Requerimientos funcionales

RF-001

Autenticación de Usuarios

ROLES INVOLUCRADOS: TODOS

CATEGORÍA: SEGURIDAD Y AUTENTICACIÓN

El sistema debe requerir autenticación mediante usuario y contraseña para acceder a cualquier funcionalidad.

Criterios de Aceptación:

- El sistema debe mostrar un formulario de inicio de sesión al acceder
- Debe validar credenciales contra la base de datos de usuarios
- Debe mostrar mensaje de error específico si las credenciales son incorrectas

RF-002

Control de Acceso por Roles

ROLES INVOLUCRADOS: TODOS

CATEGORÍA: AUTENTIFICACIÓN

El sistema debe restringir el acceso a funcionalidades según el rol asignado al usuario (Administrador, Cajero, Bodeguero).

Criterios de Aceptación:

- El sistema debe mostrar únicamente las opciones de menú autorizadas para cada rol
- Debe denegar acceso directo por URL a funciones no autorizadas
- El rol de Administrador debe tener acceso completo a todas las funcionalidades

RF-003

Creación de Usuarios

ROLES INVOLUCRADOS: ADMINISTRADOR

CATEGORÍA: GESTIÓN DE USUARIOS

El sistema debe permitir únicamente al Administrador crear nuevas cuentas de usuario en el sistema.

Criterios de Aceptación:

- Debe incluir formulario con: nombre completo, nombre de usuario, correo, rol y contraseña temporal
- Debe validar que el nombre de usuario sea único
- Debe permitir asignar uno de los roles disponibles: Administrador, Cajero, Bodeguero

RF-004

Reporte de Desempeño Financier

ROLES INVOLUCRADOS: ADMINISTRADOR

CATEGORÍA: REPORTES Y ANÁLISIS

El sistema debe generar reportes que muestren el desempeño financiero de destacables para la toma de decisiones.

Criterios de Aceptación:

- Debe calcular porcentaje de participación en ventas totales
- Debe ordenar productos por rentabilidad
- Debe incluir gráficos visuales (barras o pastel)

RF-005

Registro de Productos

ROLES INVOLUCRADOS: BODEGUERO, ADMINISTRADOR

CATEGORÍA: GESTIÓN DE INVENTARIO Y PRODUCTOS

El sistema debe permitir al Bodeguero registrar y mantener actualizada la información de los productos.

Criterios de Aceptación:

- Debe incluir: código/SKU, nombre, descripción, categoría, precio de venta, precio de compra, stock mínimo, unidad de medida, imagen
- Debe validar que el código sea único
- Debe permitir crear, modificar y consultar
- No debe permitir eliminar productos con movimientos históricos
- Debe permitir búsqueda por: código, nombre, categoría

RF-006

Registro de Compras a Proveedores

ROLES INVOLUCRADOS: BODEGUERO, ADMINISTRADOR

CATEGORÍA: GESTIÓN DE INVENTARIO Y PRODUCTOS

El sistema debe permitir registrar las compras realizadas a proveedores y actualizar automáticamente el inventario.

Criterios de Aceptación:

- Debe incluir: proveedor, fecha de compra, número de factura, productos, cantidades, precios unitarios, total
- Debe calcular automáticamente el total de la compra
- Debe actualizar stock de productos inmediatamente al guardar
- Debe generar número de compra consecutivo automático
- Debe registrar el usuario que realizó la compra

RF-007

Registro de Proveedores

ROLES INVOLUCRADOS: BODEGUERO, ADMINISTRADOR

CATEGORÍA: GESTIÓN DE INVENTARIO Y PRODUCTOS

El sistema debe permitir registrar y gestionar la información de proveedores.

- Debe incluir: nombre/razón social, NIT/RUC, dirección, teléfono, correo, contacto principal, días de crédito, productos que suministra
- Debe validar formato de NIT/RUC
- Debe permitir crear, modificar, consultar y desactivar proveedores

- Debe mostrar historial de compras por proveedor
- Debe permitir búsqueda por nombre o NIT

Criterios de Aceptación:

- Al registrar una venta, debe decrementar el stock de cada producto vendido
- Al registrar una compra, debe incrementar el stock de cada producto comprado
- Debe validar que exista stock suficiente antes de permitir una venta

RF-008

Actualización Automática de Inventario

ROLES INVOLUCRADOS: SISTEMA (AUTOMÁTICO)

CATEGORÍA: GESTIÓN DE INVENTARIO Y PRODUCTOS

El sistema debe actualizar automáticamente las cantidades de inventario ante entradas (compras) y salidas (ventas) de productos.

- Al registrar una venta, debe decrementar el stock de cada producto vendido
- Al registrar una compra, debe incrementar el stock de cada producto comprado
- Al anular una venta, debe reintegrar el stock de los productos
- Debe validar que exista stock suficiente antes de permitir una venta

RF-009

Registro de Ventas

ROLES INVOLUCRADOS: CAJERO, ADMINISTRADOR

CATEGORÍA: CAJA

El sistema debe permitir registrar notas de venta con la información requerida

Criterios de Aceptación:

- Debe incluir los datos del empleado
- Debe incluir los datos del cliente (NIT o consumidor final)

RF-010

Registro de Clientes

ROLES INVOLUCRADOS: CAJERO, ADMINISTRADOR

CATEGORÍA: VENTAS

El sistema debe permitir registrar y gestionar información de clientes.

Criterios de Aceptación:

- Debe incluir: nombre/razón social, NIT/DPI, dirección, teléfono, correo
- Debe validar formato de NIT/DPI
- Debe permitir crear, modificar, consultar y desactivar clientes
- Debe permitir búsqueda por: nombre, NIT

Requerimientos no funcionales

RNF-001

Diseño de Interfaz Atractivo

PRIORIDAD: MEDIA

Tipo: *Usabilidad*

El sistema debe proporcionar una interfaz gráfica visualmente agradable que mejore la experiencia del usuario y facilite la adopción del sistema.

Criterios de Aceptación:

- Debe utilizar una paleta de colores coherente y profesional
- Debe incluir iconografía consistente en todo el sistema
- Los elementos deben tener espaciado adecuado (no sobrecargado)
- Debe usar tipografía legible (tamaño mínimo 14px para texto)
- Debe incluir feedback visual para acciones del usuario (botones, hover, loading)
- Debe cumplir con principios básicos de diseño UI/UX moderno

RNF-002

Interfaz Intuitiva

PRIORIDAD: ALTA

Tipo: *Usabilidad*

El sistema debe ser fácil de usar y requerir mínima capacitación, con navegación clara y flujos de trabajo lógicos.

Criterios de Aceptación:

- Un usuario nuevo debe poder realizar operaciones básicas con menos de 30 minutos de capacitación
- Los elementos de navegación deben ser consistentes en todas las pantallas
- Debe usar etiquetas y textos claros en español sin tecnicismos innecesarios
- Los campos de formulario deben tener placeholders y mensajes de ayuda
- Los errores deben mostrar mensajes claros sobre qué hacer para resolverlos

RNF-003

Diseño Responsivo

PRIORIDAD: ALTA

Tipo: *Usabilidad / Compatibilidad*

La interfaz debe adaptarse correctamente a diferentes tamaños de pantalla y dispositivos (desktop, tablet, móvil).

Criterios de Aceptación:

- Debe funcionar correctamente en resoluciones desde 320px hasta 1920px de ancho
- Debe adaptarse a orientación vertical y horizontal en tablets/móviles
- Los elementos deben reordenarse apropiadamente en pantallas pequeñas
- Texto y botones deben ser legibles y tocables en dispositivos móviles (mínimo 44x44px)
- Las tablas deben ser navegables en móvil (scroll horizontal o cards)
- No debe requerir zoom para leer contenido en ningún dispositivo

RNF-004

Transmisión Segura de Token de Refresco

PRIORIDAD: ALTA

Tipo: *Seguridad*

Los tokens de refresco deben transmitirse y almacenarse como cookies HTTP-only para prevenir acceso desde JavaScript y ataques XSS.

Criterios de Aceptación:

- El refresh token debe enviarse como cookie con atributo HttpOnly
- Debe incluir atributo Secure (solo transmitirse por HTTPS)
- Debe incluir atributo SameSite=Strict o SameSite=Lax

RNF-005

Expiración de Token de Refresco

PRIORIDAD: ALTA

Tipo: *Seguridad*

Los tokens de refresco (refresh tokens) deben expirar después de 30 días para limitar el período de validez de sesiones.

Criterios de Aceptación:

- El refresh token debe incluir claim de expiración de 30 días desde emisión
- Debe invalidarse automáticamente después de 30 días
- Debe requerir nuevo inicio de sesión después de expiración
- El contador de 30 días NO debe reiniciarse con cada uso (absolute expiration)

Casos de uso

Actores

Cajero

Usuario responsable de las operaciones de venta y atención al cliente.

Bodeguero

Usuario responsable de la gestión de inventario, productos y compras.

Gerente

Usuario con privilegios administrativos completos. Hereda todos los permisos de Cajero y Bodeguero, además de funciones exclusivas de gestión.

