DRS- Sistema de Gestión de compras e Inventario

Francisco Jareth Rosales Huey

Omar Tox Dzul

Ameyali

Alejandro Zamudio Cahuil

**1. Introducción**

**1.1 Propósito del documento**

Este documento tiene como finalidad definir de manera detallada y estructurada los **requisitos funcionales y no funcionales** del sistema de gestión de compras e inventario para tiendas locales. Servirá como base para el desarrollo técnico del sistema, validación de funcionalidades y futuras evaluaciones o ampliaciones del mismo.

**1.2 Público objetivo**

Este documento está dirigido a:

* El equipo de desarrollo que implementará el sistema.
* Profesores o revisores que evaluarán el cumplimiento de la práctica académica.
* Usuarios técnicos encargados de consumir o probar el sistema a través de clientes HTTP o sistemas web.

**1.3 Alcance del sistema**

El sistema tiene como objetivo permitir que los usuarios (propietarios o encargados de tiendas locales) puedan:

* Autenticarse y proteger sus operaciones mediante un token de acceso.
* Gestionar un inventario de productos (registro, edición, eliminación, consulta).
* Registrar compras a proveedores, afectando automáticamente existencias.
* Generar reportes de inventario y compras cruzando información de múltiples tablas.
* Recibir notificaciones por correo tras registrar una compra.
* Descargar reportes en formatos CSV o PDF.
* Consumir el API desde aplicaciones web y herramientas externas como Postman

**2. Descripción general del sistema**

**2.1 Contexto del sistema**

El sistema funcionará como una API RESTful desarrollada en PHP bajo el patrón MVC. Será consumida por clientes HTTP y un sistema web (SIW) construido en Vue.js o JavaScript puro (Por definir). Toda interacción estará protegida por una clave API individual generada al iniciar sesión.

**2.2 Funciones generales**

* Registro de usuarios y autenticación con clave API.
* Gestión de productos (CRUD).
* Gestión de proveedores.
* Registro de compras que afecta tres tablas (compras, detalle\_compra y productos).
* Generación de reportes descargables y notificaciones automáticas.

**2.3 Restricciones**

* El sistema será implementado con PHP, MySQL como base de datos y será probado localmente en entorno XAMPP, se plantea la posibilidad de subirla a un servidor web
* La autenticación se realizará mediante clave API enviada por header HTTP.
* El diseño de la base de datos será relacional y normalizado.

**2.4 Suposiciones**

* Se asume que cada usuario maneja su propio inventario y no comparte productos con otros usuarios.
* Se supone que el sistema será accedido por personas con conocimientos básicos en herramientas web o móviles.

3. Requisitos Funcionales (RF)

Esta sección enumera todas las funcionalidades que el sistema debe cumplir obligatoriamente para ser considerado funcionalmente completo. Se agrupan por módulo o entidad para mayor claridad.

Módulo: Usuarios y autenticación

ID Requisito Funcional

RF-01 El sistema debe permitir a los usuarios registrarse proporcionando nombre, correo y contraseña. **LISTO**

RF-02 El sistema debe generar una clave API única por usuario tras el registro. **LISTO**

RF-03 El sistema debe permitir a los usuarios iniciar sesión con correo y contraseña. **LISTO**

RF-04 El sistema debe devolver la clave API al momento de iniciar sesión exitosamente. **LISTO**

RF-05 El sistema debe validar la clave API en cada solicitud protegida por autenticación.

RF-06 El sistema debe restringir el acceso a los datos según el ID del usuario autenticado. **LISTO**

Módulo: Productos

ID Requisito Funcional

RF-07 El sistema debe permitir al usuario registrar nuevos productos indicando nombre, precio de compra, precio de venta y stock inicial.

RF-08 El sistema debe permitir al usuario consultar todos sus productos registrados.

RF-09 El sistema debe permitir editar productos existentes.

RF-10 El sistema debe permitir eliminar productos del inventario.

RF-11 El sistema debe permitir filtrar productos con bajo stock (menor a un umbral definido).

Módulo: Proveedores

ID Requisito Funcional

RF-12 El sistema debe permitir registrar nuevos proveedores indicando nombre, contacto y teléfono.

RF-13 El sistema debe permitir consultar todos los proveedores registrados por el usuario.

RF-14 El sistema debe permitir editar la información de un proveedor.

RF-15 El sistema debe permitir eliminar proveedores.

