PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SISTEMA DE GESTIÓN DE INVENTARIO WEB PARA LA EMPRESA CHOCOROCKS

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autor: Jeremy Marin Lozano

Anghelo Alberto Montiel Sanchez

John Alexi San Martin Mera

Tutor: José Luis Galárraga Cañizares

Quito, Ecuador

2025

Tabla de contenidos

Lista de figuras.

DECLARACIÓN y AUTORIZACIÓN

Yo, **Anghelo Alberto Montiel Sanchez** con C.I. 171658716 autor(a) del trabajo de Titulación intitulado: **"Sistema de Inventario web para la empresa Chocorocks"**, previa a la obtención del título de Tecnología Superior en Desarrollo De Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro conocer la obligación de la Pontificia Universidad Católica del Ecuador, según el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del trabajo de graduación para integrarse 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, 27/08/2025

C.I. 171658716

DECLARACIÓN y AUTORIZACIÓN

Yo, **John Alexei San Martin Mera** con C.I. 1719420760 autor(a) del trabajo de Titulación intitulado: **"Sistema de Inventario web para la empresa Chocorocks"**, previa a la obtención del título de Tecnología Superior en Desarrollo De Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro conocer la obligación de la Pontificia Universidad Católica del Ecuador, según el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del trabajo de graduación para integrarse 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, 27/08/2025

C.I. 1719420760

DECLARACIÓN y AUTORIZACIÓN

Yo, **Jeremy Marin Lozano** con C.I. 1761430592 autor(a) del trabajo de Titulación intitulado: **"Sistema de Inventario web para la empresa Chocorocks"**, previa a la obtención del título de Tecnología Superior en Desarrollo De Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro conocer la obligación de la Pontificia Universidad Católica del Ecuador, según el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del trabajo de graduación para integrarse 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, 27/08/202

C.I. 1761430592

Agradecimientos

Dedico está tesis a mi mamá, por todo el apoyo que me dio a lo largo de mi carrera, mi mamá por ayudarme a salir adelante, gracias mamá, por todo el esfuerzo, todo los gastos, todos los regaños, todos los días que me decías que me esforzará, gracias por no dejarme rendir y estar ahí para mí, también dedico a mi hermana, porque sin ella quizás no tendría muchas razones para seguir superándome, porque quiero ser un buen ejemplo para ella, y al igual que mi mamá, ella ha estado conmigo en los momentos más precarios de mi vida, y eso es de agradecer de por vida, para mi familia, que están lejos, cuantas veces no amanecí, llorando, sufriendo extrañándolos a todos mientras hacia la tarea, que a pesar de la distancia sé que se sentirán orgullosos de mi al ver que logré graduarme, pese a la distancia que nos separa, sé que la lejanía es efímera, y que pronto estaremos juntos de nuevo, a mis amigos, Jeremy y John, sin ellos hubiese podido haber perdido el enfoque, quizás hasta darme de baja de la carrera, pero ellos estuvieron ahí como buenos amigos, sin duda son la definición de amistad y hermandad, gracias por estar ahí y por último, pero no menos importante dedicatoria a Dios, por permitirme Estudiar, por permitir entrar a la Universidad y seguir formando mi conocimientos y mi forma de ser, por permitir que me pudiera superar, Gloria a Dios.

Anghelo Montiel.

Introducción

En la era digital actual, el avance tecnológico se ha convertido en un factor determinante para la competitividad y sostenibilidad de las empresas, especialmente si estas empresas son pequeñas y medianas. La gestión eficiente de inventarios, ventas y procesos administrativos significa uno de los problemas más críticos que enfrentan las empresas en su búsqueda por optimizar recursos, reducir costos y mejorar la toma de decisiones.

La empresa Chocorocks, dedicada a la producción y comercialización de chocolates artesanales en Ecuador, padece esta problemática común en el sector de alimentos. Actualmente, la organización utiliza métodos manuales que incluyen el registro de inventarios en libretas físicas y control manual de ventas. Esta metodología, aunque era factible en sus etapas iniciales, representa un gran problema, mediante el cual la empresa ha crecido, y ya es imposible seguir ventas, inventario y hasta generación de recibos, esta complejidad radica en su modelo de negocio diversificado, que incluye tiendas físicas establecidas, puntos de venta móviles en ferias y eventos, y ventas al detalle y al por mayor.

El presente proyecto de tesis ofrece el desarrollo e implementación de un sistema web que aborde estos problemas mediante una solución tecnológica moderna, escalable y adaptada a las necesidades específicas de la industria de alimentos artesanales y del usuario final. La solución contempla funcionalidades avanzadas como operación en línea, interfaz responsive optimizada para dispositivos móviles, y arquitectura modular que permita futuras expansiones funcionales.

Justificación del Proyecto.

Este proyecto surge de la necesidad de modernizar los procesos operativos de la empresa Chocorocks mediante un sistema web eficiente, escalable y seguro. Actualmente, la empresa requiere soluciones tecnológicas que optimicen su gestión y permitan un crecimiento futuro sin complicaciones, la utilización de las herramientas de Spring Boot y Kotlin para el desarrollo del backend proporciona un framework empresarial con capacidades de gestión de datos, seguridad y arquitectura para microservicios, esto facilita el crecimiento y mantenibilidad del sistema.

La selección de Next.js para el desarrollo del frontend responde a la necesidad de crear interfaces de usuario intuitivas, modernas, responsivas y optimizadas para el rendimiento en dispositivos móviles. Esta tecnología permite la implementación de funcionalidades en línea, aspecto crítico para el funcionamiento en ferias y eventos donde no se tiene acceso a las computadoras de la empresa.

PostgreSQL se eligió sistema de base de datos porque garantiza seguridad, organización y crecimiento de la información de la empresa Chocorocks. Su estructura relacional permite manejar de manera eficiente las complejas relaciones entre productos, lotes, inventarios en diferentes ubicaciones y el seguimiento completo de cada ítem, algo esencial para el negocio.

Objetivo General.

Desarrollar e implementar un sistema web para automatizar el control de inventarios, ventas y recibos de Chocorocks. Permitirá manejar múltiples tiendas, rastrear productos desde su fabricación hasta la venta, y generar reportes para mejorar decisiones. Esto resolverá los problemas actuales de organización y ayudará al crecimiento del negocio.

Objetivos Específicos.

1. **Automatización del Inventario:**

- Reemplazar el actual sistema manual de registro en libretas por un sistema digital que permita un seguimiento preciso del stock de productos.
- Implementar un control visual del inventario que facilite la identificación rápida de existencias.
- Generar alertas automáticas cuando los productos alcancen niveles mínimos de stock.
- Implementar un sistema de seguimiento de lotes y fechas de distribución en cada tienda para garantizar trazabilidad completa.
- Generar códigos únicos por producto para facilitar la identificación y seguimiento en el sistema.

2. **Sistema de Ventas y reportaría:**
    - Registrar detalladamente las ventas incluyendo información completa sobre productos (tamaños, precios, sabores).
    - Implementar herramientas de análisis que permitan identificar los sabores más y menos vendidos.
    - Generar reportes visuales de ventas para facilitar la toma de decisiones.
    - Incorporar análisis detallado sobre qué productos específicos tienen mayor rotación en general y por punto de venta.
    - Implementar un sistema de ranking de tiendas según volumen de ventas para identificar puntos fuertes y débiles en la distribución.

3. **Gestión de Clientes:**
    - Crear un registro digitalizado de clientes para permitir al administrador ingresar los datos de estos, para generar recibos de manera sencilla y segura.
4. **Cálculo de Costos y Precios:**
    - Implementar un sistema para calcular y gestionar costos de producción por lote.
    - Incorporar campo específico para precios de venta a tiendas (precio mayorista), diferenciado del precio al consumidor final.

5. **Accesibilidad y Usabilidad:**

- Diseñar una interfaz responsiva que funcione óptimamente en dispositivos móviles como tablets para facilitar su uso en ferias y puntos de venta móviles.

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema

Propósito de la aplicación

Desarrollar una aplicación web integral para la empresa Chocorocks que permita la gestión automatizada de inventarios de chocolates y control de ventas, reemplazando el actual sistema manual que limita la capacidad de análisis e identificación de tendencias de ventas.

Alcance

El sistema abarcará la gestión de inventario, ventas, clientes y generación de recibos para la empresa Chocorocks, con enfoque en:

- Automatización del inventario y control de stock.
- Registro de ventas (por tienda y por cliente).
- Accesibilidad desde tablets para uso en ferias y puntos móviles.
- Generación de reportes de análisis y trazabilidad.

Exclusiones

1. Gestión Contable Avanzada
   - Integración con sistemas contables completos como SAP.
   - Cierres contables ni conciliaciones bancarias.

2. Pasarela de Pagos
   - Integración con plataformas como PayPhone, Stripe o PayPal.

3. Control de Producción Interna (Manufactura)
   - Módulo para la gestión de materias primas ni procesos de fabricación interna detallada.
   - Solo se registra el lote producido y su costo, no se automatiza la producción.

4. Gestión de Nómina o Recursos Humanos

   - Gestión de horarios, sueldos, asistencias ni perfiles de empleados.
   - Solo se manejarán roles para acceso al sistema (ej. Administrador y Vendedor).

5. Módulo de Devoluciones o Garantías

   - La gestión de devoluciones, productos defectuosos o garantías.

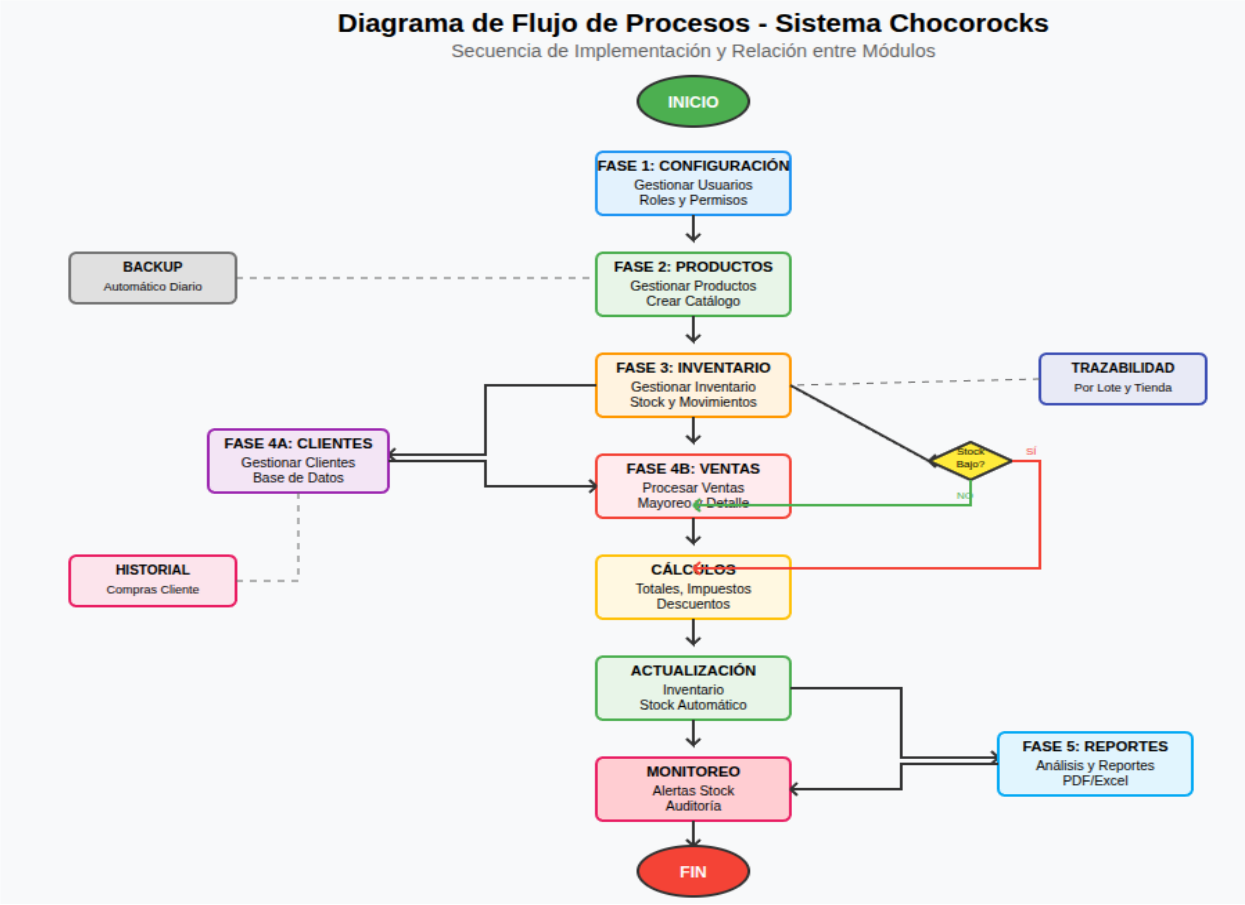
6. Multi-moneda o Multi-idioma
   - El sistema trabajará en una sola moneda (USD) y en idioma español únicamente.

7. App nativa para móviles
   - Sin inclusión de una aplicación móvil nativa (Android/iOS).
   - El sistema será web responsivo, accesible desde navegadores en tablets y teléfonos.

Modelado De Negocio.

Figura 1.

Diagrama de Flujo de Procesos.



Identificación de Roles.

Casos de uso

Figura 2.

1. **Gestionar producto**

- El sistema permite crear, modificar, eliminar y visualizar productos con información como: nombre, sabor, tamaño, precio, costo, imagen.

