
- Instancia: db.t4g.micro
- Almacenamiento escalable: configurado automáticamente
- Red (VPC): configurada para aceptar conexiones desde Railway
- Estado actual: Disponible (Available)
- Conexión: segura mediante variables de entorno en el backend

Conexión desde el backend:

En Railway se configuraron las variables de entorno utilizando los datos de conexión generados por AWS, como el host, puerto, nombre de base de datos, usuario y contraseña.

Esto permitió que la aplicación en producción tenga acceso a una base de datos estable, escalable y administrada por Amazon Web Services.

3.4 Despliegue backend

Instalación en entorno local

Para el desarrollo del sistema backend, se utilizó el framework Django junto con Django REST Framework, permitiendo crear una API robusta, segura y escalable.

Requisitos previos:

- Python 3.11 o superior
- Pip (gestor de paquetes de Python)
- PostgreSQL (instalado localmente o en contenedor)
- Editor de código como Visual Studio Code

Pasos para ejecutar el backend en entorno local:

- Clonar el repositorio del backend desde GitHub.
- Crear un entorno virtual: python -m venv env

Activar el entorno:

- En Windows: env\Scripts\activate
- En Linux/Mac: source env/bin/activate

Instalar las dependencias:

- pip install -r requirements.txt

Configurar el archivo .env con los datos de conexión a la base de datos.

Ejecutar migraciones:

- python manage.py migrate

Ejecutar el servidor:

- python manage.py runserver

Esto habilita la API REST en la dirección:

http://localhost:8000/api/

Despliegue en producción con Railway

El backend fue desplegado utilizando la plataforma Railway, que permite ejecutar proyectos en la nube de manera sencilla, especialmente aquellos desarrollados con Django.

Pasos realizados para el despliegue:

1. Se subió el código fuente del backend a un repositorio de GitHub.
2. Se conectó el repositorio a Railway mediante la opción "Deploy from GitHub".
3. Railway detectó automáticamente el entorno Python y permitió configurar las variables de entorno necesarias, tales como:
    - DEBUG=False
    - SECRET_KEY=**********
    - DB_NAME=inventorydb
    - DB_USER=**********
    - DB_PASSWORD=**********
    - DB_HOST=********.rds.amazonaws.com
    - DB_PORT=5432
    - ALLOWED_HOSTS=qualitycore-inventory-backend-production.up.railway.app
4. Railway ejecutó automáticamente los comandos necesarios para el despliegue:
    - Instalación de dependencias
    - Migraciones (python manage.py migrate)
    - Recolección de archivos estáticos (python manage.py collectstatic)
5. Se habilitó la política CORS para permitir el acceso desde el frontend desplegado en Vercel.

Resultado final del despliegue:

El backend quedó disponible públicamente a través de la siguiente URL:

https://qualitycore-inventory-backend-production.up.railway.app/api/productos

Desde esta API se gestionan las funcionalidades principales del sistema, como usuarios, productos, movimientos, reportes, cotizaciones, alertas, entre otros.

3.5 Despliegue Frontend

Instalación en entorno local

El frontend del sistema fue desarrollado con React, una biblioteca de JavaScript que permite crear interfaces interactivas y modernas. Para poder ejecutar el sistema en una computadora local, es necesario tener algunas herramientas instaladas.

Requisitos previos:

- Tener instalado Node.js (versión 18 o superior)
- Tener instalado un editor de código como Visual Studio Code

Pasos para ejecutar el frontend localmente:

1. Clonar el repositorio del proyecto desde GitHub.
2. Abrir la carpeta del proyecto con el editor.
3. Abrir la terminal en la carpeta del proyecto.
4. Instalar las dependencias necesarias ejecutando: npm install
5. Iniciar el servidor de desarrollo con el comando: npm start
6. El sistema se abrirá automáticamente en el navegador o se podrá acceder desde la dirección: http://localhost:3000

De esta forma, se puede trabajar y probar el sistema localmente antes de desplegarlo en producción.

Despliegue en producción del frontend con Railway

Para que los usuarios puedan acceder al sistema desde cualquier parte, se utilizó Railway para el despliegue del frontend. Esta plataforma permite publicar proyectos hechos en React de manera rápida y con integración directa a GitHub.

Pasos realizados para el despliegue:

1. Se subió el repositorio del frontend a GitHub.
2. En Railway se seleccionó la opción “New Project” y se conectó la cuenta de GitHub.
3. Se eligió el repositorio del frontend del sistema.
4. Railway detectó automáticamente que se trata de una aplicación en React y configuró el entorno de ejecución.
5. Se configuró la variable de entorno API_URL con la dirección del backend en Railway: https://qualitycore-inventory-backend-production.up.railway.app/api/productos
6. Railway realizó todo el proceso de instalación, compilación y publicación del sitio web.
7. Enlace del frontend en producción: https://qualitycore-inventory-frontend-production.up.railway.app

Desde este enlace, los usuarios pueden acceder al sistema de gestión de inventarios desde cualquier navegador y dispositivo con conexión a internet.