Módulo: Compras y Detalles

ID Requisito Funcional

RF-16 El sistema debe permitir registrar una compra seleccionando un proveedor y varios productos.

RF-17 El sistema debe calcular automáticamente el subtotal de cada producto y el total de la compra.

RF-18 El sistema debe registrar en la base de datos una fila en la tabla compras por cada compra realizada.

RF-19 El sistema debe registrar en la tabla detalle\_compra una fila por cada producto incluido en la compra.

RF-20 El sistema debe actualizar el stock de los productos involucrados en una compra sumando la cantidad comprada.

RF-21 El sistema debe permitir consultar todas las compras realizadas por el usuario autenticado.

RF-22 El sistema debe permitir ver el detalle completo de una compra, incluyendo proveedor, productos, precios, cantidades y totales.

Módulo: Notificaciones y Reportes

ID Requisito Funcional

RF-23 El sistema debe enviar un correo electrónico al usuario cuando se registre exitosamente una compra.

RF-24 El sistema debe permitir generar reportes de los 3 productos más comprados.

RF-25 El sistema debe permitir exportar reportes en formato CSV o PDF.

Módulo: Consumo desde interfaces externas

ID Requisito Funcional

RF-26 El sistema debe ser accesible y consumible desde herramientas de cliente HTTP como Postman, navegador o alguna otra herramienta

RF-27 El sistema debe poder ser consumido desde un sistema web externo desarrollado en Javascript o Vue.js (aún por definir).

**4. Requisitos No Funcionales (RNF)**

Los requisitos no funcionales definen criterios de calidad y condiciones que el sistema debe cumplir, más allá de sus funcionalidades. Estos requisitos aseguran que el sistema sea seguro, eficiente, mantenible y compatible con el entorno de ejecución.

**4.1 Rendimiento**

* RNF-01: El sistema debe ser capaz de procesar una solicitud de registro de compra que afecte múltiples tablas en menos de 2 segundos bajo condiciones normales de red local.
* RNF-02: La API debe ser capaz de manejar al menos 10 solicitudes simultáneas sin pérdida de datos ni errores de concurrencia.

**4.2 Usabilidad**

* RNF-03: La documentación de la API debe estar redactada con ejemplos claros de consumo para facilitar su integración desde cualquier cliente externo.
* RNF-04: El sistema web (cliente en Vue.js) debe contar con una interfaz clara, sin requerir conocimientos técnicos del usuario final para registrar productos o compras.

**4.3 Mantenibilidad**

* RNF-05: El sistema debe estar estructurado bajo el patrón Modelo-Vista-Controlador (MVC), separando claramente la lógica de negocio, presentación y acceso a datos.
* RNF-06: El código fuente debe estar debidamente comentado para facilitar futuras modificaciones o ampliaciones.

**4.4 Seguridad**

* RNF-07: Toda solicitud a la API que afecte recursos (POST, PUT, DELETE) debe requerir una clave API única por usuario enviada mediante encabezado HTTP.
* RNF-08: Las operaciones deben limitarse a los datos del usuario autenticado, garantizando la privacidad entre cuentas.
* RNF-09: Las contraseñas deben almacenarse en la base de datos utilizando algoritmos de hash seguro (por ejemplo, password\_hash).

**4.5 Fiabilidad**

* RNF-10: En caso de error de conexión a base de datos, el sistema debe retornar una respuesta JSON estructurada informando el tipo de error sin revelar detalles técnicos internos.
* RNF-11: Si ocurre un fallo durante el registro de una compra, el sistema debe realizar rollback y no dejar registros incompletos en las tablas.

**4.6 Compatibilidad**

* RNF-12: La API debe funcionar correctamente bajo entornos locales como XAMPP o WAMP utilizando PHP 7.4 o superior y MySQL.
* RNF-13: El cliente desarrollado en Vue.js debe ser compatible con los navegadores modernos (Chrome, Firefox, Edge).

**4.7 Portabilidad**

* RNF-14: El sistema debe poder ser desplegado en un servidor web estándar sin requerir configuraciones complejas fuera del entorno PHP y MySQL.

**5. Casos de Uso**

**5.1 Introducción a los Casos de Uso**

Los casos de uso describen las interacciones entre los actores (en este caso usuarios y sistemas externos) y el sistema para lograr un objetivo específico. Cada caso de aquí contiene:

* **Nombre**: claro y descriptivo.
* **Actor**: quién interactúa con el sistema.
* **Descripción**: breve resumen del propósito del caso de uso.
* **Precondiciones**: qué debe cumplirse o estar listo antes de iniciar el caso de uso.
* **Flujo principal**: pasos que siguen el actor y el sistema para completar la tarea de manera exitosa.
* **Flujos alternativos o excepciones**: qué sucede si hay errores o condiciones especiales.
* **Postcondiciones**: estado del sistema después de completar el caso de uso.