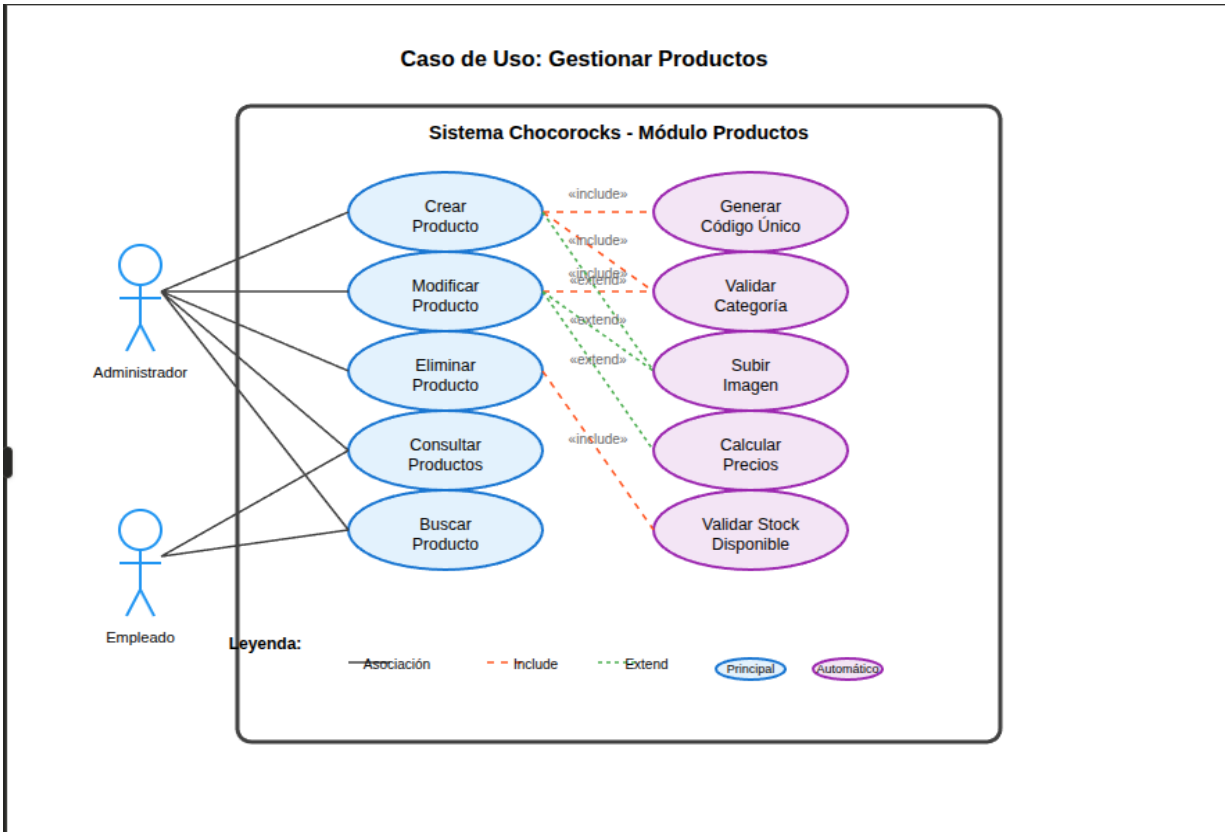


Figura 3.

2. **Gestionar inventario**

- Registrar ingreso, salida, y transferencias de stock.
- Alertas por bajo stock.
- Trazabilidad por lote y tienda.

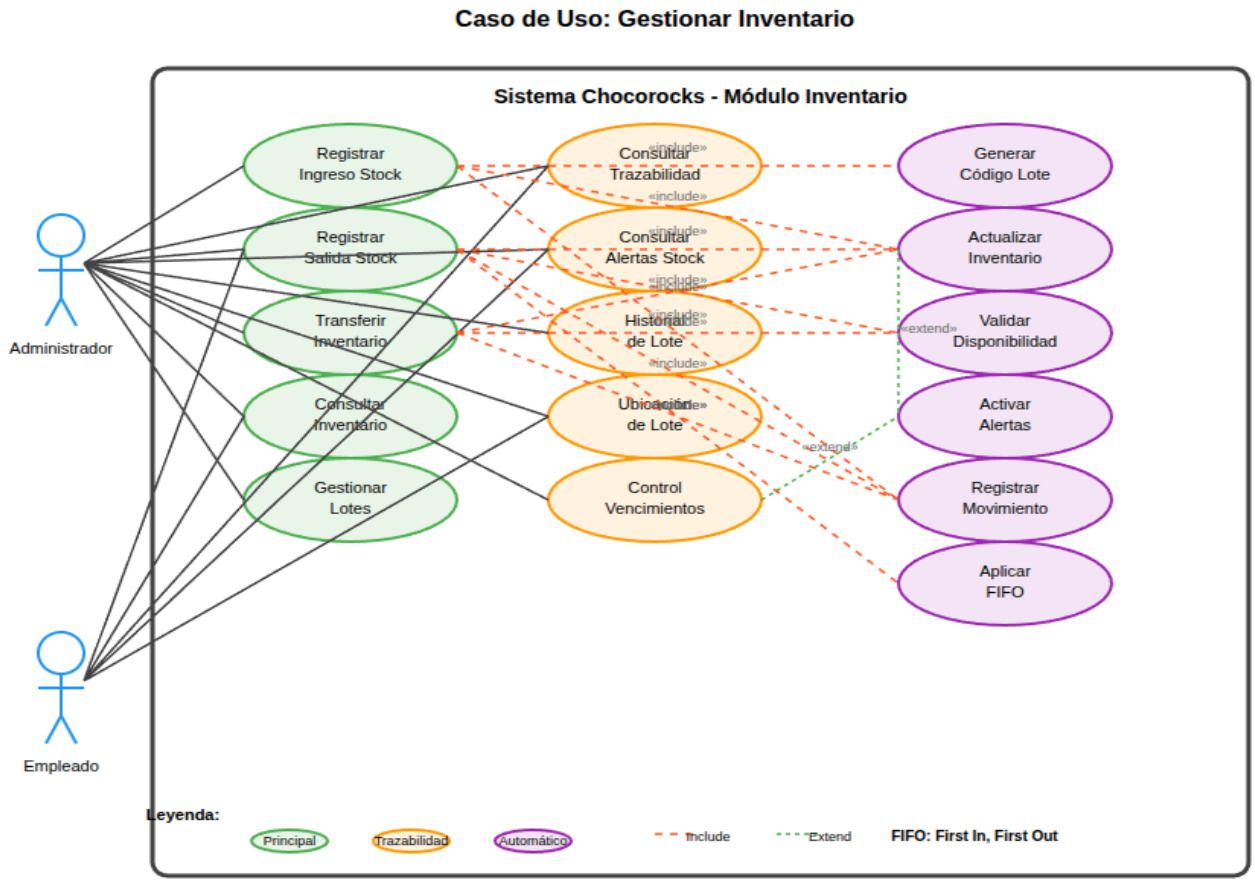


Figura 4.

3. Gestionar Reportería de Ventas

- Venta al por mayor y al detalle.
- Cálculo de totales, impuestos y descuentos.
- Historial de ventas por tienda y por producto.
- Productos más y menos vendidos.
- Exportación a Excel.
- Ranking de tiendas
- Productos más y menos vendidos.

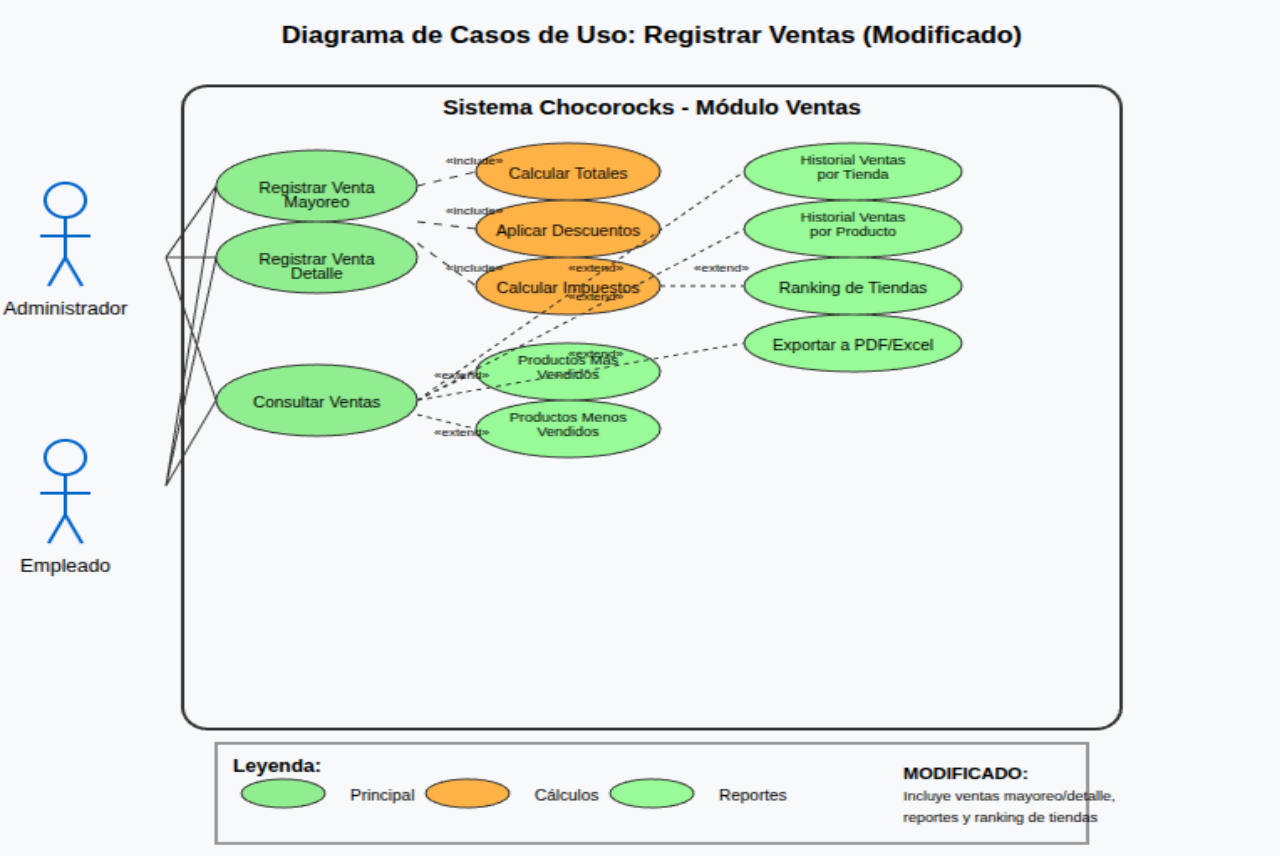


Figura 5.

4. **Gestionar clientes**

- Registro, edición y consulta de clientes.
- Historial de compras por cliente.

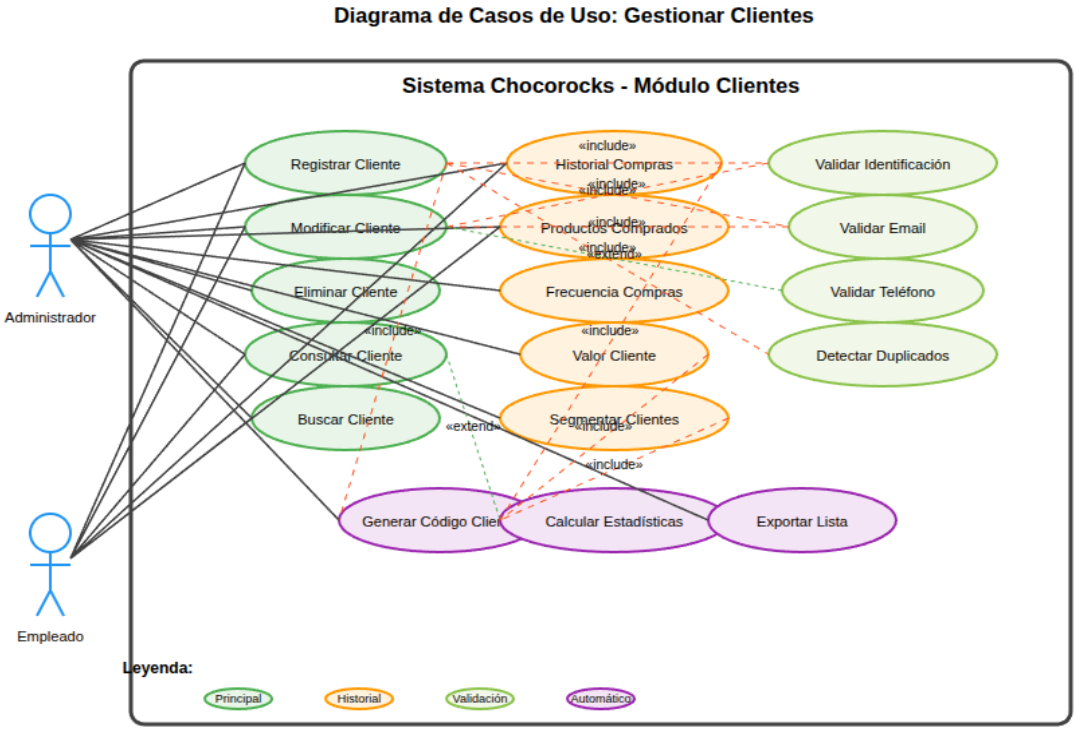
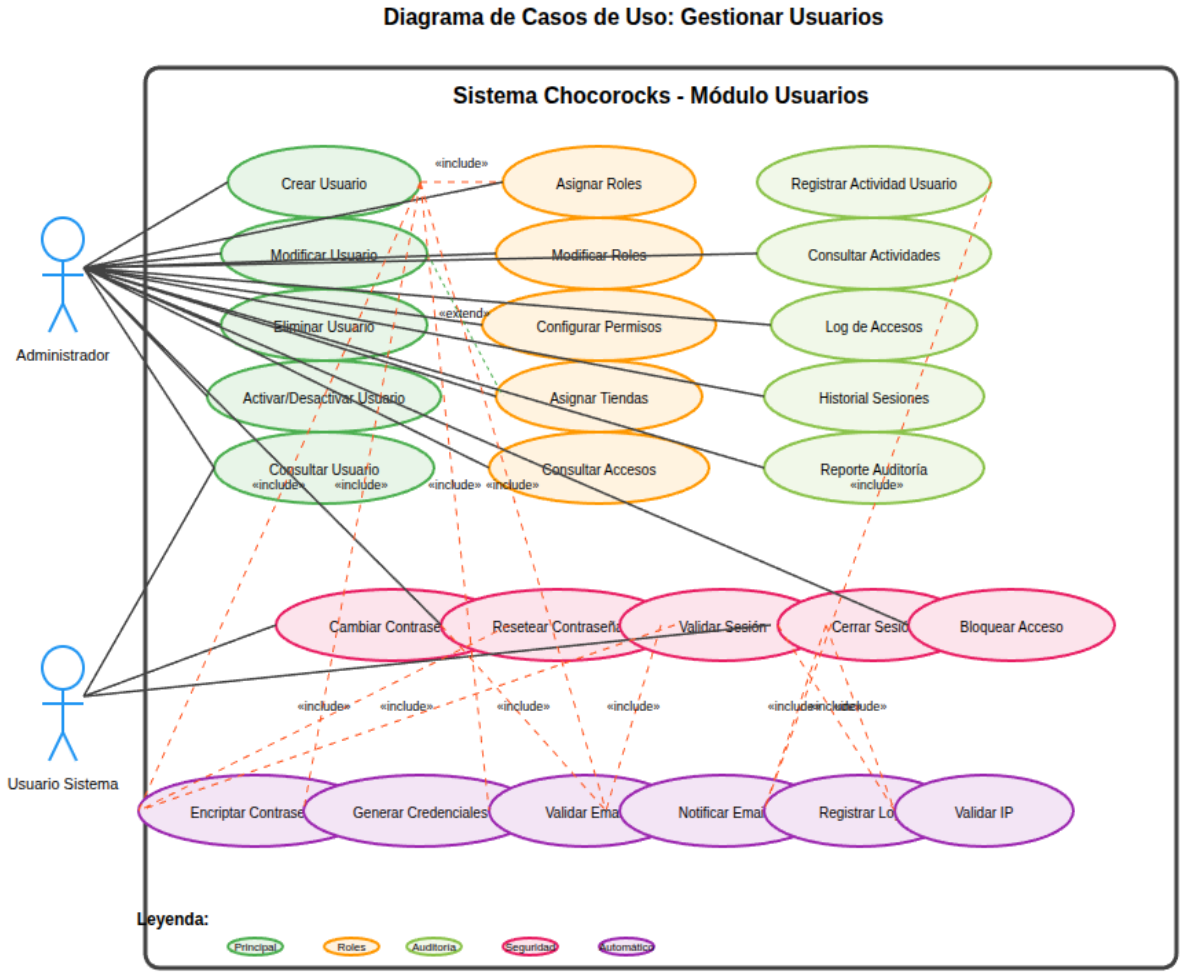