Conclusiones

- **Desarrollo tecnológico alineado a las necesidades reales:**

El sistema de gestión de inventarios desarrollado responde efectivamente a las necesidades específicas de la empresa QualityCore Services SAS, permitiendo controlar el ingreso y salida de productos, gestionar proveedores y clientes, generar cotizaciones y reportes en PDF, así como emitir alertas automáticas por bajo stock.

- **Uso de tecnologías modernas y buenas prácticas de arquitectura:**

El proyecto fue implementado utilizando tecnologías modernas como Django + PostgreSQL en el backend y React.js en el frontend, aplicando arquitectura por capas y principios de desarrollo limpio. Esto garantizó la escalabilidad, mantenibilidad y claridad del código.

- **Mejora en la eficiencia de los procesos internos:**

Antes del sistema, los registros eran manuales, lo cual generaba errores y pérdida de tiempo. Con la implementación del sistema, se logró reducir significativamente los errores en la gestión de inventarios, mejorando la toma de decisiones mediante información precisa y en tiempo real.

- **Despliegue en la nube y accesibilidad remota:**

Gracias al uso de servicios como Railway (backend), Railway (frontend) y AWS RDS (base de datos), el sistema quedó desplegado en la nube, siendo accesible

desde cualquier dispositivo con conexión a internet, lo cual mejora la operatividad de la empresa.

- **Validación funcional mediante pruebas completas:**

Se llevaron a cabo pruebas funcionales detalladas del sistema, tanto en frontend como en backend, incluyendo casos de prueba por cada módulo implementado. Esto aseguró que todas las funcionalidades se comportaran conforme a los requerimientos establecidos.

- **Proceso de desarrollo guiado por metodologías ágiles:**

Durante el desarrollo del sistema se aplicaron principios de metodologías ágiles, lo que permitió dividir el proyecto en etapas manejables, realizar entregas parciales, y ajustar funcionalidades según el avance o retroalimentación obtenida, mejorando la organización del trabajo y el cumplimiento de los objetivos.

- **Fortalecimiento de habilidades técnicas del equipo desarrollador:**

El desarrollo de este sistema permitió al equipo aplicar y reforzar conocimientos en backend con Django, bases de datos relacionales, despliegue en la nube y desarrollo de interfaces modernas con React. Además, se adquirieron habilidades en documentación técnica, pruebas de software y control de versiones.

- **Adaptabilidad del sistema a otros contextos empresariales:**

Si bien el sistema fue desarrollado con base en las necesidades de QualityCore Services SAS, su estructura modular y flexible permite que sea fácilmente adaptado a otras pequeñas y medianas empresas con procesos similares de inventario, ventas o cotizaciones.

- **Contribución significativa al control y toma de decisiones:**

Con la automatización de procesos clave y la generación de reportes, el sistema se convierte en una herramienta de apoyo para la toma de decisiones estratégicas, facilitando el análisis de productos más vendidos, control de stock y rendimiento de proveedores.

Recomendaciones

- Se recomienda utilizar Django como framework principal para el desarrollo del backend, debido a que permite una estructura organizada, segura y con gran soporte para manejar autenticación, permisos, base de datos y lógica de negocio de forma integrada.
- Se recomienda el uso de PostgreSQL como sistema de gestión de bases de datos relacional, ya que proporciona estabilidad, rendimiento y características avanzadas como integridad referencial y soporte para grandes volúmenes de datos.
- Se recomienda utilizar Railway como plataforma para el despliegue del frontend, dado que permite publicar aplicaciones React de forma rápida y automática a partir de un repositorio en GitHub, facilitando el control de versiones y la actualización del sistema.
- Se recomienda implementar alertas automáticas para productos con stock bajo, dado que esta funcionalidad permite anticiparse a quiebres de inventario y tomar decisiones preventivas para la reposición de productos.
- Se recomienda realizar validaciones tanto en frontend como en backend al momento de registrar productos, movimientos, usuarios y

cotizaciones, con el fin de garantizar que los datos almacenados sean consistentes y eviten errores durante los procesos.

- Se recomienda mantener el sistema desplegado en la nube a través de Railway para garantizar accesibilidad remota desde cualquier dispositivo con internet, lo que mejora la operatividad y disponibilidad del sistema en todo momento.

Referencias bibliográficas

- Django Software Foundation. (2024). Django documentation (version 5.1). https://docs.djangoproject.com/
- React. (2024). React documentation. Meta Platforms, Inc. https://react.dev/
- PostgreSQL Global Development Group. (2024). PostgreSQL documentation (version 16). https://www.postgresql.org/docs/
- Railway. (2024). Railway documentation. Railway. https://docs.railway.app/
- Amazon Web Services. (2024). Amazon RDS for PostgreSQL user guide. AWS. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
- Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The definitive guide to Scrum. Scrum.org. https://scrumguides.org/
- Sommerville, I. (2011). Ingeniería del software (9.ª ed.). Pearson Educación.
- Pressman, R. S., & Maxim, B. R. (2020). Ingeniería de software: Un enfoque práctico (9.ª ed.). McGraw-Hill.