Especificación

Iniciar Sesión

Actor Principal: Cajero, Bodeguero, Gerente
Objetivo: Autenticar al usuario en el sistema
Precondiciones: El usuario debe estar registrado en el sistema
Postcondiciones: El usuario accede al sistema según su rol

Flujo Principal:

1. El usuario ingresa su nombre de usuario
2. El usuario ingresa su contraseña
3. El sistema valida las credenciales
4. El sistema identifica el rol del usuario
5. El sistema redirige al usuario a su panel correspondiente
6. El caso de uso finaliza exitosamente

Flujos Alternativos:

- **Credenciales inválidas**
    - El sistema muestra mensaje de error
    - El sistema solicita reingresar credenciales
    - Retorna al paso 1

Administrar Clientes

Actor Principal: Cajero, Gerente
Objetivo: Crear, modificar, consultar o eliminar información de clientes
Precondiciones: El usuario debe haber iniciado sesión con rol de Cajero o Gerente
Postcondiciones: La información del cliente queda registrada o actualizada en el sistema

Flujo Principal:

1. El usuario accede al módulo de clientes
2. El sistema muestra la lista de clientes registrados
3. El usuario selecciona la acción deseada (Crear, Modificar, Consultar)
4. El sistema presenta el formulario o información correspondiente
5. El usuario completa o modifica los datos requeridos
6. El usuario confirma la operación
7. El sistema valida los datos
8. El sistema guarda los cambios
9. El sistema muestra mensaje de confirmación
10. El caso de uso finaliza exitosamente

Flujos Alternativos:

- **Crear nuevo cliente**
    - El sistema presenta formulario en blanco
    - Continúa en paso 5
- **Consultar cliente**
    - El sistema muestra la información del cliente seleccionado

- El caso de uso finaliza

Registrar Nota de Venta

Actor Principal: Cajero, Gerente
Objetivo: Registrar una transacción de venta de productos
Precondiciones:

- El usuario debe haber iniciado sesión con rol de Cajero o Gerente
- Debe haber productos disponibles en inventario

Postcondiciones:

- La venta queda registrada en el sistema
- El inventario se actualiza automáticamente

Flujo Principal:

1. El usuario accede al módulo de ventas
2. El sistema muestra el formulario de nueva venta
3. El usuario selecciona o busca el cliente
4. El usuario agrega productos al carrito
5. El sistema muestra el precio y disponibilidad de cada producto
6. El usuario especifica la cantidad deseada
7. El sistema calcula el subtotal
8. El usuario continúa agregando productos o finaliza la selección
9. El sistema calcula el total de la venta
10. El usuario confirma la venta
11. El sistema registra la transacción
12. El sistema actualiza el inventario
13. El sistema genera el comprobante de venta
14. El caso de uso finaliza exitosamente

Flujos Alternativos:

- **Producto sin stock suficiente**
    - El sistema muestra mensaje de advertencia
    - El sistema indica la cantidad disponible
    - Retorna al paso 6

Registrar Nota de Compra

Actor Principal: Bodeguero, Gerente
Objetivo: Registrar el ingreso de productos al inventario mediante una compra a proveedores
Precondiciones: El usuario debe haber iniciado sesión con rol de Bodeguero o Gerente
Postcondiciones:

- La compra queda registrada en el sistema
- El inventario se incrementa con los productos adquiridos

Flujo Principal:

1. El usuario accede al módulo de compras
2. El sistema muestra el formulario de nueva compra
3. El usuario ingresa datos del proveedor
4. El usuario ingresa la fecha de compra
5. El usuario agrega productos a la nota de compra
6. El usuario especifica cantidad y precio de compra por producto
7. El sistema calcula el subtotal por producto
8. El sistema calcula el total de la compra
9. El usuario confirma el registro
10. El sistema guarda la nota de compra
11. El sistema actualiza el inventario
12. El sistema muestra mensaje de confirmación
13. El caso de uso finaliza exitosamente

Administrar Paquetes

Actor Principal: Bodeguero, Gerente
Objetivo: Crear, modificar o eliminar paquetes de productos (combos o kits)
Precondiciones: El usuario debe haber iniciado sesión con rol de Bodeguero o Gerente
Postcondiciones: El paquete queda registrado o actualizado en el sistema

Flujo Principal:

1. El usuario accede al módulo de paquetes
2. El sistema muestra la lista de paquetes existentes
3. El usuario selecciona la acción deseada (Crear, Modificar, Eliminar)
4. El usuario ingresa o modifica la información del paquete
5. El usuario selecciona los productos que componen el paquete
6. El usuario especifica las cantidades de cada producto
7. El usuario define el precio del paquete
8. El usuario confirma la operación
9. El sistema valida la información
10. El sistema guarda los cambios
11. El sistema muestra mensaje de confirmación
12. El caso de uso finaliza exitosamente

Administrar Productos

Actor Principal: Bodeguero, Gerente
Objetivo: Crear, modificar, consultar o eliminar información de productos
Precondiciones: El usuario debe haber iniciado sesión con rol de Bodeguero o Gerente
Postcondiciones: La información del producto queda registrada o actualizada en el sistema

Flujo Principal:

1. El usuario accede al módulo de productos

2. El sistema muestra el catálogo de productos
3. El usuario selecciona la acción deseada (Crear, Modificar, Consultar, Eliminar)
4. El sistema presenta el formulario correspondiente
5. El usuario completa o modifica los datos del producto (nombre, descripción, precio, categoría, stock mínimo)
6. El usuario confirma la operación
7. El sistema valida los datos
8. El sistema guarda los cambios
9. El sistema muestra mensaje de confirmación
10. El caso de uso finaliza exitosamente

Gestionar Empleados

Actor Principal: Gerente
Objetivo: Crear, modificar, consultar o eliminar cuentas de empleados del sistema
Precondiciones: El usuario debe haber iniciado sesión con rol de Gerente
Postcondiciones: La información del empleado queda registrada o actualizada en el sistema

Flujo Principal:

1. El usuario accede al módulo de empleados
2. El sistema muestra la lista de empleados
3. El usuario selecciona la acción deseada (Crear, Modificar, Consultar, Eliminar)
4. El usuario ingresa o modifica los datos del empleado (nombre, usuario, contraseña, rol)
5. El usuario asigna el rol correspondiente (Cajero, Bodeguero, Gerente)
6. El usuario confirma la operación
7. El sistema valida los datos
8. El sistema guarda los cambios
9. El sistema muestra mensaje de confirmación

10. El caso de uso finaliza exitosamente

Ver Reportes

Actor Principal: Gerente
Objetivo: Consultar reportes estadísticos y financieros del negocio
Precondiciones: El usuario debe haber iniciado sesión con rol de Gerente
Postcondiciones: El sistema muestra la información solicitada

Flujo Principal:

1. El usuario accede al módulo de reportes
2. El sistema muestra las opciones de reportes disponibles
3. El usuario selecciona el tipo de reporte (ventas, compras, inventario, empleados)
4. El usuario especifica el rango de fechas o período
5. El usuario aplica filtros adicionales si es necesario
6. El sistema procesa la consulta
7. El sistema genera el reporte
8. El sistema muestra los resultados en pantalla
9. El usuario puede exportar el reporte (opcional)
10. El caso de uso finaliza exitosamente

Cerrar Sesión

Actor Principal: Cajero, Bodeguero, Gerente
Objetivo: Finalizar la sesión del usuario en el sistema
Precondiciones: El usuario debe haber iniciado sesión
Postcondiciones: La sesión del usuario se cierra y se retorna a la pantalla de inicio de sesión

Flujo Principal:

1. El usuario selecciona la opción "Cerrar sesión"
2. El sistema solicita confirmación
3. El usuario confirma el cierre de sesión

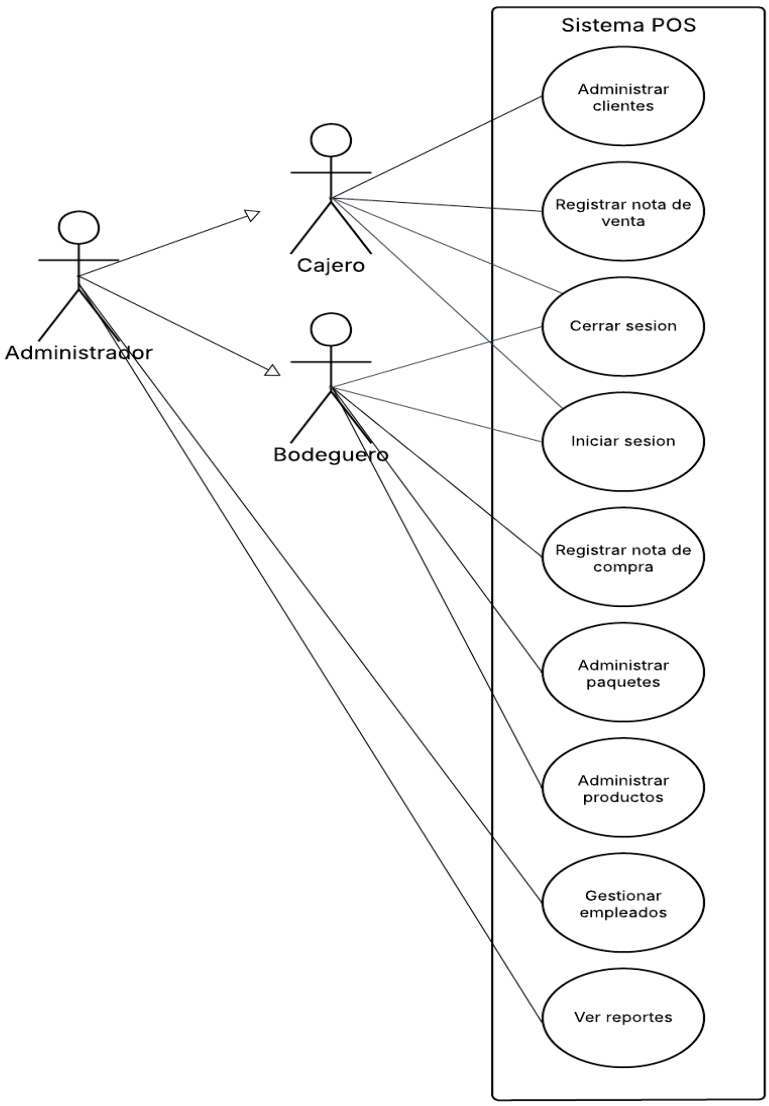
4. El sistema finaliza la sesión
5. El sistema redirige a la pantalla de inicio de sesión
6. El caso de uso finaliza exitosamente