Figura 6.

5. **Gestionar usuarios.**

- Gestión de roles y niveles de acceso.
- Registro de actividades por usuario.



Actores del sistema

- Administradores
- Empleados

Requisitos Funcionales

1. Gestión de Productos

- 1.1. El sistema debe permitir crear, modificar, consultar y eliminar productos.
- 1.2. El sistema debe generar automáticamente un código único para cada producto registrado.
- 1.3. El sistema debe almacenar para cada producto: nombre, descripción, tipo/categoría, sabor, tamaño, costo de producción, precio de venta a tiendas, precio de venta al público, imagen del producto.
- 1.4. El sistema debe permitir agrupar los productos por categorías y sabores.

2. Gestión de Inventario

- 2.1. El sistema debe registrar y actualizar automáticamente las existencias de cada producto.
- 2.2. El sistema debe permitir el registro y seguimiento de lotes de producción, incluyendo fecha de elaboración, fecha de vencimiento y costo asociado.
- 2.3. El sistema debe generar alertas cuando los productos alcancen un nivel mínimo de stock definido por el usuario.
- 2.4. El sistema debe permitir la transferencia de inventario entre la bodega central y las diferentes tiendas o puntos de venta.
- 2.5. El sistema debe registrar y mantener actualizada la ubicación de cada lote (en qué tienda se encuentra).

3. Gestión de Ventas

- 3.1. El sistema debe permitir el registro de ventas, especificando productos, cantidades, precios y cliente.
- 3.2. El sistema debe calcular automáticamente los totales, subtotales, impuestos y descuentos aplicables.
- 3.3. El sistema debe permitir registrar ventas tanto al detalle (consumidor final) como al por mayor (tiendas).
- 3.4. El sistema debe permitir la consulta del historial de ventas por periodos, productos, tiendas y clientes.
- 3.5. El sistema debe generar reportes de los productos más vendidos y menos vendidos.
- 3.6. El sistema debe generar reportes de ranking de tiendas según volumen de ventas.

4. Gestión de Clientes

- 5.1. El sistema debe permitir el registro, modificación, consulta y eliminación de datos de clientes.
- 5.2. El sistema debe almacenar información básica de los clientes: nombre/razón social, identificación fiscal, dirección, teléfono, correo electrónico.
- 5.3. El sistema debe mantener un historial de compras por cliente.

5. Gestión de Tiendas/Puntos de Venta

- 6.1. El sistema debe permitir el registro, modificación y consulta de tiendas o puntos de venta.
- 6.2. El sistema debe mantener registro del inventario específico de cada tienda.
- 6.3. El sistema debe permitir la consulta del historial de ventas por tienda.

6. Análisis y Reportes

- 7.1. El sistema debe generar reportes de costos por lote, producto y periodo.
- 7.2. El sistema debe proporcionar informes de rentabilidad por producto, tienda y periodo.
- 7.3. El sistema debe generar dashboards visuales con indicadores clave de desempeño.
- 7.4. El sistema debe permitir la exportación de reportes en formatos PDF y Excel.
- 7.5. El sistema debe generar reportes de trazabilidad de lotes desde su producción hasta su venta.

7. Gestión de Usuarios

- 8.1. El sistema debe permitir la gestión de diferentes roles de usuario con distintos niveles de acceso.
- 8.2. El sistema debe registrar y permitir la consulta de logs de actividades de los usuarios.

Requisitos No Funcionales

1. Usabilidad

- 1.1. La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para personal con mínimas habilidades técnicas.
- 1.2. El sistema debe ser responsivo, adaptándose a diferentes tamaños de pantalla (computadoras, tablets).

2. Rendimiento

- 2.1. El sistema debe soportar al menos 10 usuarios concurrentes sin degradación del rendimiento.
- 2.2. El tiempo de respuesta para cualquier operación no debe exceder los 3 segundos bajo condiciones normales.
- 2.3. El sistema debe procesar al menos 201 transacciones diarias.

3. Disponibilidad y Confiabilidad

- 3.1. El sistema debe estar disponible al menos el 99% del tiempo en horario comercial.
- 3.2. El sistema debe incluir procedimientos de recuperación ante fallos.
- 3.3. El sistema debe mantener la integridad de los datos incluso en caso de fallas de conexión o energía.

4. Seguridad

- 4.1. El acceso al sistema debe estar protegido por autenticación de usuario con contraseña.
- 4.2. El sistema debe implementar cifrado para la transmisión de datos sensibles.

5. Interoperabilidad

- 5.1. El sistema debe permitir la integración con servicios de facturación electrónica del país.
- 5.2. El sistema debe permitir la exportación e importación de datos en formatos estándar (CSV, XML).

6. Escalabilidad

- 6.1. El sistema debe manejar un crecimiento de datos.
- 6.2. La arquitectura del sistema debe permitir la adición de nuevos módulos o funcionalidades sin afectar las existentes.

7. Mantenibilidad

- 8.1. El código fuente debe estar documentado.

8. Portabilidad

- 9.1. El sistema debe ser accesible desde los principales navegadores web (Chrome, Firefox, Edge, Safari) en sus versiones más recientes.
- 9.2. El sistema debe funcionar correctamente en sistemas operativos Windows, macOS y distribuciones comunes de Linux.

HISTORIAS DE USUARIOS

Módulo 1: Gestión de Productos

HU-001: Crear producto

Como administrador
Quiero crear nuevos productos en el sistema
Para mantener actualizado el catálogo de chocolates de Chocorocks

Criterios de Aceptación:

- El sistema debe generar automáticamente un código único para cada producto
- Debe permitir ingresar: nombre, descripción, tipo/categoría, sabor, tamaño, costo de producción, precio de venta a tiendas, precio de venta al público
- Debe permitir subir una imagen del producto
- El formulario debe validar que todos los campos obligatorios estén completos
- Debe mostrar mensaje de confirmación al crear exitosamente el producto

HU-002: Modificar producto

Como administrador
Quiero editar la información de productos existentes
Para mantener actualizada la información de precios, costos y características

Criterios de Aceptación:

- Debe mostrar todos los datos actuales del producto en el formulario de edición
- No debe permitir modificar el código único del producto
- Debe validar que los nuevos datos sean correctos antes de guardar
- Debe mantener historial de cambios realizados
- Debe mostrar mensaje de confirmación al actualizar exitosamente

HU-003: Consultar productos

Como empleado
Quiero buscar y consultar información de productos
Para conocer detalles, precios y disponibilidad durante las ventas

Criterios de Aceptación:

- Debe permitir búsqueda por código, nombre, sabor o categoría
- Debe mostrar listado paginado de productos
- Debe permitir filtrar por categoría y sabor
- Debe mostrar información completa al seleccionar un producto específico
- Debe ser accesible desde dispositivos móviles

HU-004: Eliminar producto

Como administrador
Quiero eliminar productos del catálogo
Para mantener solo productos activos en el sistema

Criterios de Aceptación:

- Solo debe permitir eliminar productos sin inventario asociado
- Debe solicitar confirmación antes de eliminar
- Debe mostrar alerta si el producto tiene stock o ventas asociadas
- Debe registrar la eliminación en el log de actividades

MÓDULO 2: GESTIÓN DE INVENTARIO

HU-005: Registrar ingreso de inventario

Como administrador
Quiero registrar nuevos lotes de producción en el inventario
Para mantener control actualizado del stock disponible

Criterios de Aceptación:

- Debe permitir seleccionar producto del catálogo existente
- Debe registrar: cantidad, fecha de elaboración, fecha de vencimiento, costo del lote
- Debe generar código único para el lote
- Debe actualizar automáticamente el stock total del producto
- Debe permitir asignar el lote a una tienda específica

HU-006: Registrar salida de inventario

Como empleado
Quiero registrar salidas de inventario por ventas o transferencias
Para mantener actualizado el stock real en cada tienda

Criterios de Aceptación:

- Debe permitir seleccionar producto y cantidad a descontar
- Debe validar que existe suficiente stock antes de procesar
- Debe especificar motivo de salida (venta, transferencia, merma)
- Debe actualizar automáticamente el inventario

HU-007: Transferir inventario entre tiendas

Como administrador
Quiero transferir productos entre tiendas
Para redistribuir inventario según demanda de cada punto de venta

Criterios de Aceptación:

- Debe permitir seleccionar tienda origen y destino
- Debe mostrar inventario disponible en tienda origen
- Debe validar stock suficiente antes de procesar transferencia

HU-008: Consultar alertas de stock bajo

Como administrador
Quiero recibir alertas cuando productos alcancen nivel mínimo
Para reabastecerlos antes de quedarse sin stock

Criterios de Aceptación:

- Debe permitir configurar nivel mínimo por producto
- Debe mostrar notificaciones visibles en el dashboard
- Debe mostrar listado de productos con stock bajo
- Debe indicar cantidad actual y mínima requerida

HU-009: Consultar trazabilidad de lotes

Como administrador
Quiero rastrear el histórico completo de un lote
Para garantizar la trazabilidad desde producción hasta venta

Criterios de Aceptación:

- Debe mostrar fecha de elaboración y vencimiento del lote
- Debe mostrar ventas realizadas de ese lote específico
- Debe permitir búsqueda por código de lote
- Debe generar reporte de trazabilidad exportable

MÓDULO 3: GESTIÓN DE VENTAS

HU-010: Registrar venta al detalle

Como empleado
Quiero registrar ventas directas a consumidores finales
Para llevar control de ingresos y descuentos de inventario

Criterios de Aceptación:

- Debe permitir agregar múltiples productos a la venta
- Debe calcular automáticamente subtotal, impuestos y total
- Debe permitir aplicar descuentos manuales
- Debe descontar automáticamente del inventario de la tienda
- Debe generar ticket o comprobante de venta

HU-011: Registrar venta al por mayor

Como administrador
Quiero registrar ventas a tiendas distribuidoras
Para controlar ventas mayoristas con precios especiales

Criterios de Aceptación:

- Debe aplicar automáticamente precios mayoristas
- Debe permitir ventas de grandes cantidades
- Debe validar stock suficiente

HU-012: Consultar historial de ventas

Como administrador
Quiero consultar el historial de ventas por diferentes filtros
Para analizar rendimiento y tendencias de ventas

Criterios de Aceptación:

- Debe permitir filtrar por fecha, tienda, producto y cliente

- Debe mostrar resumen de totales por período
- Debe permitir exportar resultados a Excel
- Debe mostrar gráficos de tendencias de ventas

HU-013: Aplicar descuentos y promociones

Como empleado
Quiero aplicar descuentos en las ventas
Para ofrecer promociones especiales a los clientes

Criterios de Aceptación:

- Debe permitir descuentos por porcentaje o monto fijo
- Debe validar que el descuento no exceda límites establecidos
- Debe mostrar precio original y precio final claramente

MÓDULO 4: GESTIÓN DE CLIENTES

HU-018: Registrar cliente

Como empleado
Quiero registrar nuevos clientes en el sistema
Para facilitar la facturación y generar historial de compras

