SISTEMA DE GESTIÓN DE INVENTARIO

Documentación Técnica del Proyecto

1. Descripción General

El **Sistema de Gestión de Inventario** es una aplicación web orientada a la administración y manipulación de información mediante las operaciones básicas **CRUD (Crear, Leer, Actualizar y Eliminar)**. El sistema permite gestionar de manera estructurada datos relacionados con activos, productos, personas, marcas y movimientos, facilitando un control eficiente del inventario.

La aplicación ha sido desarrollada utilizando tecnologías web estándar, garantizando portabilidad, facilidad de mantenimiento y compatibilidad con navegadores modernos. La comunicación entre el cliente y los datos se realiza mediante una **API REST simulada**, lo que permite una clara separación entre la lógica de presentación y la gestión de información.

2. Arquitectura del Sistema

El sistema se basa en una **arquitectura Cliente–Servidor**, organizada en capas que permiten una mejor escalabilidad, mantenimiento y comprensión del software.

2.1 Capa Cliente (Frontend)

La capa cliente se ejecuta en el navegador web y es responsable de la presentación visual, la interacción con el usuario y las validaciones básicas. Está desarrollada con las siguientes tecnologías:

- **HTML5**, para la estructura semántica de la aplicación.
- **CSS3**, para el diseño visual y la adaptabilidad a distintos dispositivos.
- **JavaScript (ES6+)**, para la lógica de interacción, manipulación del DOM y consumo de servicios REST.

2.2 Capa de Lógica de Aplicación

La lógica del sistema se encuentra modularizada en archivos JavaScript, lo que favorece la separación de responsabilidades y la reutilización del código:

- **api.js:** Gestiona la comunicación HTTP con la API REST, encapsulando las operaciones GET, POST, PUT y DELETE.
- **crud.js:** Controla la generación dinámica de formularios y la ejecución de las operaciones CRUD.
- **menu.js:** Construye dinámicamente el menú lateral de navegación y gestiona las acciones del usuario.
- **dataForm.js:** Define las estructuras de datos necesarias para la creación de formularios dinámicos.

2.3 Capa Servidor (Backend Simulado)

El backend se encuentra prototipado mediante **json-server**, el cual simula una API RESTful, permitiendo realizar operaciones CRUD sin necesidad de un servidor real.

2.4 Persistencia de Datos

Los datos del sistema se almacenan en un archivo local:

- **storage/data/db.json**

Este archivo actúa como base de datos simulada durante el desarrollo y las pruebas del sistema.

3. Requisitos para el Funcionamiento

Para la correcta ejecución del sistema se requiere:

- Contar con **Node.js** instalado.
- Instalar **json-server**, preferiblemente la versión **0.17.4**, debido a su mayor estabilidad.
- Ejecutar el siguiente comando para iniciar el servicio REST:

```
json-server --watch storage/data/db.json
```

4. Tecnologías Utilizadas

- HTML5
- CSS3 (CSS Vanilla)
- JavaScript (ES6+)
- json-server

5. Diseño de la Interfaz

El diseño de la interfaz sigue criterios de orden, consistencia y mantenibilidad:

- Aplicación de la **metodología BEM** para la organización de estilos.
- Diseño **responsivo**, con enfoque **Desktop First**.
- Uso exclusivo de **CSS Vanilla**, sin frameworks externos.
- Estructura modular de estilos, donde el archivo main.css centraliza la importación de los demás módulos.

6. Servicios de la API

El archivo **api.js** define los servicios que permiten la interacción con la API REST simulada:

- **getData(endPoint):** Obtiene datos desde un endpoint específico.
- **postData(element, endPoint):** Envía nuevos registros al servidor.
- **getDataId(endPoint, id):** Recupera un recurso específico mediante su identificador.
- **deleteData(endPoint, id):** Elimina un recurso existente.
- **updateData(endPoint, id, object):** Actualiza la información de un recurso.

Todos los servicios retornan datos en formato JSON.

7. Descripción de Módulos

7.1 menu.js

Genera dinámicamente el menú lateral del sistema, permitiendo al usuario acceder a las opciones de creación, edición, eliminación y búsqueda de información. Este módulo se encarga de la navegación principal de la aplicación.

7.2 dataForm.js

Define las estructuras de los formularios utilizados en el sistema. Cada formulario contiene información sobre campos, etiquetas y tipos de entrada, permitiendo su generación dinámica en la interfaz.

7.3 crud.js

Gestiona la lógica central del sistema, integrando los formularios con los servicios de la API. Controla las acciones del usuario, el envío de datos, la visualización de resultados y la ejecución de las operaciones CRUD.

8. Backend

- Los datos se almacenan en el archivo storage/data/db.json.
- El backend es simulado mediante **json-server**.
- Se recomienda el uso de la versión **0.17.4** por su estabilidad comprobada.

9. Autoría

- **Autores:**
    - Sofía Marcela Medina Díaz
    - Liliana Paola Castellanos Pinzón
- **Institución:** CampusLands
- **Revisión técnica:** Training Leader

10. Auditoría del Sistema

La auditoría del sistema se realizó con el objetivo de **verificar el cumplimiento de los requisitos funcionales y no funcionales**, así como evaluar la calidad del software en términos de funcionamiento, seguridad, usabilidad y mantenibilidad. Este proceso permitió identificar oportunidades de mejora y asegurar que el sistema opere de acuerdo con los estándares establecidos.