Diagramas

Todos los diagramas pueden verse en Lucichart para más detalles:

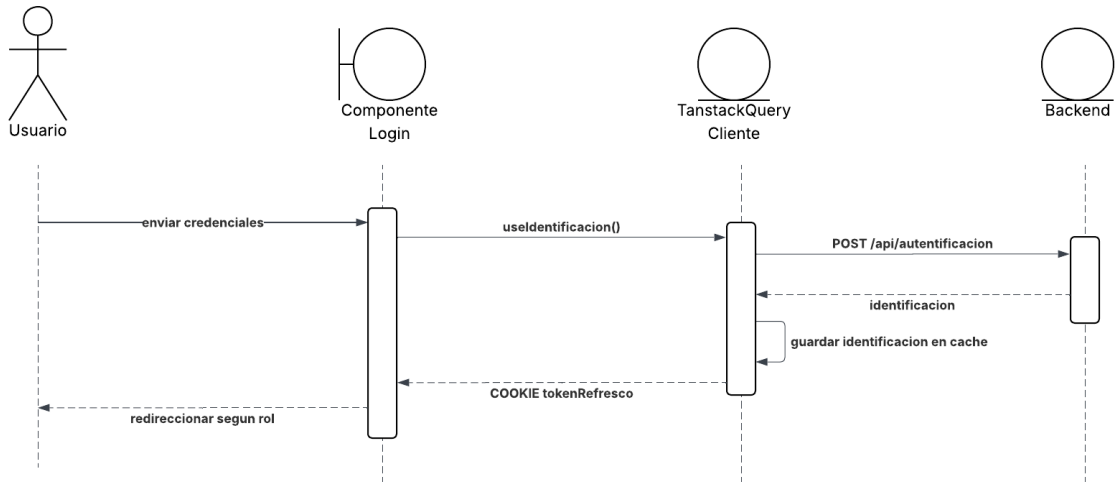
https://lucid.app/folder/invitations/accept?invitationId=inv_5e4301a0-c379-44c3-97d2-90ca380fe06a