Criterios de Aceptación:

- Debe capturar: nombre/razón social, cédula/RUC, dirección, teléfono, email
- Debe validar formato de cédula/RUC ecuatoriano
- Debe validar formato de email
- Debe verificar que no exista duplicado por identificación

HU-019: Modificar datos de cliente

Como empleado
Quiero actualizar información de clientes existentes
Para mantener datos de contacto actualizados

Criterios de Aceptación:

- Debe mostrar formulario prellenado con datos actuales
- Debe mantener historial de cambios realizados
- Debe validar nuevos datos antes de guardar
- No debe permitir cambiar número de identificación

HU-020: Consultar historial de cliente

Como administrador
Quiero ver el historial completo de compras de un cliente
Para brindar mejor servicio y analizar patrones de compra

Criterios de Aceptación:

- Debe mostrar todas las ventas del cliente ordenadas por fecha
- Debe mostrar monto total comprado históricamente

- Debe mostrar productos más comprados por el cliente

HU-021: Buscar cliente rápidamente

Como empleado
Quiero buscar clientes de forma rápida durante la venta
Para agilizar el proceso de facturación

Criterios de Aceptación:

- Debe permitir búsqueda por nombre, cédula o teléfono
- Debe mostrar resultados mientras se escribe (autocompletado)
- Debe mostrar información básica en los resultados
- Debe permitir seleccionar cliente con un clic

MÓDULO 5: GESTIÓN DE TIENDAS

HU-022: Registrar tienda

Como administrador
Quiero registrar nuevas tiendas o puntos de venta
Para llevar control independiente de inventario y ventas

Criterios de Aceptación:

- Debe capturar: nombre, dirección, responsable, teléfono
- Debe permitir definir si es tienda física o punto móvil

HU-023: Consultar inventario por tienda

Como administrador
Quiero ver el inventario específico de cada tienda
Para tomar decisiones de redistribución y reabastecimiento

Criterios de Aceptación:

- Debe mostrar stock actual de cada producto por tienda
- Debe indicar productos con stock bajo por tienda
- Debe mostrar valor total del inventario por tienda
- Debe permitir comparar inventarios entre tiendas
- Debe mostrar productos con mayor y menor rotación

HU-024: Consultar ventas por tienda

Como administrador
Quiero analizar el rendimiento de ventas por tienda
Para identificar puntos fuertes y oportunidades de mejora

Criterios de Aceptación:

- Debe mostrar ventas totales por período por tienda
- Debe generar ranking de tiendas por volumen de ventas
- Debe mostrar productos más vendidos por tienda
- Debe calcular promedio de venta diaria por tienda

MÓDULO 6: ANÁLISIS Y REPORTES

HU-025: Generar reporte de productos más vendidos

Como administrador
Quiero ver qué productos tienen mayor demanda
Para enfocar producción en sabores y tamaños exitosos

Criterios de Aceptación:

- Debe mostrar ranking de productos ordenado por cantidad vendida
- Debe permitir filtrar por período específico
- Debe mostrar tanto cantidad como ingresos por producto
- Debe separar análisis por tipo de venta (detalle vs mayoreo)

HU-026: Generar reporte de rentabilidad

Como administrador
Quiero analizar la rentabilidad por producto y tienda
Para optimizar la estrategia comercial y producción

Criterios de Aceptación:

- Debe calcular margen de ganancia por producto
- Debe mostrar rentabilidad por tienda
- Debe comparar costos vs ingresos por período
- Debe identificar productos menos rentables

HU-027: Generar dashboard ejecutivo

Como administrador
Quiero visualizar indicadores clave de desempeño
Para tomar decisiones estratégicas basadas en datos

Criterios de Aceptación:

- Debe mostrar ventas del día, semana y mes
- Debe mostrar indicadores de inventario (rotación, stock bajo)
- Debe actualizar automáticamente los datos
- Debe ser accesible desde dispositivos móviles

HU-028: Exportar reportes

Como administrador
Quiero exportar los reportes generados
Para compartir información con stakeholders externos

Criterios de Aceptación:

- Debe permitir exportar en formato Excel
- Debe generar archivo descargable inmediatamente

MÓDULO 7: GESTIÓN DE USUARIOS

HU-029: Crear usuario del sistema

Como administrador
Quiero crear cuentas de usuario para empleados
Para controlar el acceso y permisos en el sistema

Criterios de Aceptación:

- Debe definir rol (Administrador o Empleado)
- Debe asignar tienda específica a la que pertenece
- Debe permitir al administrador ingresar una contraseña para el usuario
- Debe validar que el email no esté registrado previamente

HU-030: Modificar permisos de usuario

Como administrador
Quiero cambiar las tiendas y desactivar usuarios existentes
Para ajustar accesos según responsabilidades actuales

Criterios de Aceptación:

- Debe permitir modificar tiendas asignadas
- Debe permitir activar/desactivar usuarios
- Debe registrar cambios en log de auditoría

HU-031: Consultar log de actividades

Como administrador
Quiero revisar las actividades realizadas por cada usuario
Para mantener control y auditoría del sistema

Criterios de Aceptación:

- Debe registrar todas las acciones importantes de cada usuario
- Debe mostrar fecha, hora, usuario y acción realizada
- Debe permitir filtrar por usuario, fecha y tipo de acción
- Debe ser información no modificable por ningún usuario

HISTORIAS TÉCNICAS

HU-032: Responsive design

Como usuario
Quiero que el sistema se adapte a diferentes dispositivos
Para usarlo eficientemente en tablets.

Arquitectura del Proyecto.

Backend: Arquitectura Por Capas.

Definición.

La arquitectura por capas es un modelo fundamental en el desarrollo de aplicaciones empresariales, ya que permite organizar el proyecto por capas horizontales, cada capa tiene responsabilidades específicas y solo pueden comunicarse con la capa inferior, esto lo que hace es crear una separación bastante clara de responsabilidades y flujo de datos unidireccional lo que facilita el testing, mantenimiento y escalabilidad del sistema.

Características Principales.

Comunicación Unidireccional:

- Cada capa solo puede comunicarse con la capa inferior.
- Nunca existe comunicación hacia las capas superiores.
- El flujo de datos es descendente y las respuestas son ascendente.

Separación de Responsabilidades:

- **Capa de Presentación:** Manejo de interfaces de usuario y protocolos de comunicación.
- **Capa de lógica de negocio:** Implementación de lógica del negocio y procesos empresariales.
- **Capa de acceso a datos:** Abstracción y manejo de operaciones de persistencia.
- **Capa de datos:** Almacenamiento y gestión de la información.

Encapsulación Por niveles:

- Cada capa encapsula su función específica.
- Los detalles internos de una capa están ocultos de las capas superiores.
- Se promueve el bajo acoplamiento y la alta cohesión.

Escalabilidad:

- Cada capa puede optimizarse independientemente.
- Permite distribución de capas en diferentes servidores.
- Facilita el mantenimiento y actualización sin afectar otras capas.

Frontend.

Arquitectura: Feature – Based Clean Architecture con BaaS (Backend-as-a-Service)

Está arquitectura es un modelo moderno de desarrollo de aplicaciones web donde el frontend se organiza en varias capas verticales por características de negocio, siguiendo los principios de Clean Architecture, mientras que el backend se externaliza por completo como un servicio que está siendo gestionado por la nube, este modelo, cada funcionalidad del sistema encapsula como un módulo independiente que contiene todos

los elementos necesarios como son los componentes, lógica, tipos y servicios para que puedan operar de forma autónoma, estas se comunican con el backend de manera remota a través de APIs REST o GraphQL.

Funcionamiento Fundamental.

El sistema se estructura en capas horizontales las cuales son presentación, aplicación, dominio e infraestructura que se interceptan con módulos verticales como la autenticación, gestión de usuarios, inventario, etc. Cada Feature contiene su propia implementación de estas capas, permitiendo que los equipos de desarrollo trabajen de forma independiente en diferentes partes del sistema sin interferir entre sí.

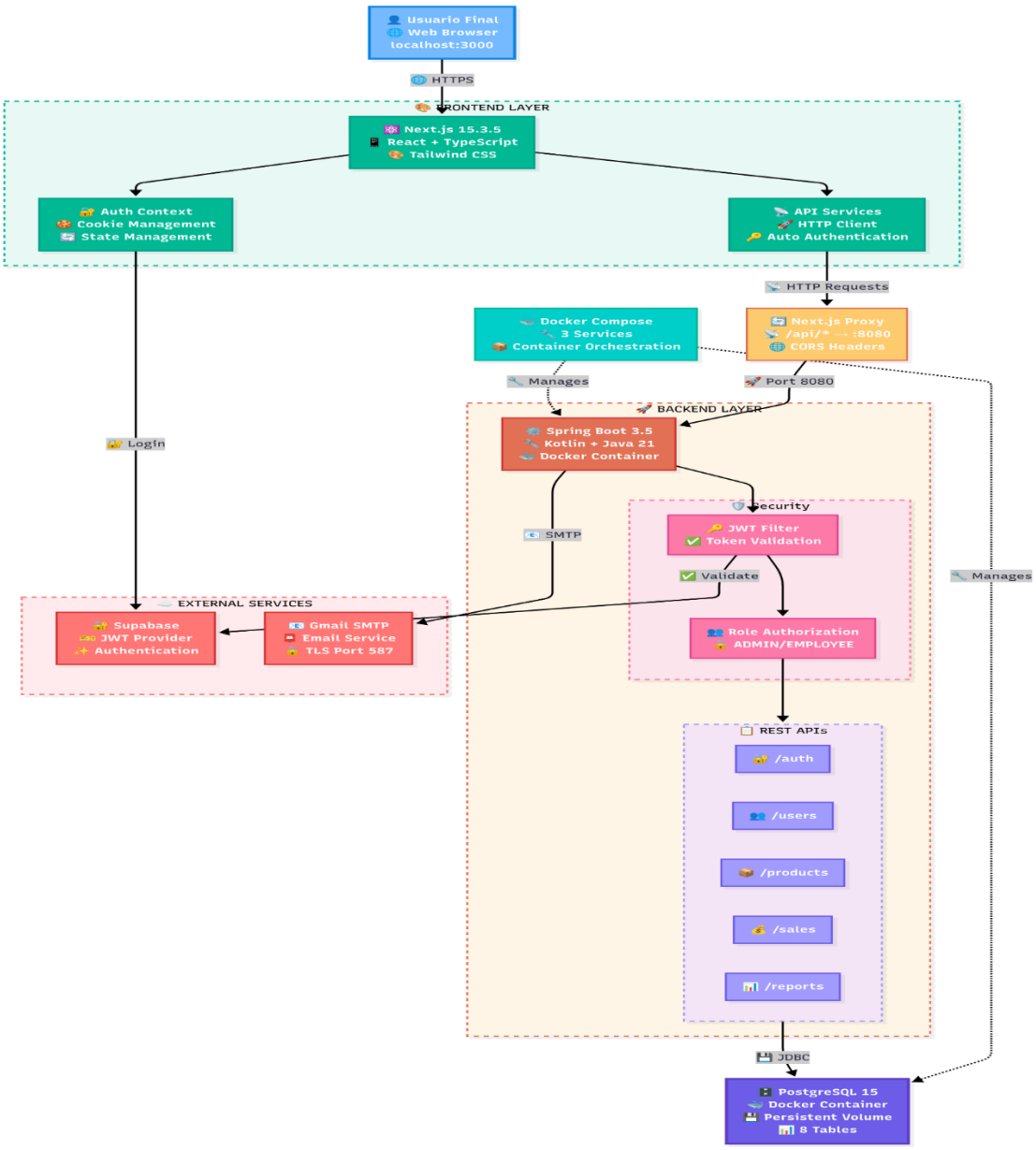
La comunicación con el backend únicamente se realiza a través de servicios en la nube especializados como supabase, firebase, AWS, etc. Estas proporcionan funcionalidades completas de base de datos, autenticación, almacenamiento y APIs sin necesidad de mantener una infraestructura propia de servidor**.**

Características Principales.

- **Organización Vertical por Features:** Cada módulo del sistema se desarrolla como una unidad completa que incluye desde la interfaz hasta la comunicación con el backend.
- **Frontend Autónomo:** La aplicación cliente contiene toda la lógica de negocio y se comunica con servicios externos solamente para persistencia y operaciones de backend.
- **Backend Externalizado:** No existe servidor propio, todos los servicios de backend son proporcionados por proveedores de cloud externos.
- **Comunicación API-First:** La comunicación entre frontend y backend se realiza exclusivamente a través de APIs REST.
- **Escalabilidad Modular:** Nuevas funcionalidades se agregan como módulos independientes sin afectar el código existente.