**5.2 Actores**

* **Usuario autenticado**: propietario o encargado de la tienda, que usa la API o la interfaz web para administrar productos, compras, proveedores y reportes.
* **Sistema de notificaciones**: módulo que envía correos al usuario cuando se registra una compra.
* **Sistema externo (cliente HTTP / Postman / sistema web externo Vue)**: consume la API para interactuar con el sistema.

**5.3 Casos de Uso Principales**

**Caso de Uso 1: Registro de usuario**

* **Actor:** Usuario no registrado
* **Descripción:** Permite al nuevo usuario crear una cuenta proporcionando nombre, correo y contraseña.
* **Precondiciones:** Ninguna (usuario nuevo).
* **Flujo principal:**
  1. El usuario envía datos de registro (nombre, correo, contraseña).
  2. El sistema valida que el correo no esté registrado.
  3. El sistema crea el registro del usuario con la contraseña hasheada.
  4. El sistema genera una clave API única.
  5. El sistema confirma el registro exitoso y devuelve la clave API.
* **Flujos alternativos:**
  1. 2a. Si el correo ya existe, se devuelve un mensaje de error.
* **Postcondiciones:** Usuario registrado y clave API asignada.

**Caso de Uso 2: Inicio de sesión**

* **Actor:** Usuario registrado
* **Descripción:** Permite al usuario autenticarse y obtener su clave API.
* **Precondiciones:** Usuario debe estar registrado.
* **Flujo principal:**
  1. El usuario envía correo y contraseña.
  2. El sistema valida las credenciales.
  3. Si son correctas, el sistema devuelve la clave API.
* **Flujos alternativos:**
  1. 2a. Si las credenciales son incorrectas, el sistema devuelve un mensaje de error.
* **Postcondiciones:** Usuario autenticado y en posesión de su clave API.

**Caso de Uso 3: Gestión de productos (registro, consulta, edición y eliminación)**

* **Actor:** Usuario autenticado
* **Descripción:** Permite al usuario administrar sus productos.
* **Precondiciones:** Usuario autenticado con clave API válida.
* **Flujo principal:**
  1. Registrar producto: usuario envía datos (nombre, precios, stock).
  2. Consultar productos: usuario solicita listado de sus productos.
  3. Editar producto: usuario envía modificaciones para un producto existente.
  4. Eliminar producto: usuario solicita eliminar un producto.
* **Flujos alternativos:**
  1. 4a. Si el producto no existe, se devuelve un error.
* **Postcondiciones:** Productos actualizados en la base de datos.

**Caso de Uso 4: Registro de compra**

* **Actor:** Usuario autenticado
* **Descripción:** Permite registrar una compra con uno o varios productos, actualizando inventario y generando notificaciones.
* **Precondiciones:** Usuario autenticado, productos y proveedor existentes.
* **Flujo principal:**
  1. Usuario selecciona proveedor y productos con cantidades.
  2. El sistema calcula subtotales y total.
  3. Se registra la compra en tabla compras.
  4. Se registran los detalles en detalle\_compra.
  5. Se actualiza el stock de productos.
  6. El sistema envía correo de notificación al usuario.
* **Flujos alternativos:**
  1. 4a. Si ocurre un error en la base de datos, se realiza rollback y se informa el error.
* **Postcondiciones:** Compra registrada, stock actualizado, notificación enviada.

**Caso de Uso 5: Generación y descarga de reportes**

* **Actor:** Usuario autenticado
* **Descripción:** Permite generar reportes personalizados y exportarlos en formatos CSV o PDF.
* **Precondiciones:** Usuario autenticado.
* **Flujo principal:**
  1. Usuario solicita reporte (por ejemplo, los 3 productos más comprados).
  2. Sistema genera el reporte cruzando datos necesarios.
  3. Usuario elige formato de exportación (CSV o PDF).
  4. Sistema entrega archivo descargable.
* **Flujos alternativos:**
  1. 3a. Si hay error en generación del reporte, se informa al usuario.
* **Postcondiciones:** Reporte generado y disponible para descarga.