Casos de uso

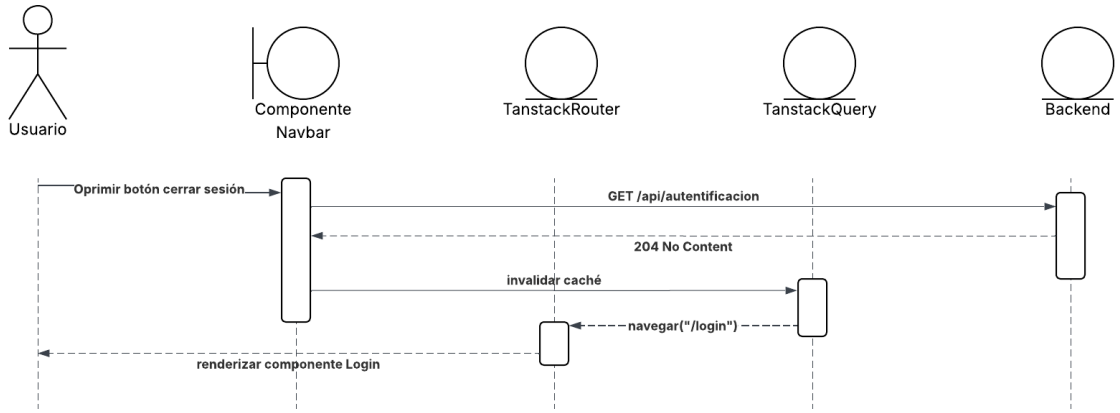


Secuencias

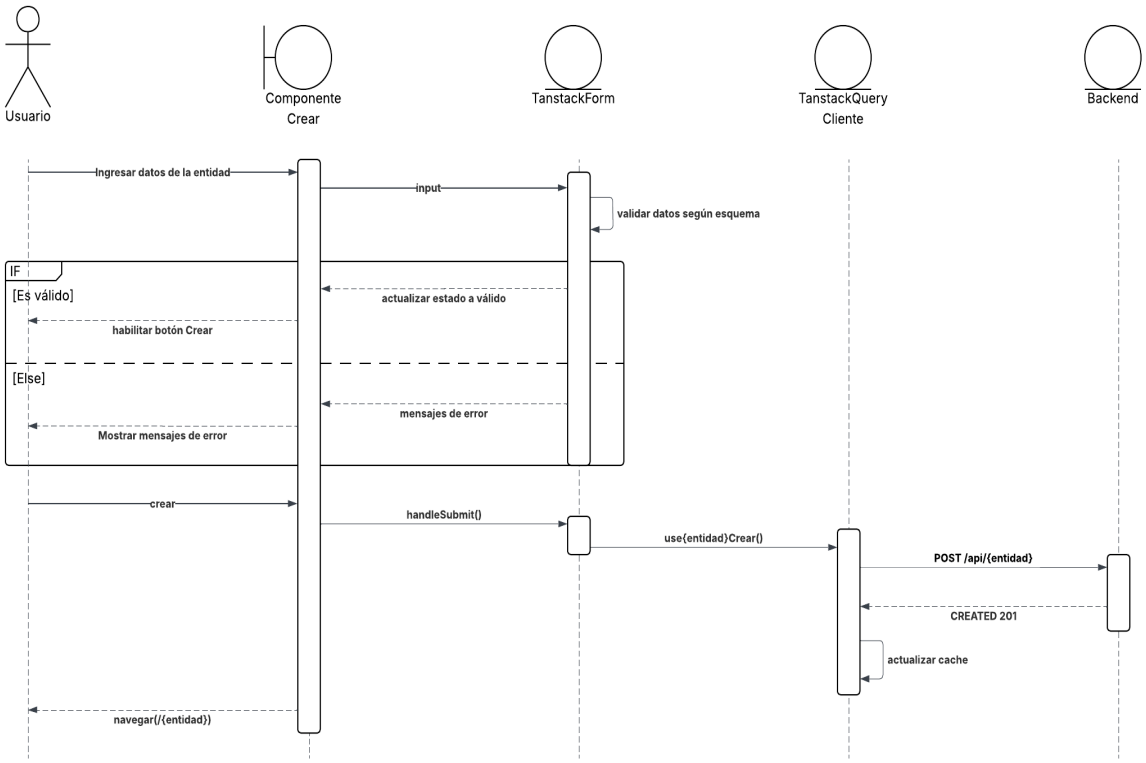
Iniciar sesión



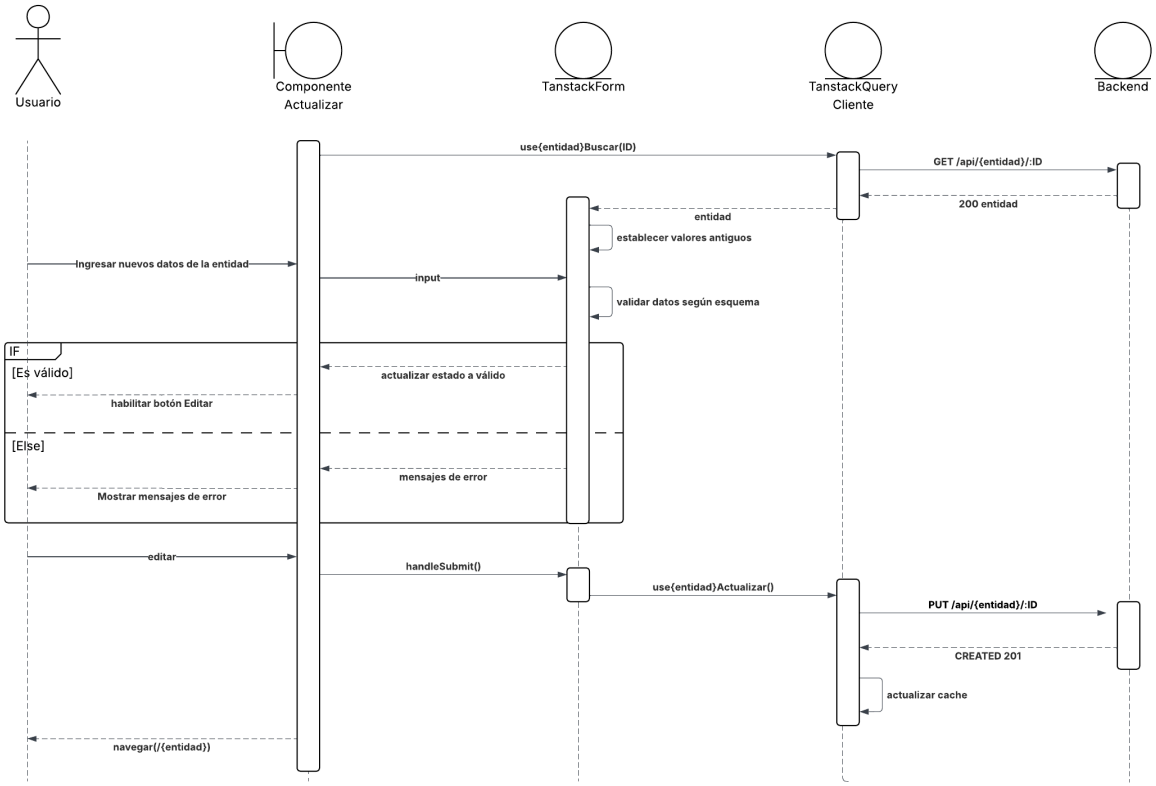
Cerrar sesión



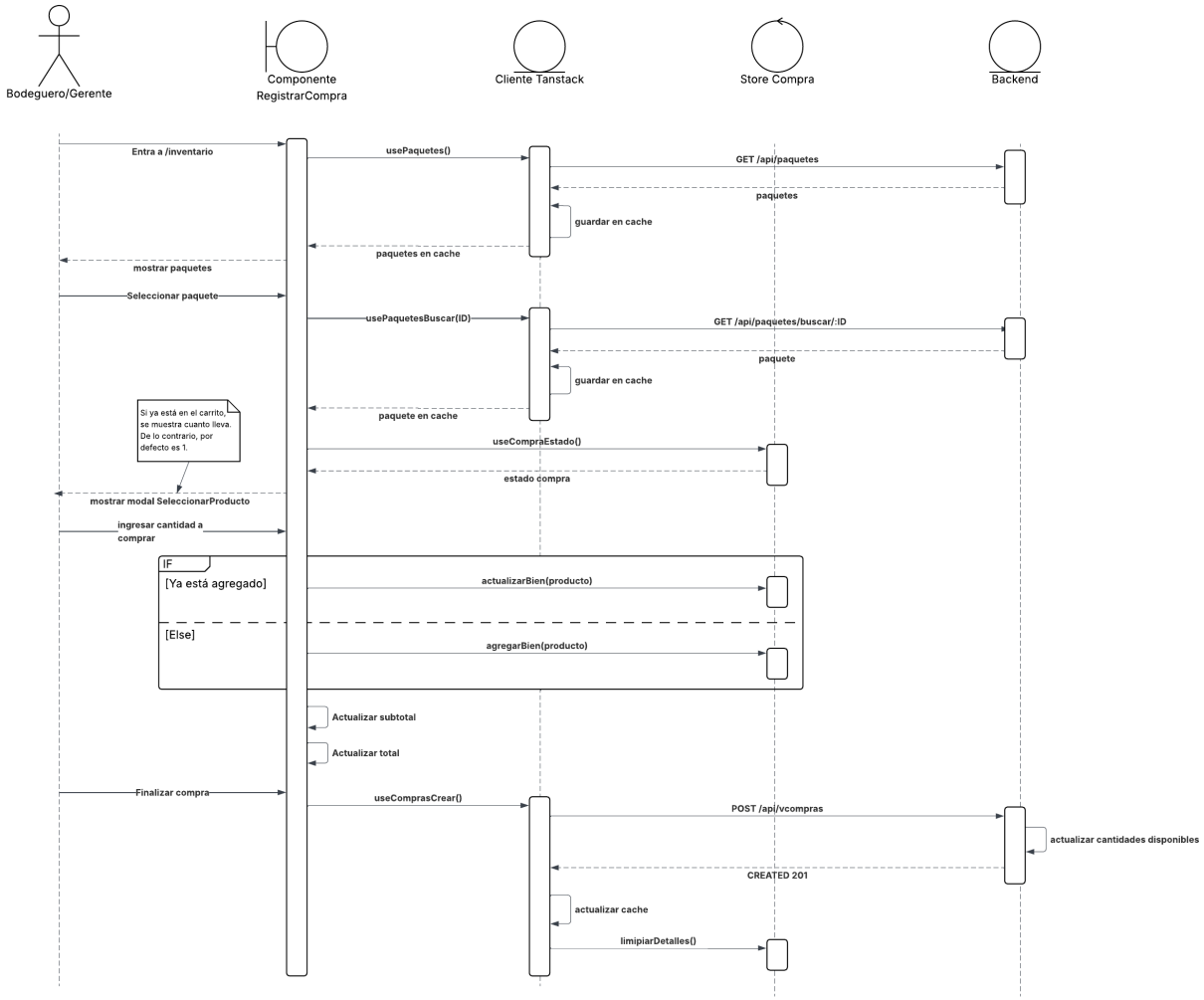
Crear entidad



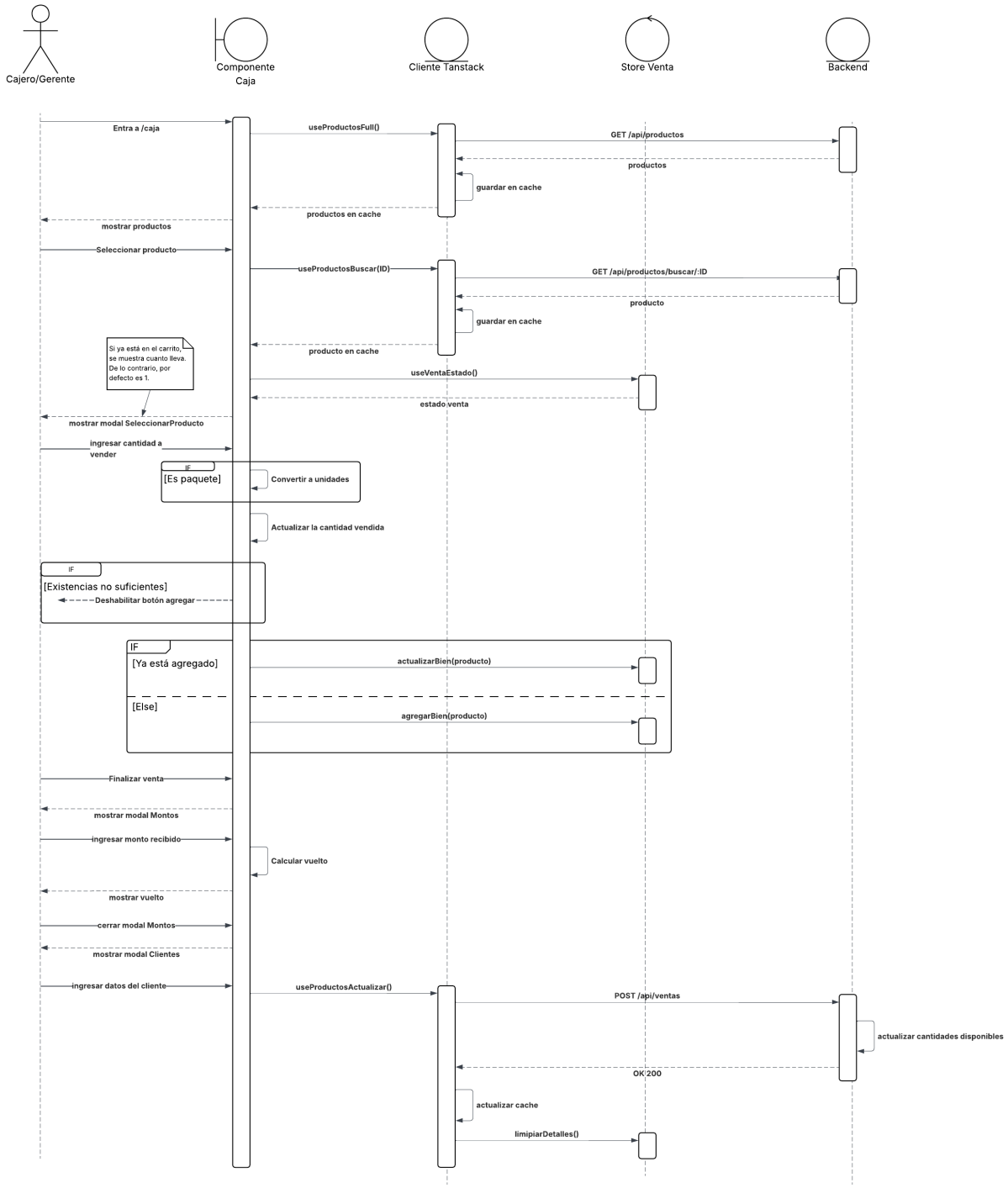