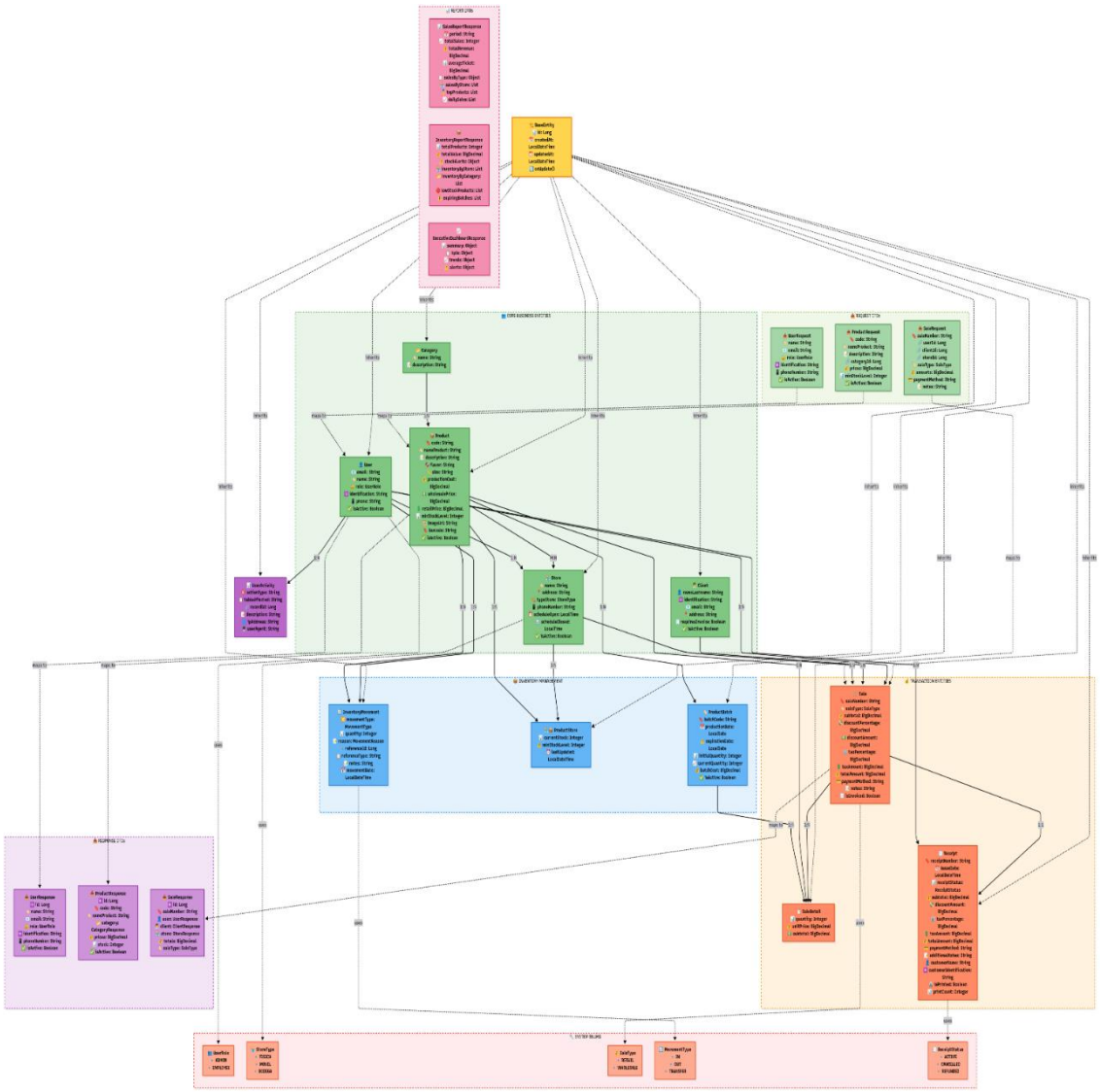
Diagrama de Arquitectura.

Figura 7.



Diagramas de Clases.

Figura 8.



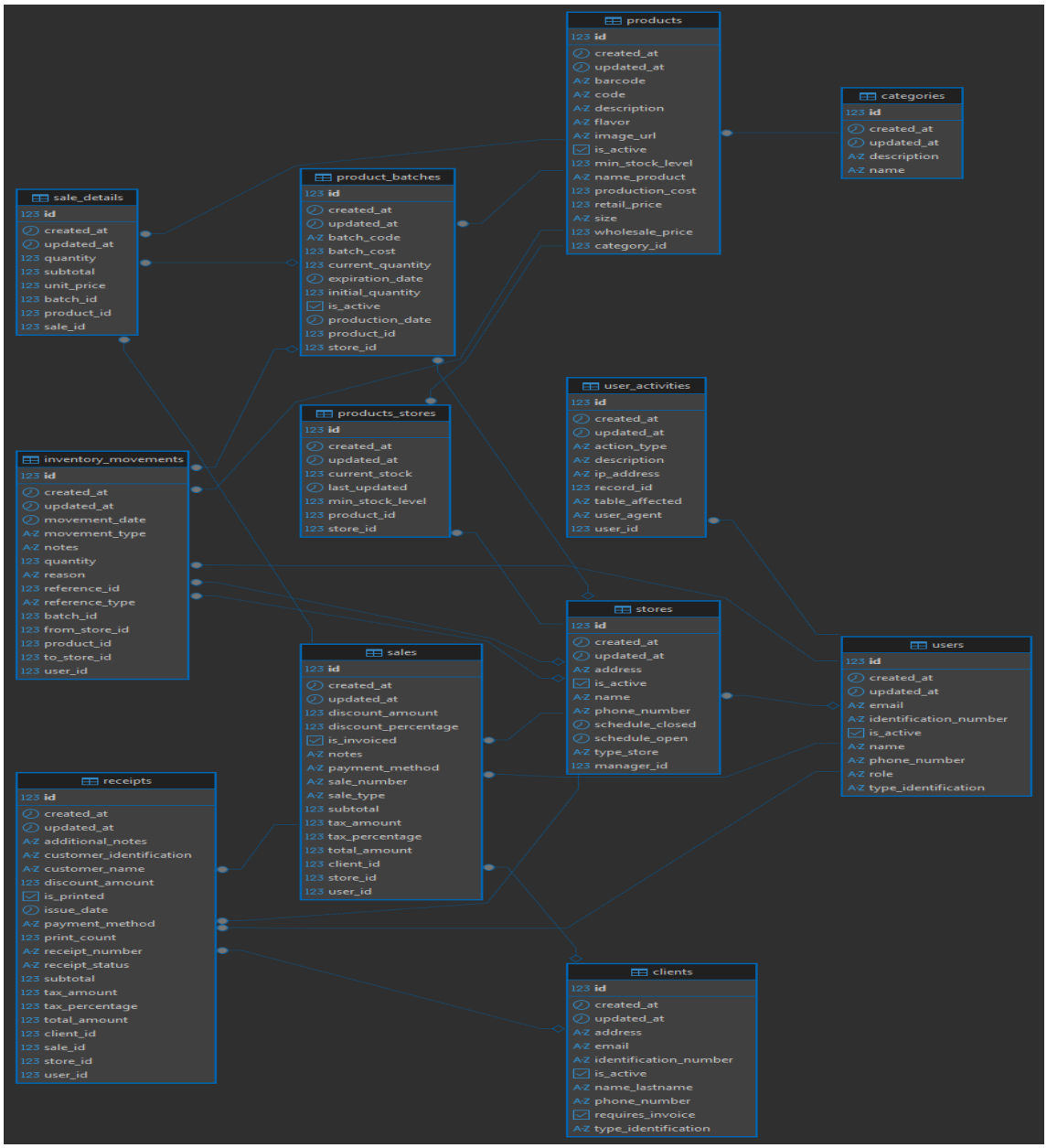
Chocorocks-backend/

src/main/kotlin/com/puce/chocorocks_backend/

```
├── config/                 # Configuraciones
│   ├── DataSeeder.kt         # Datos de prueba
│   └── SecurityConfig.kt     # Configuración de seguridad
├── controllers/            # Controladores REST (17)
│   ├── AuthController.kt     # Autenticación
│   ├── CategoryController.kt  # Gestión de categorías
│   ├── ProductController.kt   # Gestión de productos
│   ├── SaleController.kt     # Gestión de ventas
│   ├── ReportController.kt    # Sistema de reportes
│   └── ...                 # Otros controladores
├── dtos/                   # Data Transfer Objects
│   ├── requests/           # DTOs de entrada (13)
│   ├── responses/          # DTOs de salida (17+)
│   └── xml/                # DTOs para XML
├── models/entities/        # Entidades JPA (14)
│   ├── BaseEntity.kt       # Entidad base con auditoría
│   ├── Product.kt          # Productos
│   ├── Sale.kt             # Ventas
│   └── ...                 # Otras entidades
├── services/               # Interfaces de servicios
│   └── Impl/               # Implementaciones (16)
├── mappers/                # Mappers bidireccionales (12)
├── exceptions/             # Excepciones personalizadas (10+)
├── security/               # Configuración de seguridad JWT
├── routes/                 # Constantes de rutas
└── utils/                  # Utilidades y validaciones
```

Diagrama Entidad-Relación (Base de Datos).

Figura 9.



Capítulo II

Construcción del Sistema

Metodología de Trabajo.

Scrum.

Scrum es una metodología ágil, también se le puede considerar como un marco de trabajo ágil, para gestionar proyectos complejos, específicamente en desarrollo de software, Surgió en los años 90, basados en los principios del Manifiesto ágil del 2001, que valora: Individuos e iteraciones, Software Funcional, colaboración con el cliente, y respuesta al cambio.

Características Clave.

- **Iterativo e incremental:**
  El trabajo se divide en ciclos cortos llamados sprints, normalmente entre 2 o 4 semanas.

- **Roles Definidos:**
  Product Owner: Representa el cliente también llamados stakeholders, prioriza requisitos.
  Scrum Master: Facilita el proceso y elimina obstáculos, es experto en Metodología Scrum.
  Equipo de Desarrollo: Se refiere al equipo de desarrollo, programadores, diseñadores, etc.

- **Artefactos Principales:**
  Product Backlog: Lista dinámica de requisitos priorizados.
  Sprint Backlog: Tareas Seleccionadas para el sprint actual.
  Incremento: Producto funcional entregado al final del sprint.

- **Eventos o Ceremonias:**
  Sprint Planning: Planificación de tareas para el sprint.
  Daily Scrum: Reunión diaria de 15 minutos.
  Sprint Review: Demostración del incremento.
  Sprint Retrospective: Retroalimentación del sprint.
  Ventajas.
- **Flexibilidad:** Permite incorporar cambios en los requisitos.
- **Transparencia:** Facilita la supervisión del tutor.
- **Entrega Rápida:** MVP Mínimo Producto Viable en pocas semanas.
- **Trabajo Colaborativo:** Ideal para equipos de varios integrantes.

Requerimientos de Software y Hardware.

Software.

Navegadores Web Recomendados:

- Google Chrome: Versión 120.0.0 o Posterior.
- Mozilla Firefox: Versión 121.0.0 o posterior.
- Microsoft Edge: 120.0.0 o Posterior.
- Safari: 17.0.0 o Posterior.

Navegadores Web Mínimos soportados.

- Chrome: 109.0.0 o posterior.
- Firefox: 109.0.0 o posterior.
- Edge: 109.0.0 o posterior.
- Safari: 15.0.0 o posterior.

Tecnologías Web Requeridas.

- JavaScript: Habilitado (Obligatorio).
- Cookies: Habilitadas (Para autenticación).
- Local Storage: Soportado.
- CSS 3: Soporte Completo.
- HTML 5: Soporte Completo.

Sistemas Operativos Compatibles.

Desktop/Laptops.

- Windows: 10, 11.
- MacOS: 10.15 (Catalina o Superior).
- Linux: Ubuntu 18.04 o Posterior, CentOS 7 o superior, Debian 10 o posterior.

Dispositivos Móviles.

- Android 8 o Superior (API level 26 +)
- IOS: 13 o superior.
- iPadOS: 13.0 o Superior.

Tablets.

- Android: 8.0 o posterior.
- iPad: IOS 13.0 o superior.
- Windows Tablets. Windows 10 o superior.

Plugins y Extensiones.

- No se requiere plugins adicionales.
- No se requiere Flash Player.
- No se requiere Java.

Requerimientos de Hardware.

Computadoras Desktop/Laptops.

Especificaciones Mínimas:

- Procesador: Intel I3 / AMD Ryzen 3 o Equivalente.
- RAM: 4GB.
- Almacenamiento: 1GB de espacio libre.
- Tarjeta Gráfica: Integrada Básica.
- Resolución: 1024 x 768 píxeles.
- Conexión a Internet: 1 Mbps.

Especificaciones Recomendadas:

- Procesador: Intel I5 / AMD Ryzen 5 o Superior.
- RAM: 8GB o más.
- Almacenamiento: 2GB de espacio Libre.
- Tarjeta Gráfica: Integrada Moderna.
- Resolución: 1366 x 728 o superior.
- Conexión a Internet: 5 Mbps o superior.

Dispositivos Móviles:

Android:

- RAM: 3GB mínimo, 4GB recomendado o superior.
- Almacenamiento: 1GB de espacio Libre.
- Procesador: Snapdragon 660 / Exynos 8895 o superior.
- Resolución: 720p mínimo, 1080p recomendado.
- Android 8 o superior.

iPhone:

- RAM: 3GB mínimo, 4GB recomendado o superior.
- Almacenamiento: 1GB de espacio libre.
- Procesador: A11 Bionic o superior.
- Resolución: 375 x 667 mínimo o superior.
- IOS: 13 o superior.

Tablets

Android Tablets:

- RAM: 3 GB mínimo, 4 GB recomendado
- Almacenamiento: 2 GB espacio libre
- Resolución: 1024x768 mínimo
- Android: 8.0+

iPad:

- RAM: 3 GB mínimo

- Almacenamiento: 2 GB espacio libre
- Resolución: 1024x768 mínimo
- iOS/iPadOS: 13.0+

Diseño de Interfaces de Usuario.

1.-Pantalla Principal (Dashboard)

Función: Panel de control central que proporciona una vista general y resumida del estado del sistema.

Elementos principales.

- **Métricas claves:** Totalidad de productos, Categorías, Tiendas Activas, Alertas de Stock.
- **Resumen:** Clientes Registrados, Productos Activos.
- **Acciones Rápidas:** Botones para gestionar productos, control de inventario, ver alertas.
- **Actividad Reciente:** Movimientos recientes de inventario con fechas y tipos.
- **Alertas:** Próximos a vencer, es decir, productos que caducan en 30 días.

2. Gestión de productos.

Función: Administrar el catálogo completo de productos de la empresa.

Elementos principales.

