**Informe de Requerimientos del Sistema**

**Sistema de Gestión de Stock de Camu Camu**  
**Versión**: 1.0.0

**1. Introducción**

El presente documento describe los requerimientos funcionales y no funcionales del sistema de gestión de stock de Camu Camu. Este sistema tiene como finalidad modernizar y automatizar el registro de ingresos, pesajes, salidas, y la administración integral de productos, socios, clientes, personal y usuarios.

La solución ha sido desarrollada con una arquitectura moderna y robusta. El backend está construido con **Node.js**, **Express**, y utiliza **Sequelize** como ORM para interactuar con **MySQL**. La seguridad está garantizada mediante el uso de **JWT (JSON Web Tokens)** para autenticación. Además, se incorpora **SSE (Server-Sent Events)** para transmitir datos en tiempo real, especialmente útil en operaciones de pesaje. El frontend está desarrollado en **React** con la plantilla administrativa **CoreUI**, lo que garantiza una experiencia de usuario fluida, adaptable y profesional.

**2. Requerimientos Funcionales**

**2.1 Gestión de Usuarios**

* Autenticación segura con JWT.
* Administración de usuarios: creación, edición, eliminación y activación/desactivación.
* Asignación de roles con control de permisos.
* Restablecimiento de contraseña con valores predefinidos.
* Acceso controlado según perfil de usuario.

**2.2 Gestión del Personal**

* Registro del personal con sus datos completos: nombre, apellidos, DNI, teléfono, dirección, correo, cargo y área.
* Asociación de cada persona con un cargo funcional y un área estructural.
* Posibilidad de actualización, eliminación y control de estado.

**2.3 Gestión de Roles, Cargos y Áreas**

* Definición y mantenimiento de los roles del sistema con sus respectivas descripciones.
* Registro y gestión de cargos funcionales.
* Gestión de las áreas internas de la organización.

**2.4 Gestión de Socios y Clientes**

* Registro de socios con código, nombres, apellidos, dirección, caserío, teléfono, correo y estado.
* Registro de clientes con razón social, RUC, dirección, contacto y estado.
* Funciones de búsqueda, actualización y control de estado.

**2.5 Gestión de Productos**

* Registro y mantenimiento de productos, incluyendo nombre, tipo de fruta, unidad de medida y descripción.
* Administración de catálogos de tipos de fruta y unidades de medida.
* Habilitación o desactivación de productos para uso operativo.

**2.6 Gestión de Órdenes de Compra**

* Registro de órdenes de compra con cliente, código de lote, fechas, forma de pago, lugar de entrega y observaciones.
* Asociación de productos a la orden con cantidades, precios y subtotales.
* Seguimiento del estado de la orden (pendiente, en proceso, completado, cancelado).
* Control de cumplimiento por medio de cantidades ingresadas.

**2.7 Gestión de Ingresos**

* Registro de ingresos provenientes de socios, asociados a órdenes de compra.
* Captura de peso bruto, jabas, mermas y cálculo de peso neto.
* Cálculo automático de subtotales, impuestos, pagos al socio, y aportes a la cooperativa.
* Control de estado, observaciones y seguimiento histórico.

**2.8 Gestión de Pesajes**

* Registro de múltiples pesajes por ingreso, con control de peso de jaba y merma por cada evento.
* Cálculo automático del peso neto por pesaje.
* Registro de fecha y observaciones por pesaje.
* Transmisión de datos en tiempo real mediante **Server-Sent Events (SSE)**.

**2.9 Conectividad con Balanza**

* Detección de puertos disponibles para conectar balanza electrónica.
* Obtención de datos en tiempo real de forma continua o manual.
* Modo de simulación de datos para pruebas.
* Registro histórico de datos y observaciones.
* Asociación de datos de pesaje a ingresos registrados.

**2.10 Gestión de Salidas**

* Registro de salidas con guía de remisión, destino, observaciones y detalle asociado.
* Asociación de salidas a órdenes de compra específicas.
* Consultas por cliente, producto o fecha.
* Generación de reportes y dashboards con resumen de salidas.

**2.11 Reportes y Dashboard**

* Generación de reportes detallados sobre ingresos, salidas, productos, clientes y socios.
* Filtros avanzados por fecha, cliente, producto o estado.
* Exportación de reportes para análisis externo.

**3. Requerimientos No Funcionales**

**3.1 Seguridad**

* Uso de **JWT (JSON Web Tokens)** para autenticación de usuarios y autorización de recursos.
* Control de accesos por roles con permisos específicos.
* Protección de contraseñas mediante encriptación segura.

**3.2 Rendimiento**

* Las respuestas de API deben ejecutarse en menos de dos segundos bajo condiciones normales.
* Capacidad de manejar múltiples conexiones simultáneas sin degradar el rendimiento.
* Optimización de consultas SQL e índices en la base de datos.

**3.3 Escalabilidad**

* Arquitectura modular que permite la incorporación de nuevos módulos sin afectar los existentes.
* APIs estructuradas que facilitan la integración con otros sistemas.
* Uso de ORM (Sequelize) que facilita migraciones y mantenimiento.

**3.4 Usabilidad**

* Interfaz basada en **CoreUI React Admin**, moderna, responsiva y fácil de usar.
* Formularios interactivos con validación en tiempo real.
* Alertas, notificaciones y tablas dinámicas que mejoran la experiencia del usuario.

**3.5 Documentación Técnica**

* Documentación generada automáticamente con **Swagger UI** y especificaciones **OpenAPI 3.0**.
* Descripciones claras de endpoints, parámetros, esquemas y respuestas.
* Código backend documentado y organizado por módulos.

**3.6 Mantenibilidad**

* Estructura limpia del backend con separación de responsabilidades (rutas, controladores, servicios, modelos).
* Componentes reutilizables en frontend y pruebas de integridad básica del sistema.

**4. Conclusión**

Este informe define de manera precisa las funcionalidades que conforman el sistema de gestión de stock de Camu Camu, así como los estándares técnicos y de calidad implementados. El uso de tecnologías modernas como React, Express, Sequelize, JWT y SSE aseguran una solución segura, escalable y preparada para operar en entornos reales con exigencias operativas constantes. La documentación estructurada y las prácticas de desarrollo adoptadas permiten un mantenimiento ágil y futuras mejoras continuas.