Editar entidad



Nota de compra

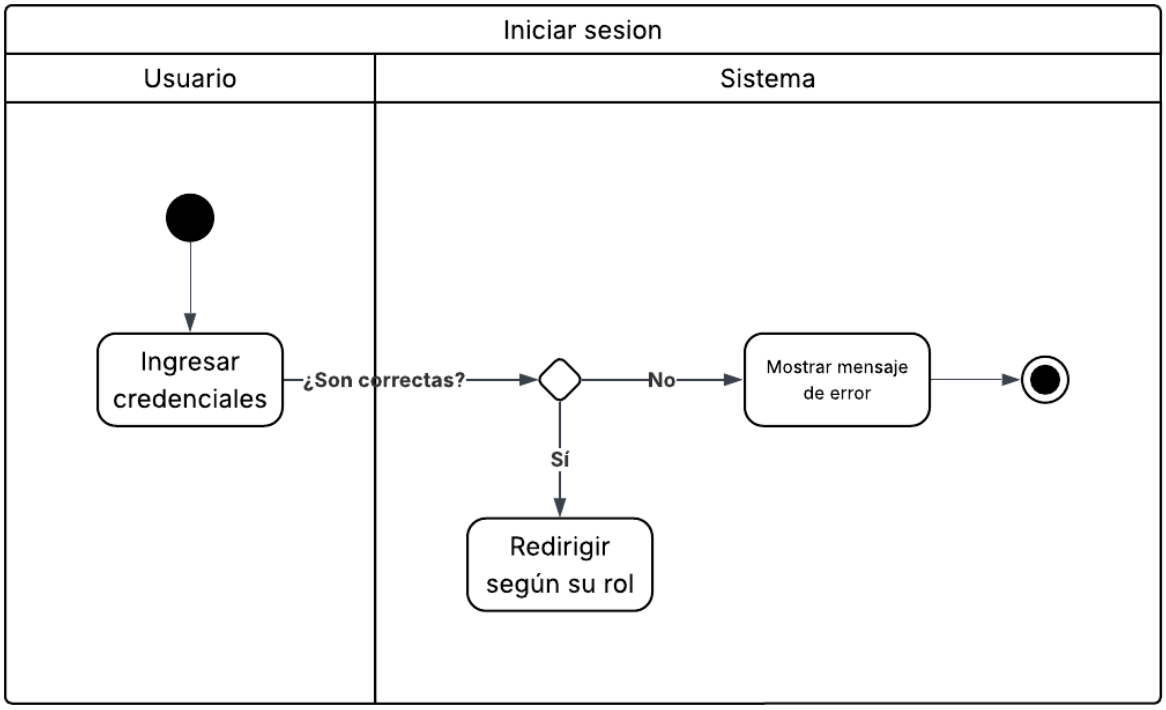


Nota de venta

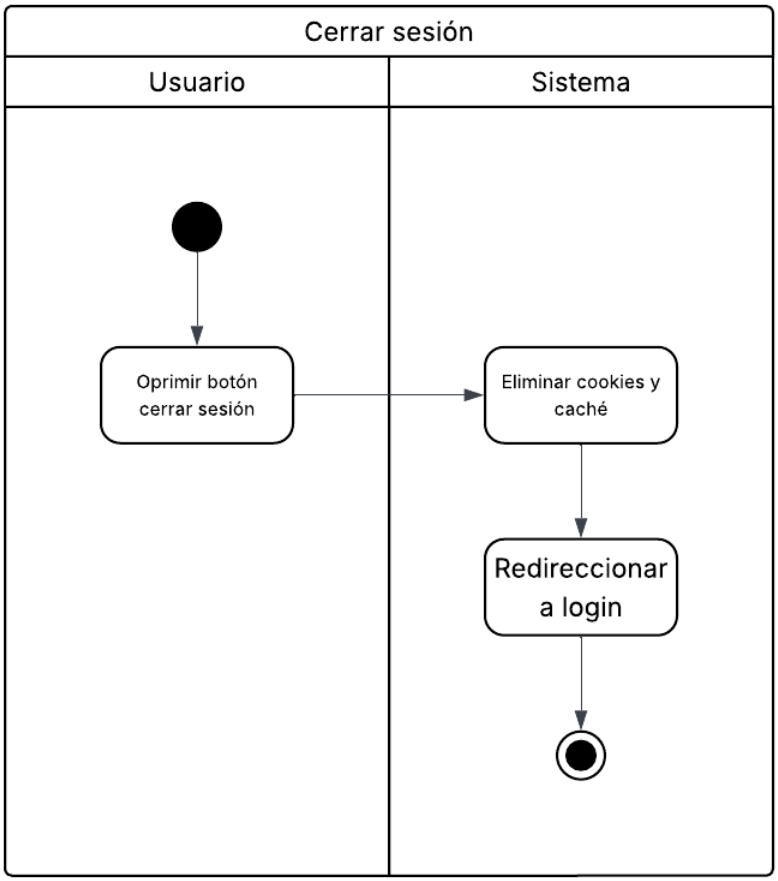


Actividades

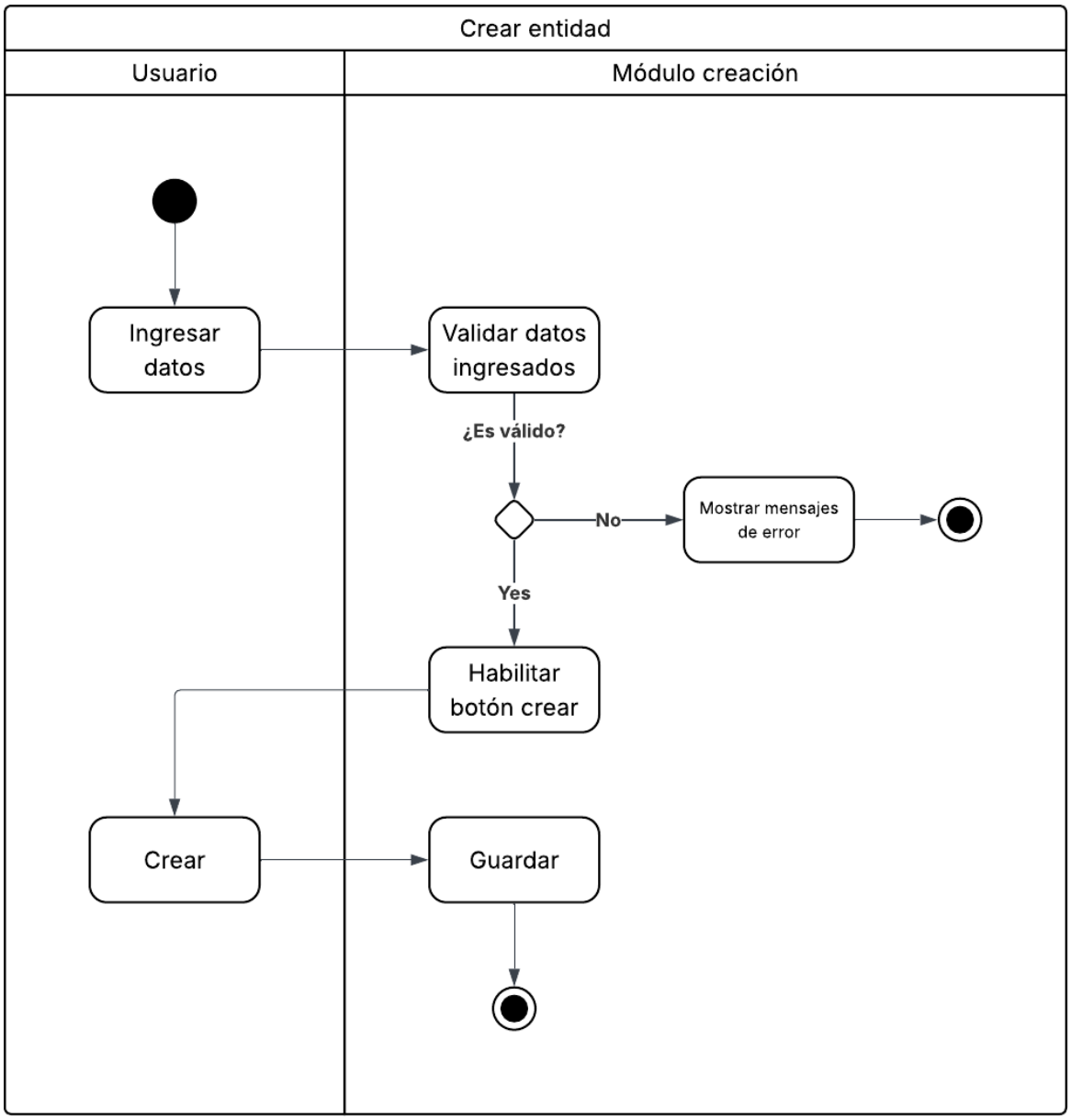
Iniciar session



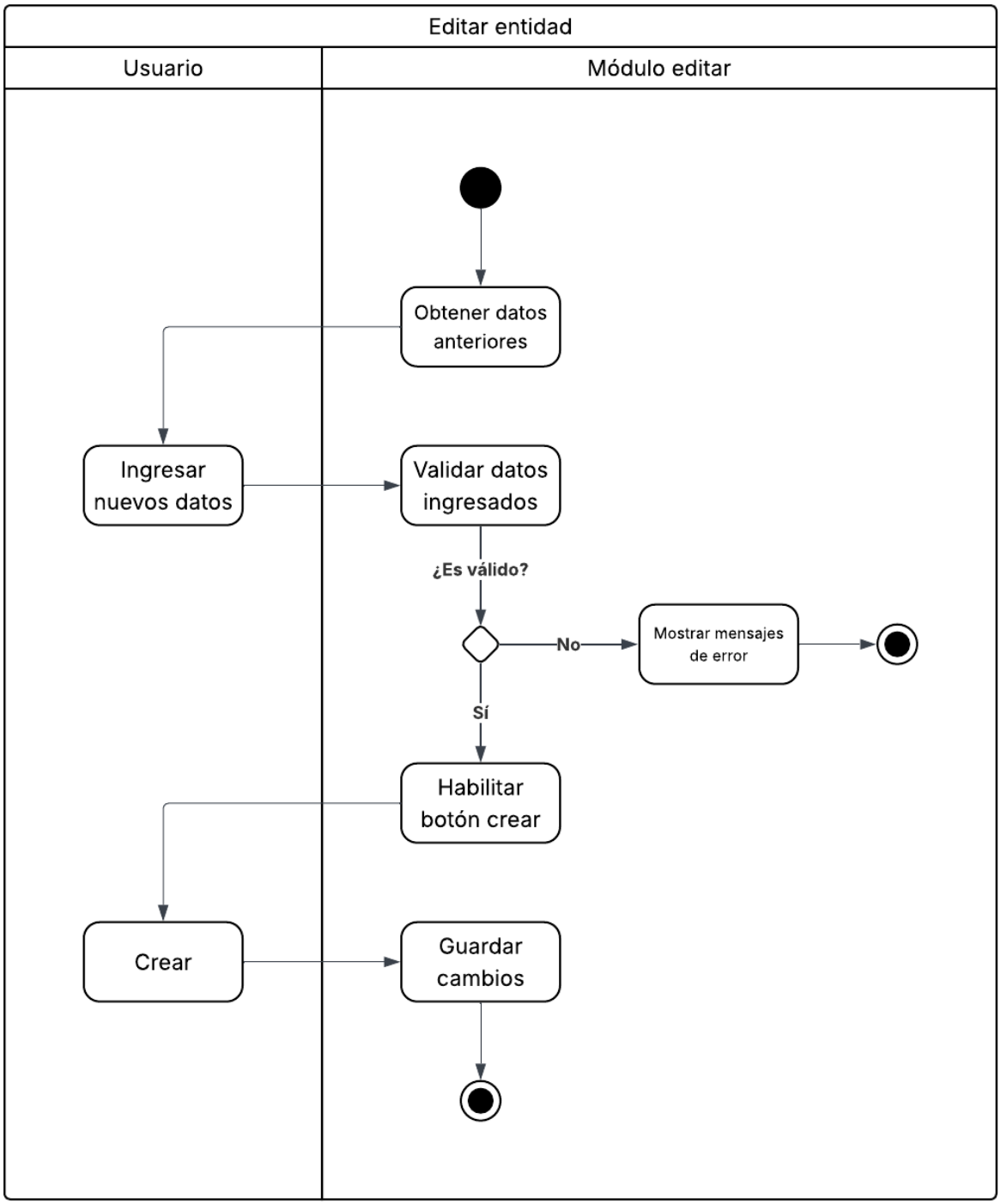