- **Filtros de búsqueda:** Por nombre, código, categoría, sabor, estado.
- **Tabla de productos:** Muestra imagen, código, nombre, categoría, sabor, tamaño, precios detalles y mayoristas, stock mínimo, estado.
- **Acciones:** Editar y eliminar producto individualmente.
- **Gestión:** Botones para gestionar las categorías y crear nuevos productos.

3. Gestión de Inventario.

Función: Control detallado de los lotes, stock, movimientos de inventario.

Elementos Principales.

- **Filtros:** Por tienda, ubicación, producto, filtros especiales.
- **Control de lotes:** Código de lote, producto, ubicación, fechas de producción, vencimiento.
- **Estado de Stock:** Indicadores visuales como stock bajo, normal, vencido.
- **Movimientos:** Botones de transferencias, nuevo lote.
- **Métricas:** Total del stock, valor total del inventario.

4. Gestión de Ventas.

Función: Administrar ventas realizadas y generar nuevas facturas.

Elementos principales.

- **Filtros de búsqueda:** Por número, vendedor, cliente, fechas, tienda, tipo, estado.
- **Lista de ventas:** Número de ventas, fecha, vendedor, cliente, tienda, tipo, total, estado de facturación.
- **Acciones:** Ver detalle, editar ventas.
- **Función:** Crear nueva venta.

5.- Gestión de tiendas.

Función: Administrar puntos de ventas y sucursales de la empresa.

Elementos principales.

- **Filtros:** Por nombre, dirección, teléfono, tipo, gerente, estado.
- **Información de tiendas:** Nombre, Tipo de tienda, dirección, gerente, teléfono, horario, estado.
- **Estado Operativo:** Activa o Inactiva.
- **Acciones:** Editar tiendas.
- **Gestión:** Crear nueva tienda.

6.- Gestión de Clientes.

Función: Administrar Base de datos de los clientes de la empresa.

Elementos principales.

- **Filtros de búsqueda:** Por nombre, identificación, email, tipo, estado.
- **Datos del cliente:** Nombre, identificación, teléfono, email, dirección, facturación, estado.
- **Acciones:** Editar y eliminar clientes.
- **Función:** Agregar nuevo cliente.

7.- Gestión de Usuarios.

Función: Administrar usuarios del sistema y sus permisos.

Elementos del principales.

- **Búsqueda:** Por nombre, email o identificación.
- **Información del usuario:** Nombre, email, rol (Administrador/empleado), identificación, teléfono, estado, fecha registro.
- **Control de acceso:** Diferentes roles con distintos niveles de permiso.
- **Acciones.** Editar.
- **Gestión:** Crear nuevo usuario.

8. – Reportes.

Función: Generar análisis y reportes del sistema.

Módulos:

- **Reporte de ventas:** Análisis de ventas por periodo, productos y tiendas

- **Reporte de inventario:** Estado Actual del inventario y rotación de productos.
- **Reporte de rentabilidad:** Análisis de Costos, ingresos por producto y tienda.
- **Reporte de productos más vendidos:** Ranking de productos con mayor demanda en ventas.
- **Reporte de trazabilidad:** Seguimiento completo de lotes desde la producción hasta venta.

9.- Alertas.

Función: Notificaciones automáticas del sistema sobre situaciones que requieren de atención.

Tipos de alertas:

- Stock bajo en productos.
- Productos próximos para vencerse.
- Productos vencidos.

Movimientos de inventario crítico.

Gestión de la base de datos.

1. Entidades.

1.1 Products.

Descripción: Entidad central que almacena el catálogo de productos de la empresa.

- **id (PK)** - Identificador único del producto
- **created_at** - Fecha de creación del registro
- **updated_at** - Fecha de última actualización
- **barcode** - Código de barras del producto
- **code** - Código interno del producto
- **description** - Descripción del producto
- **flavor** - Sabor del producto
- **image_url** - URL de la imagen del producto
- **is_active** - Estado activo/inactivo
- **min_stock_level** - Nivel mínimo de stock
- **name_product** - Nombre del producto
- **production_cost** - Costo de producción
- **retail_price** - Precio de venta al detalle
- **size** - Tamaño del producto
- **wholesale_price** - Precio de venta al mayoreo
- **category_id (FK)** - Referencia a categorías

1.2 Categories.

Descripción: Clasificaciones de los productos en diferentes categorías.

- **id (PK)** - Identificador único de la categoría
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **description** - Descripción de la categoría
- **name** - Nombre de la categoría

1.3 Product_Batches.

Descripción: Control de lotes de producción con fechas de vencimientos y cantidades.

- **id (PK)** - Identificador único del lote
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **batch_code** - Código único del lote
- **batch_cost** - Costo total del lote
- **current_quantity** - Cantidad actual disponible
- **expiration_date** - Fecha de vencimiento
- **initial_quantity** - Cantidad inicial del lote
- **is_active** - Estado del lote
- **production_date** - Fecha de producción
- **product_id (FK)** - Referencia al producto
- **store_id (FK)** - Referencia a la tienda

1.4 Sales.

Descripción: Registro de todas las ventas realizadas en el sistema.

- **id (PK)** - Identificador único de la venta
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **discount_amount** - Monto de descuento aplicado
- **discount_percentage** - Porcentaje de descuento
- **is_invoiced** - Indica si se ha facturado
- **notes** - Notas adicionales
- **payment_method** - Método de pago utilizado
- **sale_number** - Número de venta
- **sale_type** - Tipo de venta (detalle/mayoreo)
- **subtotal** - Subtotal antes de impuestos
- **tax_amount** - Monto de impuestos
- **tax_percentage** - Porcentaje de impuestos
- **total_amount** - Monto total de la venta
- **client_id (FK)** - Referencia al cliente
- **store_id (FK)** - Referencia a la tienda
- **user_id (FK)** - Referencia al usuario vendedor

1.5 Sale_Details.

Descripción: Detalles de los productos que se venden en una venta.

- **id (PK)** - Identificador único del detalle
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **quantity** - Cantidad vendida
- **subtotal** - Subtotal del producto
- **unit_price** - Precio unitario
- **batch_id (FK)** - Referencia al lote

- **product_id (FK)** - Referencia al producto
- **sale_id (FK)** - Referencia a la venta

1.6 Stores.

Descripción: Puntos de ventas y ubicaciones de las sucursales de la empresa.

- **id (PK)** - Identificador único de la tienda
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **address** - Dirección de la tienda
- **is_active** - Estado activo/inactivo
- **name** - Nombre de la tienda
- **phone_number** - Número de teléfono
- **schedule_closed** - Horario de cierre
- **schedule_open** - Horario de apertura
- **type_store** - Tipo de tienda
- **manager_id** (FK) - Referencia al gerente

1.7 Clients.

Descripción: Base de Datos de clientes de la empresa.

- **id (PK)** - Identificador único del cliente
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **address** - Dirección del cliente
- **email** - Correo electrónico
- **identification_number** - Número de identificación
- **is_active** - Estado activo/inactivo
- **name_lastname** - Nombre completo
- **phone_number** - Número de teléfono
- **requires_invoice** - Requiere facturación
- **type_identification** - Tipo de identificación

1.8 Users.

Descripción: Usuarios del sistema con roles y permisos.

- **id (PK)** - Identificador único del usuario
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **email** - Correo electrónico
- **identification_number** - Número de identificación
- **is_active** - Estado activo/inactivo
- **name** - Nombre del usuario
- **phone_number** - Número de teléfono
- **role** - Rol del usuario (admin/empleado)
- **type_identification** - Tipo de identificación

1.9 Products_Stores.

Descripción: Relación entre productos y tiendas con stock.

- **id (PK)** - Identificador único
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **current_stock** - Stock actual en la tienda
- **last_updated** - Última actualización de stock
- **min_stock_level** - Nivel mínimo de stock
- **product_id (FK)** - Referencia al producto
- **store_id (FK)** - Referencia a la tienda

1.10 Inventory_movements.

Descripción: Registro de todos los movimientos de inventario, técnicamente entradas y salidas.

- **id (PK)** - Identificador único
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **movement_date** - Fecha del movimiento
- **movement_type** - Tipo de movimiento (entrada/salida/transferencia)
- **notes** - Notas del movimiento
- **quantity** - Cantidad movida
- **reason** - Razón del movimiento
- **reference_id** - ID de referencia (venta, compra, etc.)
- **reference_type** - Tipo de referencia
- **batch_id (FK)** - Referencia al lote
- **from_store_id (FK)** - Tienda origen
- **product_id (FK)** - Referencia al producto
- **to_store_id (FK)** - Tienda destino
- **user_id (FK)** - Usuario que realizó el movimiento

1.11 Receips.

Descripción: Documentos tipo facturas generado por las ventas.

- **id (PK)** - Identificador único
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **additional_notes** - Notas adicionales
- **customer_identification** - Identificación del cliente
- **customer_name** - Nombre del cliente
- **discount_amount** - Monto de descuento
- **is_printed** - Indica si se ha impreso
- **issue_date** - Fecha de emisión
- **payment_method** - Método de pago
- **print_count** - Número de impresiones

- **receipt_number** - Número del recibo
- **receipt_status** - Estado del recibo
- **subtotal** - Subtotal
- **tax_amount** - Monto de impuestos
- **tax_percentage** - Porcentaje de impuestos
- **total_amount** - Monto total
- **client_id (FK)** - Referencia al cliente
- **sale_id (FK)** - Referencia a la venta
- **store_id (FK)** - Referencia a la tienda
- **user_id (FK)** - Referencia al usuario documento tipo factura generados por las ventas.

1.12 User_Activities.

Descripción: Registro de actividades y auditoria del sistema.

- **id (PK)** - Identificador único
- **created_at** - Fecha de creación
- **updated_at** - Fecha de actualización
- **action_type** - Tipo de acción realizada
- **description** - Descripción de la actividad
- **ip_address** - Dirección IP
- **record_id** - ID del registro afectado
- **table_affected** - Tabla afectada
- **user_agent** - Agente de usuario
- **user_id (FK)** - Referencia al usuario

2. Relaciones entre entidades.

2.1 Relaciones Uno a muchos.

1. CATEGORIES ← PRODUCTS

- Una categoría puede tener múltiples productos
- Un producto pertenece a una sola categoría

2. PRODUCTS ← PRODUCT_BATCHES

- Un producto puede tener múltiples lotes
- Un lote pertenece a un solo producto

3. STORES ← PRODUCT_BATCHES

- Una tienda puede tener múltiples lotes
- Un lote se encuentra en una sola tienda

4. USERS ← SALES

- Un usuario puede realizar múltiples ventas
- Una venta es realizada por un solo usuario

5. CLIENTS ← SALES

- Un cliente puede tener múltiples ventas
- Una venta pertenece a un solo cliente

6. STORES ← SALES

- Una tienda puede tener múltiples ventas
- Una venta se realiza en una sola tienda

7. SALES ← SALE_DETAILS

- Una venta puede tener múltiples detalles
- Un detalle pertenece a una sola venta

8. PRODUCTS ← SALE_DETAILS

- Un producto puede estar en múltiples detalles de venta
- Un detalle de venta corresponde a un solo producto

9. PRODUCT_BATCHES ← SALE_DETAILS

- Un lote puede estar en múltiples detalles de venta
- Un detalle de venta proviene de un solo lote

2.2 Relaciones muchos a muchos.

1. PRODUCTS ↔ STORES (a través de PRODUCTS_STORES)

- Un producto puede estar en múltiples tiendas
- Una tienda puede tener múltiples productos
- La tabla intermedia almacena el stock específico

3. Diccionario de Datos.

3.1 Tipos de datos utilizados.

- **INTEGER/BIGINT:** Identificadores, cantidades, precios (en centavos)
- **VARCHAR:** Textos cortos (nombres, códigos)
- **TEXT:** Descripciones largas y notas
- **DECIMAL/NUMERIC:** Valores monetarios y porcentajes
- **BOOLEAN:** Campos de estado (is_active, is_printed)
- **TIMESTAMP:** Fechas y horas
- **DATE:** Solo fechas

3.2 Convenciones de nomenclatura.

- **Tablas:** Plural en inglés (products, sales)
- **Campos:** Snake_case en inglés
- **Claves primarias:** Siempre "id"
- **Claves foráneas:** Nombre_tabla + "_id"
- **Timestamps:** created_at, updated_at

Implementación del Backend(API).

1. Resumen.

El backend API de Chocorocks es una aplicación empresarial desarrollada en Kotlin con Spring Boot que gestiona de manera integral el inventario, ventas y reportería de una empresa de chocolates artesanales, este sistema contiene una arquitectura moderna y escalable.

1.1 Características Principales.

- **Arquitectura por capas:** Con separación clara de responsabilidades, con bajo acoplamiento.
- **Seguridad Robusta:** Con autenticación JWT y autenticación externa con Supabase.
- **Sistemas de Reportes:** Con métricas empresariales.
- **Generación de recibos:** Generación XML y envío por Email.
- **Control de inventario:** Con trazabilidad completa de lotes.
- **Alto rendimiento:** Con optimizaciones de base de datos.

2. Stack Tecnológico.

- **Lenguaje de programación:** Kotlin.
- **Framework:** Spring Boot.
- **Base de Datos:** PostgreSQL.
- **ORM:** JPA/Hibernate.
- **Seguridad:** Spring Security.
- **Auth Provider:** Supabase.
- **Email: Spring**Mail.
- **XML:** JAXB.
- **Testing DB:** H2.
- **Containerización:** Docker.