La auditoría se centró en la revisión de los siguientes aspectos:

- Correcta ejecución de las operaciones CRUD.
- Integridad y consistencia de los datos almacenados en el sistema.
- Adecuado control de accesos mediante el formulario de autenticación.
- Organización del código fuente y aplicación de buenas prácticas de programación.
- Correcta interacción entre el frontend y la API REST simulada.

Como resultado del proceso de auditoría, se validó que el sistema cumple con los criterios definidos para su operación en un entorno controlado, recomendándose mejoras orientadas a la optimización del rendimiento y fortalecimiento de la seguridad en futuras versiones.

11. Requisitos del Sistema

11.1 Requisitos Funcionales

El sistema debe cumplir con los siguientes requisitos funcionales:

- RF-01: El sistema debe permitir el **registro de productos y activos** mediante formularios dinámicos.
- RF-02: El sistema debe permitir la **visualización de productos** para los usuarios en modo Cliente.
- RF-03: El sistema debe permitir **editar la información** de productos y activos existentes.
- RF-04: El sistema debe permitir **eliminar registros** del inventario previa confirmación.
- RF-05: El sistema debe permitir la **búsqueda de productos** mediante criterios definidos.
- RF-06: El sistema debe permitir la **gestión de stock**, controlando entradas y salidas de productos.
- RF-07: El sistema debe contar con un **formulario de autenticación** para el acceso al sistema.
- RF-08: El sistema debe diferenciar los **roles de Cliente y Administrador**, limitando el acceso a funciones administrativas.

- RF-09: El sistema debe comunicarse con la API REST simulada para realizar operaciones CRUD.

11.2 Requisitos No Funcionales

El sistema debe cumplir con los siguientes requisitos no funcionales:

- RNF-01 (**Usabilidad**): La interfaz debe ser intuitiva y fácil de usar, permitiendo al usuario realizar sus tareas sin dificultad.
- RNF-02 (**Rendimiento**): El sistema debe responder a las solicitudes del usuario en un tiempo adecuado, incluso con múltiples registros.
- RNF-03 (**Seguridad**): El acceso al sistema debe estar protegido mediante autenticación de usuarios.
- RNF-04 (**Mantenibilidad**): El código debe estar organizado y documentado, aplicando buenas prácticas de programación.
- RNF-05 (**Escalabilidad**): El sistema debe permitir la incorporación de nuevas funcionalidades sin afectar las existentes.
- RNF-06 (**Portabilidad**): El sistema debe ser compatible con navegadores web modernos.
- RNF-07 (**Disponibilidad**): El sistema debe estar disponible durante el horario de operación definido.

DICCIONARIO DE DATOS DEL SISTEMA

1. Tabla: Products (Productos / Activos)

Campo	Tipo de Dato	Descripción	Observaciones
id	Entero	Identificador único del producto	Clave primaria
code	Cadena	Código interno del producto	Único
name	Cadena	Nombre del producto o activo	—
serialNumber	Cadena	Número de serie del producto	Identificación física
price / unitaryPrice	Decimal	Precio unitario del producto	Moneda local
productProvider / activeProvider	Entero	Proveedor asociado	FK → Providers

productCategory / activeCategory	Entero	Categoría del producto	FK → Categories
productBrand / activeBrand	Entero	Marca del producto	FK → Brands
activeType	Entero	Tipo de activo	FK → TypesActive
activeResponsible	Entero	Responsable del activo	FK → Responsibles
activeStatus	Entero	Estado del activo	FK → States
stock	Entero	Cantidad disponible en inventario	≥ 0

2. Tabla: TypesActive (Tipos de Activo)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del tipo de activo
name	Cadena	Nombre del tipo de activo (CPU, Monitor, etc.)

3. Tabla: Categories (Categorías)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador de la categoría
name	Cadena	Nombre de la categoría

4. Tabla: States (Estados del Activo)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del estado
name	Cadena	Estado del activo (Asignado, No Asignado, De Baja)

5. Tabla: Brands (Marcas)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador de la marca
name	Cadena	Nombre de la marca

6. Tabla: Providers (Proveedores)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del proveedor
name	Cadena	Nombre del proveedor

7. Tabla: Responsibles (Responsables)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del responsable
name	Cadena	Nombre del responsable del activo

8. Tabla: Users (Usuarios del Sistema)

Campo	Tipo de Dato	Descripción	Observaciones
id	Entero	Identificador del usuario	Clave primaria
username	Cadena	Nombre de usuario	Único
password	Cadena	Contraseña del usuario	Encriptada (recomendado)
role	Cadena	Rol del usuario (admin, buyer)	Control de acceso
name	Cadena	Nombre completo del usuario	—

9. Tabla: Sales (Ventas)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador de la venta
date	Fecha	Fecha de la venta
total	Decimal	Valor total de la venta
customer	Cadena	Tipo de cliente (Invitado / Registrado)
items	Arreglo	Detalle de productos vendidos

10. Tabla: Sales Items (Detalle de Venta)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del producto vendido
code	Cadena	Código del producto
name	Cadena	Nombre del producto

serialNumber	Cadena	Número de serie
quantity	Entero	Cantidad vendida
unitaryPrice / price	Decimal	Precio unitario

11. Tabla: TypesCustomer (Tipos de Cliente)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del tipo de cliente
name	Cadena	Tipo de cliente (Natural, Jurídica)

12. Tabla: TypesMovActive (Tipos de Movimiento de Activo)

Campo	Tipo de Dato	Descripción
id	Entero	Identificador del movimiento
name	Cadena	Nombre del tipo de movimiento