Cerrar session



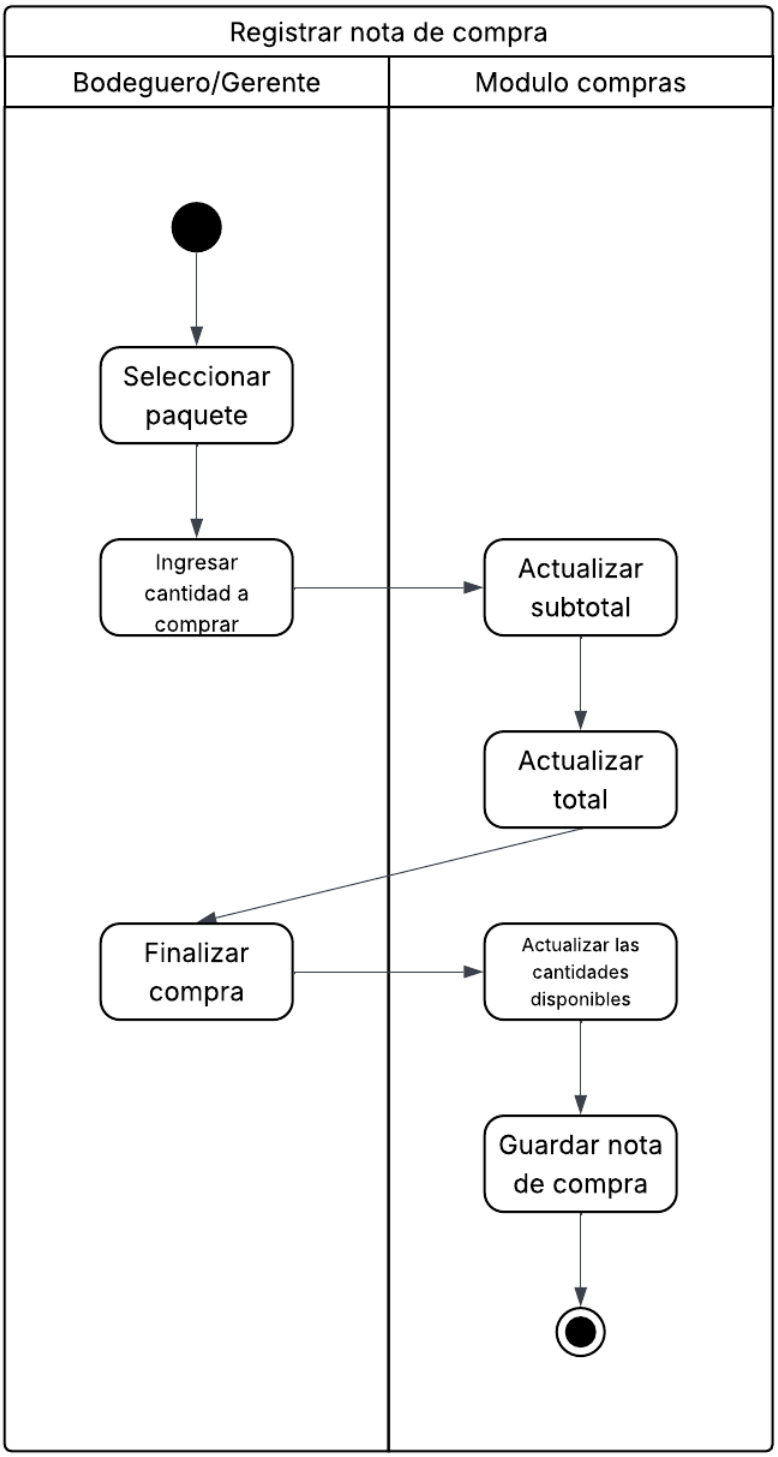
Crear entidad



Editar entidad



Nota de compra



Nota de venta

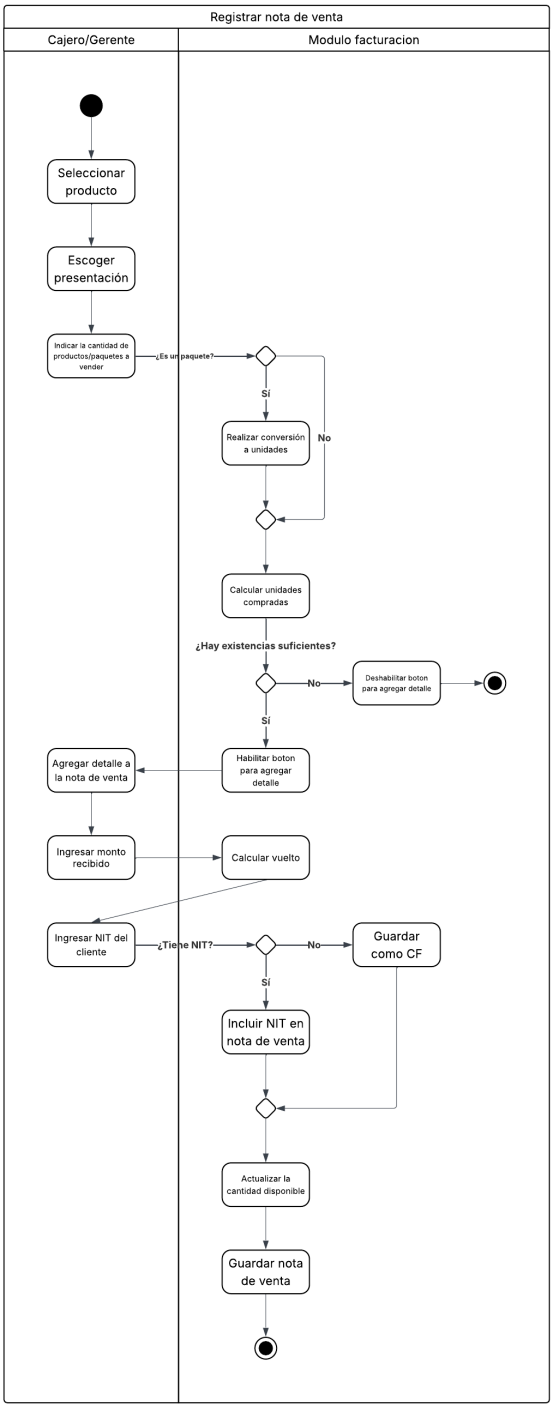


Diagrama de clases

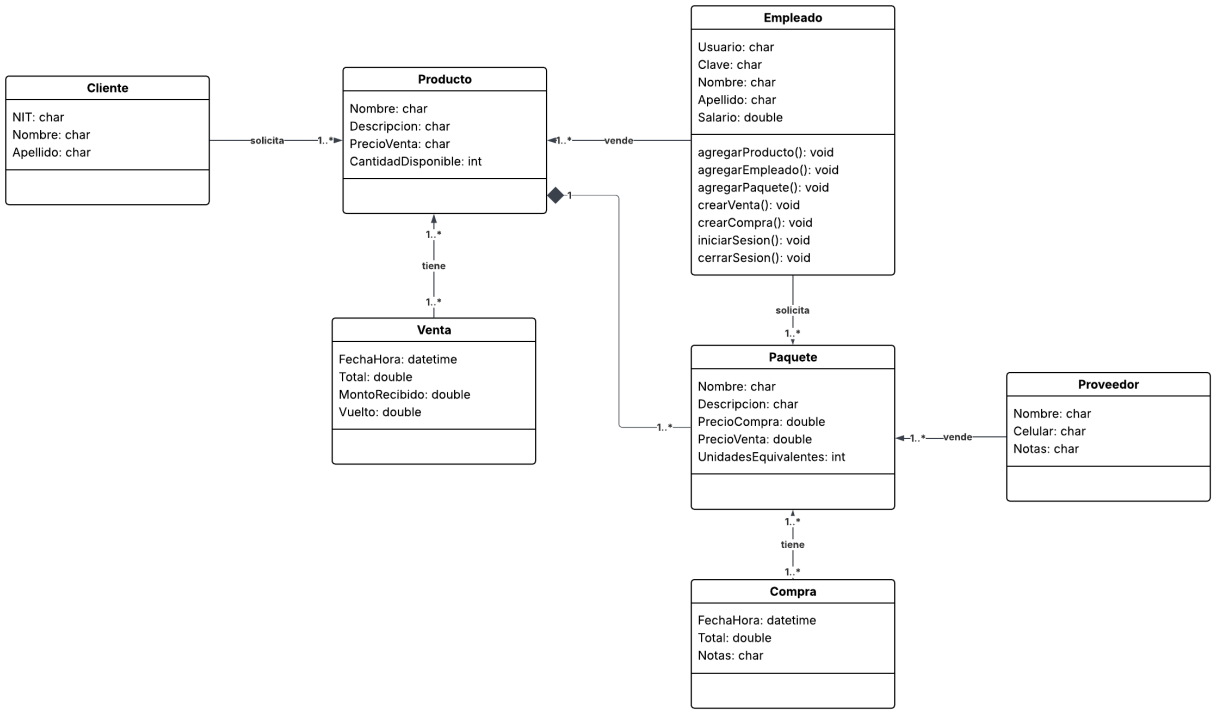
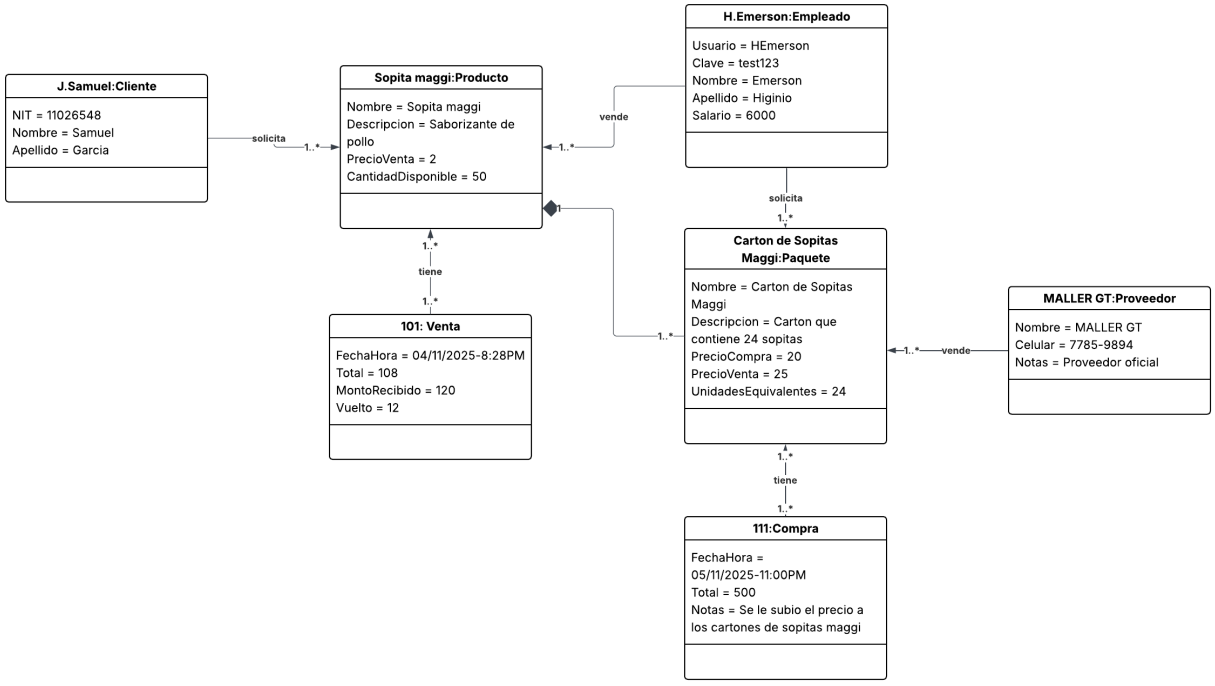