3. Especificaciones de Api.

3.1 Base URL.

https://api.chocorocks.com/chocorocks/api

3.2 Autenticación.

Toda la API requieren autenticación JWT mediante header:

- Authorization: Bearer <jwt_token>

3.3 Endpoints principales.

3.3.1 Autenticación.

Método	Endpoint	Descripción	Roles
GET	/auth/me	Obtener usuario actual	USER
GET	/auth/status	Estado de autenticación	USER
POST	/auth/validate	Validar token	USER
GET	/auth/permissions	Permisos del usuario	USER

3.3.2 Gestión de Productos.

Método	Endpoint	Descripción	Roles
GET	/products	Listar productos	EMPLOYEE, ADMIN
GET	/products/{id}	Obtener producto	EMPLOYEE, ADMIN
POST	/products	Crear producto	EMPLOYEE, ADMIN
PUT	/products/{id}	Actualizar producto	EMPLOYEE, ADMIN
DELETE	/products/{id}	Eliminar producto	ADMIN

3.3.3 Gestión de categorías.

Método	Endpoint	Descripción	Roles
GET	/categories	Listar categorías	EMPLOYEE, ADMIN
POST	/categories	Crear categoría	EMPLOYEE, ADMIN
PUT	/categories/{id}	Actualizar categoría	EMPLOYEE, ADMIN

DELETE	/categories/{id}	Eliminar categoría	ADMIN

3.3.4 Gestión de Inventario.

Método	Endpoint	Descripción	Roles
GET	/product-batches	Listar lotes	EMPLOYEE, ADMIN
POST	/product-batches	Crear lote	EMPLOYEE, ADMIN
GET	/inventory-movements	Movimientos	EMPLOYEE, ADMIN
POST	/inventory-movements	Registrar movimiento	EMPLOYEE, ADMIN

3.3.5 Gestión de ventas.

Método	Endpoint	Descripción	Roles
GET	/sales	Listar ventas	EMPLOYEE, ADMIN
POST	/sales	Crear venta	EMPLOYEE, ADMIN
POST	/sales/{id}/complete-with-receipt	Completar con recibo	EMPLOYEE, ADMIN
GET	/sale-details	Detalles de ventas	EMPLOYEE, ADMIN

3.3.6 Sistemas de recibos.

Método	Endpoint	Descripción	Roles
GET	/receipts	Listar recibos	EMPLOYEE, ADMIN
POST	/receipts	Generar recibo	EMPLOYEE, ADMIN

PATCH	/receipts/{id}/cancel	Cancelar recibo	ADMIN
GET	/receipts/{id}/download-xml	Descargar XML	EMPLOYEE, ADMIN
POST	/receipts/{id}/send-email	Enviar por email	EMPLOYEE, ADMIN

3.3.7 Sistemas de reportes.

Método	Endpoint	Descripción	Roles
GET	/reports/sales?startDate=yyyy-mm-dd&endDate= yyyy-mm-dd	Reporte de ventas	EMPLOYEE, ADMIN
GET	/reports/inventory	Reporte de inventario	EMPLOYEE, ADMIN
GET	/reports/profitability?startDate=yyyy-mm-dd&endDate= yyyy-mm-dd	Reporte de rentabilidad	ADMIN
GET	/reports/best-selling-products?startDate=yyyy-mm-dd&endDate= yyyy-mm-dd	Productos más vendidos	EMPLOYEE, ADMIN
GET	/reports/traceability/batch/{code}	Trazabilidad de lote	EMPLOYEE, ADMIN
GET	/reports/executive-dashboard?startDate=yyyy-mm-dd&endDate= yyyy-mm-dd	Dashboard ejecutivo	ADMIN

3.3.8 Gestión de Clientes.

Método	Endpoint	Descripción	Roles
GET	/clients	Listar clientes	EMPLOYEE, ADMIN
POST	/clients	Crear cliente	EMPLOYEE, ADMIN
PUT	/clients/{id}	Actualizar cliente	EMPLOYEE, ADMIN

3.3.9 Gestión de Tiendas.

Método	Endpoint	Descripción	Roles
GET	/stores	Listar tiendas	EMPLOYEE, ADMIN

POST	/stores	Crear tienda	ADMIN
PUT	/stores/{id}	Actualizar tienda	ADMIN

3.3.10 Gestión de Usuarios.

Método	Endpoint	Descripción	Roles
GET	/users	Listar usuarios	ADMIN
POST	/users	Crear usuario	ADMIN
PUT	/users/{id}	Actualizar usuario	ADMIN

4. Roles y Permisos.

4.1. ADMIN.

- Gestión completa de usuarios
- Gestión de tiendas y configuración
- Eliminación de registros
- Reportes de rentabilidad
- Dashboard ejecutivo
- Cancelación de recibos

4.2 EMPLOYEE.

- Gestión de productos y categorías
- Gestión de inventario y lotes
- Procesamiento de ventas
- Generación de recibos
- Reportes básicos

5. Funcionalidades.

5.1 Sistemas de Reportes.

Reportes Ventas.

- Ventas por período con filtros
- Análisis por tipo (retail/wholesale)
- Distribución por tiendas
- Productos más vendidos

Reportes de Inventario.

- Estado actual de stock
- Alertas de stock bajo/crítico
- Productos próximos para vencer

- Distribución por categorías/tiendas

Reporte de rentabilidad.

- Análisis de costos vs ingresos
- Rentabilidad por producto
- Rentabilidad por categoría/tienda
- Márgenes de ganancia

Reporte Productos más vendidos.

- Ranking por cantidad/ingresos
- Precio promedio de venta

Trazabilidad de lotes.

- Seguimiento completo de lotes
- Movimientos de inventario
- Control de vencimientos

Dashboard.

- Métricas de rendimiento
- Tendencias y alertas
- Resumen operacional

5.2 Gestión de Inventario avanzada.

Control de lotes.

- Control de vencimientos: Alertas automáticas
- Movimientos registrados: Salida, transferencia

Alertas Inteligentes.

Algoritmo de Alertas inteligentes:

```kotlin
// Algoritmo de alertas de stock
when {
    currentStock == 0 -> "OUT_OF_STOCK"
    currentStock <= (minStockLevel * 0.5) -> "CRITICAL"
    currentStock <= minStockLevel -> "LOW"
    else -> "NORMAL"
}
```

5.3 Recibos electrónicos.

Generación de XML código:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<recibo>
    <numeroRecibo>REC-TDA-20240123-143000</numeroRecibo>
    <fechaEmision>2024-01-23 14:30:00</fechaEmision>
    <estado>ACTIVE</estado>
    <tienda>
        <nombre>Tienda Principal</nombre>
        <direccion>Av. Principal 123, Centro, Quito</direccion>
    </tienda>
    <totales>
        <subtotal>100.00</subtotal>
        <montoImpuesto>12.00</montoImpuesto>
        <total>112.00</total>
    </totales>
</recibo>
```

Envío por email.

- Templates HTML
- Adjuntos XML automáticos
- Personalización por cliente

6. Seguridad.

6.1 Autenticación y Autorización.

- Validación de sesiones mediante tokens JWT generados por supabase.
- Extracción y verificación de credenciales en cada petición través de un filtro de seguridad.
- Asignación de roles dinámicos como ADMIN y EMPLOYEE a partir de los metadatos del usuario.
- Control de Acceso basado en roles, para cada endpoints según el método HTTP.

6.2 Protección de endpoints.

- Acceso público solo a rutas básicas: /, /health, /api-info.
- Acceso restringido a operaciones sensibles como la eliminación de usuarios o los reportes, estos son exclusivos para el usuario ADMIN.
- Autorización diferenciada entre lecturas y escrituras EMPLOYEE – ADMIN y eliminaciones solo ADMIN.

- Endpoints específicos de autenticación: Auth/me, /auth/status, /auth/validate.

6.3 Gestión de sesiones y seguridad adicional.

- Operación en modo stateless sin sesiones de servidor, únicamente con JWT.
- Configuración de CORS para restringir orígenes permitidos, por ejemplo, dominios del frontend autorizados.
- Manejo centralizado de excepciones con respuestas estandarizadas de error.
- Validación estricta de parámetros como fechas, límites y filtros en los reportes para evitar abusos de consultas.

Implementación Frontend.

1. Stack Tecnológico.

- **Frontend:** Next.js + TypeScript + Tailwind CSS
- **Estado:** Context API + Custom Hooks
- **Validaciones:** Custom validators + TypeScript
- **API:** RESTful service layer

2. Módulos implementados.

2.1Autenticación y Seguridad.

- LoginForm con validaciones
- AuthContext para manejo global
- ProtectedRoute para rutas privadas
- Manejo de tokens y sesiones

2.2 Gestión de Clientes.

- CRUD completo (Crear, Leer, Actualizar, Eliminar)
- Validaciones Ecuador: Cédula, RUC, Pasaporte
- Filtros avanzados por tipo identificación
- ClientForm modal con validaciones en tiempo real
- Estados activo/inactivo

2.3 Gestión de Productos.

- CRUD de productos con categorías
- Precios diferenciados (detalle/mayorista)
- Gestión de sabores y tamaños
- Costos de producción
- Códigos de productos únicos
- Estados activo/inactivo

2.4 Inventario Avanzado.

- Sistema de lotes
- Movimientos de inventario (salida/transferencia)
- Alertas automáticas de stock bajo
- Control de fechas de vencimiento
- Trazabilidad completa por lote

- Múltiples tiendas y bodegas

2.5 Sistema de ventas.

- Ventas detalle vs mayorista
- Gestión de descuentos e impuestos
- Múltiples métodos de pago
- Numeración automática de ventas
- SaleDetailModal para ver detalles completos

2.6 Gestión de tiendas.

- Tipos: Física, Móvil, Bodega
- Asignación de gerentes
- Horarios de operación
- Información de contacto

2.7 Dashboard y métricas.

- Widgets de resumen (ventas, inventario, etc.)
- Alertas críticas centralizadas
- Accesos rápidos a funciones principales
- Métricas en tiempo real

3. Sistemas de reportes.

3.1 reportes de ventas.

- Ventas por período, tienda, vendedor
- Análisis de productos más vendidos

3.2 Reportes de inventario.

- Stock actual vs mínimo
- Productos vencidos/próximos a vencer
- Valor total del inventario
- Rotación de productos

3.3 Reporte de rentabilidad.

- Análisis de costos vs ingresos
- Márgenes por producto y tienda
- Productos más rentables

3.4 Top Products.

- Ranking de productos más vendidos
- Análisis por categorías

3.5 Trazabilidad.

- Seguimiento completo de lotes
- Historial de movimientos
- Ventas por lote específico

4. Notificaciones.

- NotificationProvider centralizado
- Diferentes tipos (success, error, warning, info)
- Auto-dismiss configurable
- Métodos helper (success, error, etc.)

5. Patrones de Diseño.

5.1 Estructura consistente:

```typescript
// Patrón de estado en listas
const [data, setData] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState('');
const [filters, setFilters] = useState({});

// Patrón de formularios
const [formData, setFormData] = useState({});
const [submitting, setSubmitting] = useState(false);
const [errors, setErrors] = useState({});

// Patrón de modales
const [showModal, setShowModal] = useState(false);
const [editingItem, setEditingItem] = useState(null);
```

5.2 Manejo de errores.

- BackendErrorHandler centralizado
- Validaciones en tiempo real
- Mensajes de error específicos
- Recuperación automática

5.3 Optimizaciones.

- useDebounce para búsquedas
- useCallback para optimizar renders
- Lazy loading donde corresponde
- Filtrado del lado cliente cuando apropiado

6. Responsive y UX.

6.1 Características.

- Diseño completamente responsive
- Sidebar colapsable en móviles

- Tablas con scroll horizontal
- Estados de loading consistentes
- Empty states informativos
- Confirmaciones para acciones críticas

7. Funcionalidades avanzadas.

7.1 Filtros y búsquedas.

- Búsqueda con debounce
- Filtros múltiples combinables
- Filtros por fechas
- Filtros por estado/categoría
- Limpieza de filtros

7.2 Exportación de datos.

- Exportación CSV en todos los reportes
- Nomenclatura con timestamp
- Datos formateados correctamente

7.3 Validaciones Robustas.

```typescript
// Validaciones específicas Ecuador
validators.cedula()
validators.ruc()
validators.phone()
validators.email()