**Caso de Uso 6: Consumo de API desde clientes externos**

* **Actor:** Sistema externo / Cliente HTTP / Aplicación web externa
* **Descripción:** Permite el acceso y consumo de la API para gestión remota.
* **Precondiciones:** Poseer clave API válida.
* **Flujo principal:**
  1. Cliente externo envía solicitudes HTTP con clave API en encabezados.
  2. API valida autenticidad y permisos.
  3. API responde con los datos solicitados o confirma operaciones realizadas.
* **Flujos alternativos:**
  1. 3a. Si la clave API es inválida o no está presente, la API devuelve error de autorización.
* **Postcondiciones:** Respuesta adecuada a la solicitud.

**6. Modelo Entidad – Relación**

**6.1. Tabla usuarios**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de dato | Clave | Restricciones y observaciones |
| idUsuario | INT AUTO\_INCREMENT | PK | Índice primario. |
| nombre | VARCHAR(100) |  | No nulo. |
| correo | VARCHAR(100) | ÚNICO | No nulo. Debe ser único. |
| contrasena | VARCHAR(255) |  | Almacenada de forma hasheada. |
| claveApi | VARCHAR(64) | ÚNICO | Token único por usuario. |

**6.2. Tabla productos**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de dato | Clave | Restricciones y observaciones |
| idProducto | INT AUTO\_INCREMENT | PK | Índice primario. |
| idUsuario | INT | FK, INDEX | Foránea hacia usuarios(idUsuario).ON DELETE CASCADE, ON UPDATE CASCADE. |
| nombre | VARCHAR(100) |  | No nulo. |
| precioCompra | DECIMAL(10,2) |  | No nulo. |
| precioVenta | DECIMAL(10,2) |  | No nulo. |
| stock | INT |  | No nulo. Valor inicial al registrar el producto. |

**6.3. Tabla proveedores**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de dato | Clave | Restricciones y observaciones |
| idProveedor | INT AUTO\_INCREMENT | PK | Índice primario. |
| idUsuario | INT | FK, INDEX | Foránea hacia usuarios(idUsuario).  ON DELETE CASCADE, ON UPDATE CASCADE. |
| nombre | VARCHAR(100) |  | No nulo. |
| contacto | VARCHAR(100) |  | Opcional. Nombre de contacto. |
| telefono | VARCHAR(20) |  | Opcional. |

**6.4. Tabla compras**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de dato | Clave | Restricciones y observaciones |
| idCompra | INT AUTO\_INCREMENT | PK | Índice primario. |
| idUsuario | INT | FK, INDEX | Foránea hacia usuarios(idUsuario).ON DELETE CASCADE, ON UPDATE CASCADE. |
| idProveedor | INT | FK, INDEX | Foránea hacia proveedores(idProveedor).ON DELETE SET NULL, ON UPDATE CASCADE. |
| fecha | DATETIME |  | Asignada automáticamente al crear la compra. |
| total | DECIMAL(10,2) |  | Se calcula sumando subtotales de los productos comprados. |

**6.5 Tabla detalle\_compra**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Tipo de dato | Clave | Restricciones y observaciones |
| idDetalle | INT AUTO\_INCREMENT | PK | Índice primario. |
| idCompra | INT | FK, INDEX | Foránea hacia compras(idCompra).ON DELETE CASCADE, ON UPDATE CASCADE. |
| idProducto | INT | FK, INDEX | Foránea hacia productos(idProducto).ON DELETE RESTRICT, ON UPDATE CASCADE. |
| cantidad | INT |  | No nulo. Debe ser mayor que cero. |
| precioUnitario | DECIMAL(10,2) |  | Precio del producto al momento de la compra. |
| subtotal | DECIMAL(10,2) |  | Calculado: precioUnitario \* cantidad. |

**6.6.** **Justificación de reglas de integridad**

* **ON DELETE CASCADE (usuarios)**: al eliminar un usuario, se eliminan automáticamente sus productos, proveedores, compras y sus detalles. Esto evita dejar huérfanos los registros dependientes.
* **ON DELETE SET NULL (proveedores en compras)**: si se elimina un proveedor, se conserva la compra como histórico, pero se desvincula el proveedor.
* **ON DELETE RESTRICT (productos en detalle\_compra)**: no se puede eliminar un producto que haya sido parte de una compra registrada. Esto protege la integridad histórica de los movimientos.
* Flujo de construcción:
  + Usuarios
  + Productos
  + Proveedores
  + Compras
  + Detallecompora