Diagrama de objetos



Manual técnico

Arquitectura

El sistema implementa una arquitectura Cliente-Servidor de tres capas con API REST, que separa claramente las responsabilidades entre la interfaz de usuario, la lógica de negocio y el almacenamiento de datos.

Capa de Presentación (Frontend)

El frontend es una interfaz desarrollada en React, que se ejecuta completamente en el navegador del usuario. Esta capa es responsable de:

- Presentar la interfaz gráfica al usuario
- Capturar y validar las entradas del usuario
- Realizar peticiones HTTP a la API REST del backend
- Renderizar dinámicamente la información recibida
- Gestionar la navegación y el estado de la aplicación

El frontend se comunica con el backend exclusivamente mediante llamadas HTTP a endpoints REST, enviando y recibiendo datos en formato JSON.

Capa de Lógica de Negocio (Backend)

El backend es una API RESTful desarrollada en expressjs, desplegada de forma independiente en la nube utilizando Azure Container Apps. Esta capa maneja:

- Autenticación y autorización de usuarios mediante JWT
- Validación de datos del lado del servidor
- Implementación de reglas de negocio
- Procesamiento de transacciones
- Generación de reportes
- Comunicación con la base de datos

El backend expone endpoints HTTP que el frontend consume, siguiendo los principios REST (GET para consultas, POST para creación, PUT/PATCH para actualización, sin DELETE para eliminación debido a integridad referencial). Es una API Restful por lo que la mayoría

de controladores hacen lo mismo: en GET retornar los recursos, en POST crea y valida la calidad de datos ingresados y PUT que actualiza y valida datos. Los mecanismos son explicados en detalle en la sección de diagramas.

Backend

Lenguaje: Typescript

Base de datos: MySQL

Framework: Express js

ORM: Prisma

Autentificación: JWT

Validador de esquemas: ZOD

Base de datos

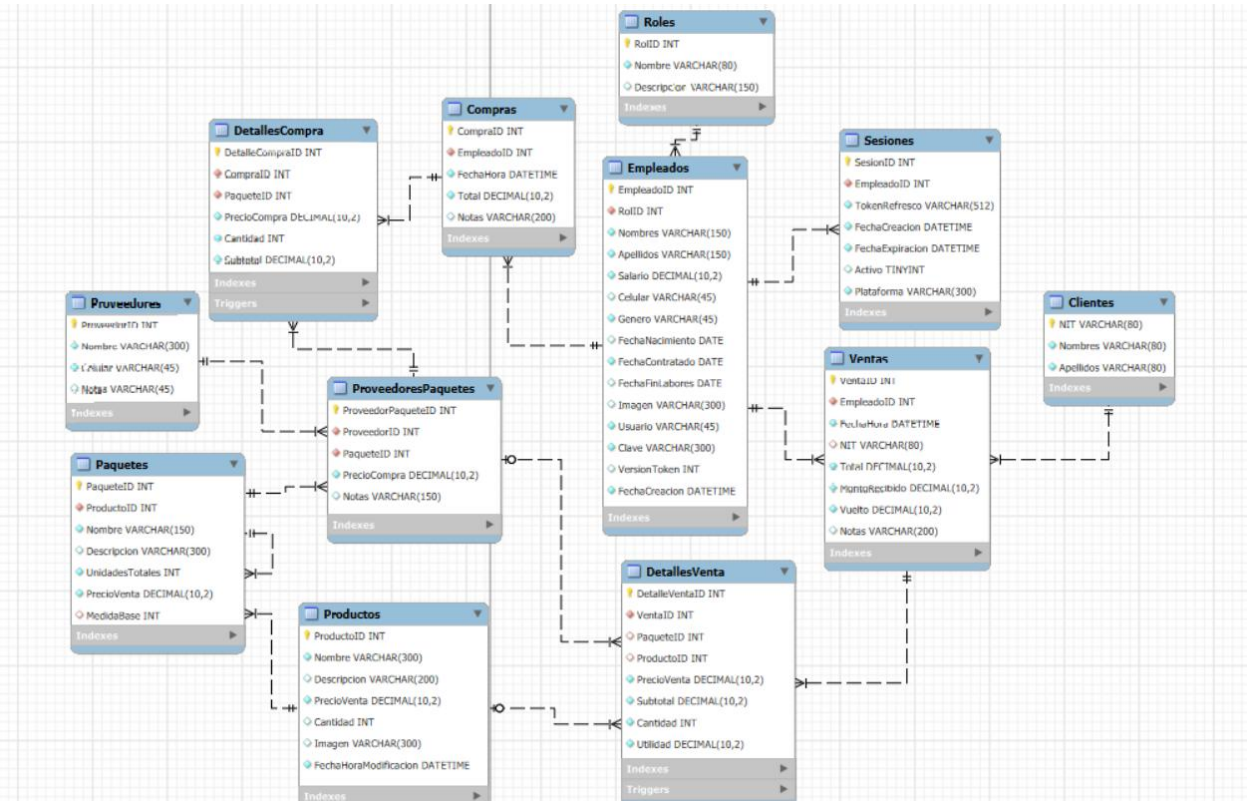


TABLA	DESCRIPCIÓN

EMPLEADOS	Contiene la información laboral y personal de todos los empleados. En esta tabla también se incluyen las credenciales del usuario; la clave está encriptada
SESIONES	Almacena datos útiles para crontolar el acceso a la aplicación.
ROLES	Describe y enlista los roles del sistema.
PRODUCTOS	Lista información esencial sobre cada producto. La columna Cantidad lleva el inventario del producto.
PAQUETES	Representa los productos al mayoreo. Este es la “entrada” ya que actualiza la Cantidad disponible de cada producto.
PROVEEDORESPAQUETES	Es una tabla de unión que indica los distintos precios a los que los proveedores ofrecen el mismo producto.
PROVEEDORES	Guarda información de contacto sobre los proveedores
COMPRAS	En esta tabla se guardan todos los registros de las notas de compras.
DETALLESCOMPRAS	Contiene información del detalle de cada compra.
VENTAS	Guarda las notas de ventas hechas.
DETALLESVENTA	Muestro los detalles de una venta incluyendo la utilidad de cada ítem.
CLIENTES	Guarda información básica de los clientes.

Recurso	Ruta	Método	Descripción
Autentificación	/autentificación	GET	Elimina las cookies de sesión e invalida la sesión en la base de datos
		POST	Valida las credenciales del usuario y, de ser correctas, registra la sesión en la base de datos y envía las cookies de sesión
		PUT	Actualiza la calve de los empleados.
Identificación	/identificación	GET	Retorne la información del usuario para usarla dentro del sistema.