// Validaciones de negocio
- Stock mínimo vs actual
- Fechas de vencimiento
- Horarios de tienda
- Precios y costos
```

8. Métricas y Analytics.

8.1 Dashboard Widgets.

- DailySalesWidget (ventas del día vs ayer)
- InventoryAlertsWidget (stock bajo/vencidos)
- TopProductsWidget (productos más vendidos)
- ProfitabilityWidget (rentabilidad últimos 30 días)

8.2 Cálculos automáticos.

- Totales de ventas con descuentos/impuestos
- Márgenes de rentabilidad
- Rotación de inventario
- Alertas de stock crítico

9. Funcionalidades.

- Sistema de Ventas: Venta completa con descuentos/impuestos
- Inventario Avanzado: Lotes, movimientos, trazabilidad, alertas
- Multi-tienda: Física/Móvil/Bodega con gerentes asignados
- Gestión de Usuarios: Roles ADMIN/EMPLOYEE
- Reportes Profesionales: Rentabilidad, top productos, trazabilidad
- Notificaciones: Sistema centralizado con auto-dismiss
- Responsive: Funciona perfecto en todos los dispositivos

Capítulo III

Pruebas y Estabilización

Objetivo.

Validar que la API de Chocorocks cumple los requisitos funcionales y no funcionales como correcta exposición de los endpoints, seguridad, integridad de datos, manejo de errores y desempaño estable previo al despliegue productivo, se consideraron rutas públicas y metadatos del servicio, flujos CRUD, reportes, ventas y recibos y operaciones auxiliares.

Resumen.

En la etapa final del desarrollo, se llevó a cabo el apartado de pruebas para garantizar el correcto funcionamiento del sistema y la experiencia fluida del usuario final, Para esto se combinaron dos tipos de pruebas, de integración ejecutadas con Postman para validar el funcionamiento y coherencia de los endpoints expuestos en el backend, y pruebas funcionales realizadas directamente sobre la interfaz del frontend, esto para poder simular escenarios reales de uso, estas pruebas de integración permitieron verificar que cada servicio recibiera y procesará correctamente los datos de entrada y devolvieran respuestas esperadas en cuanto a estructura, contenido y códigos de estados HTTP, las pruebas funcionales, que fuero desarrolladas en ,a interfaz de usuario, confirmaron la correcta implementación de los flujos del negocio, el manejo adecuado de roles, la presentación clara de mensajes y resultados al usuario.

Pruebas de integración.

A continuación, documentaremos los endpoints por módulo, y cada caso se indicará la ruta, roles, requests, y responses:

Autenticación.

GET /auth/me — Devuelve el usuario autenticado.

- Headers: Authorization: Bearer <token>

  Request body: —

  Response 200: { id,

  email,

  name,

  role,

  isAuthenticated }

- Errores: 401 si token inválido o ausente.

GET /auth/status — devuelve:

- Response 200: { isAuthenticated: boolean,

  user?: AuthUserResponse }

POST /auth/validate — Valida el token.

- Response 200: { valid: boolean,

  message,

  user?: { id, email, role } }

Gestión de clientes.

GET /clients — Lista clientes.

- Response 200: { id,

  nameLastname,

  typeIdentification,

  identificationNumber,

  phoneNumber?,

  email?,

  address?,

  requiresInvoice,

  isActive }.

GET /clients/{id} — Obtiene cliente por id.

- Response 200: Devuelve ClientResponse.
- Errores: 404 si no existe.

POST /clients — Crea cliente.

- Body (ClienteRequest) :

  { nameLastname,

  typeIdentification,

  identificationNumber,

  phoneNumber?,

  email?,

  address?,

  requiresInvoice?,

  isActive? }

- Response 201: ClientResponse (con id)
- Errores habituales: validaciones de unicidad/formatos → 400

PUT /clients/{id} — Actualiza cliente.

- Response 200 o 404 si no existe.

DELETE /clients/{id} — Elimina cliente (solo ADMIN).

- Response 204 o 404. (Reglas de rol en seguridad).

Gestión de Productos.

- GET /products — Lista productos. 200 → ProductResponse[].
- GET /products/{id} — 200 ProductResponse | 404.
- POST /products — Crea (Body ProductRequest) → 201 ProductResponse | 400 en referencias inválidas.
- PUT /products/{id} — 200 ProductResponse | 404.
- DELETE /products/{id} — 204 | 404 (solo ADMIN).

Lotes de producto.

- GET /product-batches — 200 ProductBatchResponse[].
- GET /product-batches/{id} — 200 | 404.
- POST /product-batches — 201 | 400 (referencias). Body: ProductBatchRequest.
- PUT /product-batches/{id} — 200 | 404.
- DELETE /product-batches/{id} — 204 | 404.

Detalles de ventas.

- GET /sale-details — 200 SaleDetailResponse[].
- GET /sale-details/{id} — 200 | 404.
- POST /sale-details — 201 | 400 (relaciones inválidas). Body: SaleDetailRequest.
- PUT /sale-details/{id} — 200 | 404.
- DELETE /sale-details/{id} — 204 | 404.

6. Ventas.

- GET /sales — 200 SaleResponse[].
- GET /sales/{id} — 200 | 404.
- POST /sales — 201 | 400. Body: SaleRequest.
- PUT /sales/{id} — 200 | 404.
- DELETE /sales/{id} — 204 | 404 (solo ADMIN).
- POST /sales/{id}/complete-with-receipt — Completa la venta y crea recibo.
  - Body: { paymentMethod?:
  - string,
  - additionalNotes?: string }
  - Response 201: ReceiptResponse con totales y enlace lógico a venta/cliente/tienda
  - Errores: 404 venta inexistente; 400 en validaciones

7. Recibos.

- GET /receipts — 200 ReceiptResponse[].

- GET /receipts/{id} — 200 | 404.
- POST /receipts — 201 | 400. Body: ReceiptRequest (valida unicidad de número, existencia de venta/usuario/tienda/cliente).
- PUT /receipts/{id} — 200 | 404.
- DELETE /receipts/{id} — 204 | 404.
- PATCH /receipts/{id}/cancel — 200 ReceiptResponse con status=CANCELLED | 404.
- PATCH /receipts/{id}/print — 200 ReceiptResponse marcando impreso.
- /receipts/{id}/download-xml — 200 application/xml (adjunto) | 404.
  - Contenido XML: estructura ReceiptXmlDto (recibo con tienda, usuario, venta, totales, etc.).
- POST /receipts/{id}/send-email — Envía email con HTML + XML adjunto.
  - Body opcional: { email?:

    string } (si no, toma email del cliente del recibo)

  - Response 200: true/false o 200 sin body (según implementación del controlador/servicio)
  - Errores: 400 email inválido o sin email del cliente. (Validación/plantilla en EmailServiceImpl + template HTML).

8. Reportes.

- GET /reports/sales?startDate=yyyy-MM-dd&endDate=yyyy-MM-dd&storeIds?=1,2
  - Response 200 (SalesReportResponse): totales, ventas por tipo/tienda, top productos, series diarias.
  - Errores: 400 por fechas inválidas; 500 genérico.
- GET /reports/inventory?storeIds?&categoryIds?
  - Response 200 (InventoryReportResponse): alertas de stock, inventario por tienda/categoría, lotes por vencer.
- GET/reports/profitability?startDate&endDate&storeIds?&categoryIds?
  - Response 200 (ProfitabilityReportResponse): margen, profit por producto/categoría/tienda.
  - Errores: 400 por fechas/rango inválido. (Acceso sensible: ADMIN).
- GET/reports/bestsellingproducts?startDate&endDate&limit=20&storeIds?&categoryIds?
  - Response 200 (BestSellingProductsReportResponse) con lista rankeada.
  - Errores: 400 por fechas o limit fuera de rango.

9. Tiendas.

- GET /stores — 200 StoreResponse[].
- GET /stores/{id} — 200 | 404**.**
- POST /stores — 201 | 400. Body: StoreRequest (valida nombre único y horario coherente apertura/cierre). Conflictos → DuplicateResourceException / BusinessValidationException.
- PUT /stores/{id} — 200 | 404 | 400 (mismas validaciones).
- DELETE /stores/{id} — 204 | 404 | 400 (operación inválida si tiene historial).

Pruebas funcionales.

Flujo 1 — Venta completa con recibo y envío por email (rol: EMPLOYEE)

1. Login – La UI obtiene /auth/status y muestra el usuario activo. El resultado esperado es isAuthenticated = true.

2. Catálogo – Lista productos y lotes /products, producto-batches, El resultado esperado: Las tablas listadas sin errores.

3. Crear venta – La Interfaz envía SaleRequest y muestra el número de venta, el resultado debe ser 201 con totales calculados, si faltan datos claves, el mensaje recibido será un 400 con mensaje de negocio.

4. Completar el recibo - /sales/id/complete-with-receipt con paymentMethod, resultado un estado 201 ReceiptResponse con ReceiptNumber, relación a sale/store/user.

5. Descargar XML/Ver Recibo – Desde el detalle, La interfaz gráfica llama a sale/id/download-xml, el resultad esperado debe ser una descarga application/xml y la estructura conforme ReceiptXmalDto.

6. Enviar por Email – "Enviar recibo" activa /receipts/id/send-email. El resultado esperado es la confirmación exitosa en IU, en validaciones de email, la UI muestra el mensaje del servicio.

Flujo 2 — Control de acceso y operaciones críticas (rol: ADMIN)

1. Eliminar entidades – Resultado, solo el admin ve el botón "Eliminar", la operación devuelve 204 o error controlado si hay historial asociado, por ejemplo, tiendas con ventas.

2. Reportes Avanzados - /reports/profitability – Resultado: Visible y ejecutable solo para ADMIN, muestra margen, revenue, costos, desgloses por producto/categoría/tienda; rechaza fechas inválidas con mensaje claro.

Flujo 3 — Búsqueda y trazabilidad de inventario

1. Listar Inventario - /reports/inventory con filtros de tienda/categoría, el resultado tarjetas/tablas con conteos, alertas de stock bajo y lotes por expirar, la UI resalta alertas críticas.

2. Validaciones de negocio (stock suficiente, lotes vencidos) – Resultado: ante errores, interfaz muestra mensajes específicos ("Stock insuficiente", "Lote vencido") provenientes de excepciones de negocio.

Flujo 4 — Gestión de Tiendas con reglas de horario

1. Crear/editar tienda – el resultado, si scheduleOpen > scheduleClosed o falta uno de los dos, la interfaz gráfica muestra el error de validación ("La hora de apertura no puede ser posterior", debe proporcionar tanto la hora de apertura como la de salida") mensaje de estado 201/200 en casos válidos o 409/400 en nombres duplicados u horarios inválidos.

Identificación de errores durante las pruebas funcionales

Durante la fase de pruebas funcionales detectamos diversos inconvenientes que afectaban el correcto funcionamiento del sistema. Estas iban desde fallos menores hasta errores críticos que, de no haber sido identificados en esta etapa, habrían comprometido el funcionamiento del sistema. A continuación, se describen algunos de los más relevantes:

1. **Error en la actualización del inventario**
   Uno de los problemas más críticos detectados fue que el inventario no se reducía tras una venta. Al revisar el código del *frontend* nos dimos cuenta de que el origen del fallo se encontraba en la visualización del stock, ya que se estaba mostrando un campo incorrecto en la interfaz de usuario.

2. **Inconsistencia en la generación de códigos para los recibos**
   En el *backend* se presentó un error en la lógica de generación de códigos para los recibos. Aunque el sistema creaba el código, este devolvía una excepción al no cumplir con el formato adecuado.
3. **Problemas en la generación y persistencia de recibos**
   Se identificaron dos problemas relacionados con la creación de recibos. En primer lugar, no era posible descargar el archivo XML correspondiente, debido a que la llamada al *endpoint* desde el *frontend* no se estaba realizando correctamente. En segundo lugar, al cerrar la ventana de creación, el recibo desaparecía a pesar de haberse creado exitosamente, lo cual se debía a un manejo incorrecto de los estados en la aplicación.

4. **Mal funcionamiento de los filtros por fecha**
   Al implementar filtros por rangos de fecha, nos dimos cuenta que no funcionaban correctamente. La causa estaba en que los campos de creación almacenados en la base de datos no estaban siendo expuestos en las respuestas del sistema (*responses*). Fue necesario incorporarlos explícitamente para que los filtros funcionen de manera adecuada.

Conclusiones

Durante el desarrollo del documento y del sistema propuesto, podría concluirse que el sistema ha sido construido con una arquitectura escalable y de bajo acoplamiento entre servicios, es decir, hay una separación bastante clara de responsabilidades entre capas siguiendo una estructura controladores, servicios, repositorios, modelos y utilidades, esto ha permitido una gestión ordenada de todo el código y de la persistencia de datos, este favorece a la mantenibilidad de la aplicación, también facilita la implementación a pruebas, e implementación de pruebas unitarias en un futuro, lo que contribuye a la detección temprana de errores y a la optimización del sistema.

Las pruebas de integración ejecutadas a través de Postman han demostrado que los endpoints responde de manera consistente frente a solicitudes válidas e inválidas concluyendo que además de tener un funcionamiento óptimo también maneja errores de manera correcta, es indicativo de que se ha implementado las validaciones, excepciones y políticas de seguridad correctamente.

Por otro lado, las pruebas funcionales realizadas en la interfaz gráfica han sido confirmados que los flujos críticos del negocio como la gestión de usuarios, ventas, generación de reportes y manejo de usuarios operan de manera afín con las reglas definidas, Se ha visualizado una correcta sincronización entre frontend y backend, asegurando que los cambios en la capa de datos se reflejen inmediatamente en la experiencia del usuario, lo que incrementan la usabilidad, eficiencia de todo el sistema.

Para finalizar, la documentación y el enfoque metodológico que refleja el documento respalda la calidad del desarrollo, destacando la importancia de integrar herramientas de prueba y control de calidad desde las fases iniciales del ciclo de vida de software, lo que puede llegar a concluir un producto bien hecho, estable y cumpliendo las necesidades operativas de la empresa chocorocks.

Recomendaciones

En vista de los resultados que se han obtenido, recomendamos establecer un plan de pruebas automatizadas que completen las pruebas manuales realizadas con Postman y desde la UI, esto va a permitir reducir la dependencia de intervención humana, aumentar la productividad, cobertura de escenarios y detección de errores con mayor rapidez en entornos de desarrollo y producción.

Asimismo, sería ideal implementar un sistema de monitoreo y logging centralizado que registre el comportamiento de la aplicación en tiempo real, esta medida ayudará a la facilitación de identificación y resolución temprana de incidencias en producción, además de servir como respaldo para auditorías técnicas o evaluaciones de seguridad.

Otra recomendación que podríamos ofrecer que es clave, es ampliar la batería de pruebas de seguridad, incluyendo evaluaciones de vulnerabilidad y pruebas de penetración o mejor llamado pentesting, con el fin de mejorar la protección contra accesos no autorizados y posibles ataques a los datos de usuarios de la aplicación.

Para finiquitar, sugerimos mantener actualizada la documentación técnica y funcional, asegurando que cada cambio en la lógica de negocio, en las rutas de la API o en las reglas de validaciones se refleje en un repositorio accesible para el equipo, esto será de vital importancia para la incorporación de nuevos desarrolladores, y no solo para eso, sino para garantizar coherencia del producto a largo plazo.

Referencias bibliográficas

- International Software Testing Qualifications Board. (2023). Standard glossary of terms used in software testing. ISTQB. https://www.istqb.org/downloads/send/20-istqb-glossary/229-istqb-glossary

- Postman Inc. (2023). Postman API Platform. Postman. https://www.postman.com/

- Richardson, C. (2018). Microservices patterns: With examples in Java. Manning Publications.

- IEEE. (2008). IEEE Standard for Software and System Test Documentation (IEEE Std 829-2008). IEEE. https://doi.org/10.1109/IEEESTD.2008.4478670

- Fowler, M. (2018). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.