Reportes	/reportes	POST	Retorna los siguientes reportes: • Ventas de hoy • Balance (gastos – utilidad) • Ganacia del mes • Top 10 productos • Ventas por mes
Clientes	/clientes	GET	Retorna todos los clientes
		POST	Crea un nuevo cliente
Empleados	/empleados	GET	Retorna la información y los roles de todos los empleados.
		POST	Permite crear un nuevo empleado.
	/empleados/buscar/:ID	GET	Retorna la información y rol de un usuario a excepción de la clave.
		PUT	Actualiza los datos de un empleado.
Roles	/roles	GET	Retorne la información de los roles.
Productos	/productos	GET	Retorna la información de los productos incluyen la información de sus paquetes.
		POST	Permite crear un nuevo producto
	/productos/buscar/:ID	GET	Retorna un producto y sus paquetes.
		PUT	Permite actualizar un producto
Paquetes	/paquetes	GET	Permite retornar todos los paquetes junto con la información de sus proveedores
		POST	Permite crear un nuevo paquete
	/paquetes/buscar/:ID	GET	Retorna un paquete con la información de sus proveedores
		PUT	Actualiza los datos de un paquete
Proveedores	/proveedores	GET	Retorne todos los proveedores.
		POST	Crear un proveedor
	/proveedores/buscar/:ID	GET	Retorne los datos de un proveedores

		PUT	Actualiza los datos de un proveedor
Ventas	/ventas	GET	Retorne todos las ventas justo con sus detalles
		POST	Es una transacción y permite registrar notas de venta
	/ventas/buscar/:ID	GET	Retorne los datos y detalles de una venta
Compras	/compras	GET	Retorna todas las compras justo con sus detalles
		POST	Permite registrar notas de compra

Autentificación

Cuando un usuario inicia sesión, el sistema valida sus credenciales contra la base de datos, comparando la contraseña ingresada con el hash almacenado mediante bcrypt. Si las credenciales son correctas, se generan dos tokens JWT distintos: un token de acceso con validez de 15 minutos que contiene el ID del empleado, su rol y una versión del token, y un token de refresco con validez de 30 días que contiene únicamente el ID del empleado y la versión del token. Ambos tokens se envían al cliente como cookies HTTP-only, lo que significa que no son accesibles desde JavaScript y por tanto están protegidos contra ataques XSS. Adicionalmente, el token de refresco se almacena en la tabla de sesiones de la base de datos junto con información sobre el dispositivo desde el cual se realizó el inicio de sesión, permitiendo así un control granular de las sesiones activas.

Para cada petición a endpoints protegidos, el middleware de verificación JWT extrae automáticamente el token de acceso de las cookies del request. Este middleware verifica la firma digital del token, su fecha de expiración y crucialmente, compara la versión del token contenida en el JWT con la versión almacenada en la base de datos para el empleado correspondiente. Esta verificación de versión es fundamental para el mecanismo de invalidación de tokens, ya que permite revocar inmediatamente todos los tokens emitidos anteriormente cuando un usuario cambia su contraseña. Si todas las validaciones son exitosas, el middleware adjunta la información decodificada del token al objeto request y permite que la petición continúe su flujo normal hacia el controlador correspondiente.

Frontend

Lenguaje: Typescript

Framework: React

Framework estilos: Tailwindcss

Validacion de esquemas: ZOD

Cache: Tanstack query

Router: Tanstack router

Formularios: Tanstack form

Estilos

Los estilos se encuentran en el index.css del proyecto debido a la poco necesidad de utilizar clases debido al enfoque de utilidad primero de tailwindcss. El sistema utiliza una paleta de colores definida y consistente enfocada en el modo claro. Todos los módulos fueron diseños pensando en la responsividad, así que todos pueden ser vistos y utilisados normalmente desde celulares. La responsividad está implementada mediante los breakpoints estándar de Tailwind (sm, md, lg, xl) aplicados directamente en las clases utility, permitiendo que los componentes se adapten fluidamente a diferentes tamaños de pantalla. Por ejemplo, el contenido principal agrega margen lateral solo en pantallas medianas y superiores para compensar el espacio ocupado por el sidebar mediante sm:ml-panel-menu-ancho, mientras que en móviles ocupa el ancho completo. Las grids ajustan dinámicamente el número de columnas según el viewport disponible, mostrando una columna en móvil y escalando hasta cinco columnas en pantallas extra grandes, asegurando que el contenido siempre se visualice apropiadamente sin scroll horizontal innecesario.

Ruta	Contenido
Index	Dashboard
/empleados	Lista de empleados
/empleados/:ID	Detalles de un empleado
/empleados/editar/:ID	Modificación de un empleado.
/inventario/index	Registro de nota de compras
/inventario/productos	Lista de los productos del sistema
/inventario/productos/editar/:ID	Formulario de edición de productos

/inventario/producto/crear	Formulario para crear un producto
/inventario/paquetes/crear	Formulario para crear paquete
/inventario/proveedores	Mustra la información de los proveedores
/inventario/proveedores/editar	Formulario para editar los datos de los proveedores
/inventario/proveedores/crear	Formulario para crear un proveedor
/caja	Muestra todos los productos y sus presentaciones además de agregar interfaces para crear la nota de venta
/historial/ventas	Muestra las ventas y sus detalles
/historial/compras	Muestra las compras y sus detalles
/login	Formulario para ingresar las credenciales de un usuario

Protección de Rutas

El frontend del sistema utiliza TanStack Router. Para gestionar la navegación, implementando un esquema de rutas protegidas basado en autenticación y control de acceso por roles. La estructura de rutas está organizada jerárquicamente, donde todas las funcionalidades principales del sistema requieren que el usuario haya iniciado sesión, exceptuando únicamente la ruta /login que es de acceso público.Se utilizan ruta layout que implementa dos niveles de protección: primero verifica la autenticación del usuario mediante el hook beforeLoad, que se ejecuta antes de renderizar cualquier componente hijo, y segundo valida los permisos específicos según el rol del usuario en el componente mismo. Una vez superada la validación de autenticación, cada ruta implementa su propia validación de permisos mediante la función verificarPermisos, que recibe un arreglo de roles autorizados y el rol actual del usuario.

Formularios

El sistema implementa un patrón consistente de manejo de formularios utilizando TanStack Form. Todos los formularios del sistema siguen una arquitectura de tres capas claramente definida: el custom hook que orquesta la lógica del formulario, los componentes de input reutilizables que encapsulan comportamiento común, y la página que compone estos elementos en un formulario completo.

La capa de lógica está encapsulada en custom hooks como useFormularioCrearEmpleado, que configura el formulario con valores iniciales tipados, integra validación mediante schemas Zod que se ejecutan en el evento onBlur para

proporcionar feedback inmediato al usuario, y conecta el submit del formulario con las mutaciones de React Query que envían los datos al backend.

El estado de mutation.isPending se utiliza para deshabilitar el botón de envío y mostrar indicadores de carga, previniendo envíos duplicados mientras la petición está en proceso.

Consultas

El sistema utiliza TanStack Query como capa de gestión de estado del servidor que se integra perfectamente con TanStack Table para crear un flujo de datos unidireccional y optimizado. Esta combinación permite que las tablas siempre muestren datos frescos sincronizados con el backend, implementen caché inteligente para mejorar el rendimiento, y reaccionen automáticamente a cambios en los datos sin necesidad de recargas manuales o gestión compleja de estado.

Cada entidad del sistema tiene su propio conjunto de custom hooks basados en TanStack Query que encapsulan todas las operaciones relacionadas con esa entidad. Para ventas, useVentas implementa una query que obtiene el listado completo de ventas desde el backend, mientras que useDetallesVentasBuscar obtiene los detalles específicos de una venta individual mediante su ID. Estas queries se identifican mediante queryKey, un arreglo que funciona como identificador único para el caché de Query, donde claves simples como ["ventas"] representan colecciones completas y claves compuestas como ["ventas", id] representan recursos específicos dentro de esa colección. Esta jerarquía de claves permite invalidaciones granulares donde se puede refrescar todo el listado sin afectar los detalles en caché, o viceversa.

El patrón se replica consistentemente para todas las entidades del sistema: empleados tienen useEmpleados y useTablaEmpleados, productos tienen useProductos y useTablaProductos, proveedores tienen useProveedores y useTablaProveedores, y así sucesivamente.

Manejador de estado

Los módulos de compra y venta representan la funcionalidad más compleja del sistema debido a que deben mantener estado local transaccional que persiste durante toda la sesión de captura de la transacción pero se descarta una vez completada. Para gestionar esta complejidad, el sistema utiliza Zustand como solución de estado global ligera,

creando stores específicos que funcionan como carritos de compras temporales donde se van agregando, modificando y eliminando ítems antes de confirmar la transacción final con el backend.

Modales

Los modales son componentes fundamentales que se renderizan fuera del flujo normal del DOM mediante portales de React, lo que permite mostrar contenido superpuesto sobre la interfaz principal sin afectar la jerarquía de componentes. El sistema utiliza dos tipos principales de modales: el Modal estándar y el ModalEntidad, ambos compartiendo una arquitectura similar pero con propósitos ligeramente diferentes.

El componente Modal base funciona mediante un portal que se monta en un elemento con id "portales", creando una estructura de tres capas: un overlay semitransparente que cubre toda la pantalla, el contenedor dialog que aloja el contenido, y un header fijo con el título y botón de cierre. Este componente es totalmente controlado mediante props, recibiendo el estado activo y su función modificadora setActivo, lo que permite al componente padre gestionar cuándo debe mostrarse u ocultarse. El overlay tiene una opacidad del 55% y un z-index de 20, mientras que el dialog tiene un z-index de 30 para garantizar que siempre esté sobre el overlay.

El ModalEntidad extiende esta funcionalidad base añadiendo la capacidad de gestionar un ID de entidad mediante setID, lo que resulta especialmente útil cuando se trabaja con detalles de registros específicos como productos o clientes. Este patrón se observa claramente en el componente Caja, donde el ModalEntidad se utiliza para mostrar los detalles de un producto seleccionado, pasando tanto el estado del modal como el ID del producto seleccionado, permitiendo así que el contenido del modal se actualice dinámicamente según la selección del usuario.