2025

INVENTARIO

SISTEMA DE INVENTARIO PARA UNA TIENDA

ARIADNA DENISSE GARCIA ESTRADA

https://github.com/ariadna-ge/Inventario_tienda.git

CONTENIDO

SISTEMA DE INVENTARIO PARA TIENDA

1. Introducción

Este documento describe la implementación y estructura del sistema de inventario para una tienda, desarrollado como proyecto de programación utilizando tecnologías basadas en JavaScript. El sistema permite gestionar productos, registrar entradas y salidas, y visualizar estadísticas relevantes mediante un panel principal (dashboard).

2. Objetivo del Proyecto

El objetivo principal del proyecto es ofrecer una solución funcional para la gestión de inventario de una tienda, permitiendo el registro de movimientos, visualización de datos analíticos, y organización estructurada de productos.

3. Tecnologías Utilizadas

- Node.js
- Express.js
- JavaScript (JS)
- CSS
- MySQL

4. Estructura del Proyecto

El proyecto está organizado en los siguientes directorios principales:

- ***config:*** Configuración de la base de datos y parámetros generales.
- ***controllers:*** Lógica de control para manejo de rutas y solicitudes.

- ***models:*** Definición de modelos de datos.
- ***public:*** Archivos estáticos como CSS, JS y recursos multimedia.
- ***routes:*** Definición de rutas del servidor.
- ***services:*** Servicios auxiliares o lógicos complementarios.
- ***views:*** Vistas renderizadas en el cliente.

Controllers

Archivo: indexController.js

Función principal: mostrarDashboard

Descripción general: Este módulo se encarga de obtener los datos necesarios para mostrar el panel de control (dashboard) de la aplicación de inventario. Utiliza el modelo salidasModel para acceder a la base de datos y recuperar estadísticas clave.

Proceso

1. Obtiene la fecha y el día actual en formato local.
2. Llama a diferentes métodos del modelo salidasModel para:
    - Obtener el total de ventas del mes.
    - Calcular el número total de productos.
    - Obtener la cantidad de categorías registradas.
    - Contar el total de SKU.
    - Detectar productos con stock bajo.
    - Listar los productos más vendidos.
3. Utiliza Promise.all para ejecutar las consultas de forma paralela y más eficiente.
4. Renderiza la vista index con todos los datos recopilados.

5. Maneja errores mostrando una vista error con un mensaje amigable.

Archivo: productosController.js

Función principal: mostrarDashboard

Descripción general: Este módulo define controladores genéricos para operaciones **CRUD** sobre tablas de la base de datos relacionadas con productos, utilizando el servicio productosService para interactuar con la capa de datos.

Dependencias

- productosService: Contiene las funciones para consultar e interactuar con la base de datos.

Funciones exportadas

1. getAll(tableName, viewName)

Descripción: Genera un controlador que obtiene todos los registros de una tabla y renderiza una vista con esos datos.

Parámetros:

Nombre	Tipo	Descripción
tableName	String	Nombre de la tabla a consultar.
viewName	String	Nombre de la vista a renderizar.

Flujo:

1. Llama a productosService.getAllData(tableName).
2. Maneja errores devolviendo un 500.
3. Si es exitoso, renderiza la vista especificada con los resultados.

2. post(tableName, redirectPath)

Descripción: Genera un controlador que inserta un nuevo registro en la tabla indicada.

Parámetros:

Nombre	Tipo	Descripción
tableName	String	Nombre de la tabla donde insertar.
redirectPath	String	Ruta a la que redirigir tras la inserción.

Flujo:

1. Recibe req.body como datos del formulario.
2. Valida que los datos sean un objeto válido.
3. Llama a productosService.postData().
4. Redirige a redirectPath si es exitoso.

3. getById(tableName, viewName, idName = 'id')

Descripción: Obtiene un registro específico por su ID y lo pasa a una vista.

Parámetros:

Nombre	Tipo	Descripción
tableName	String	Nombre de la tabla.
viewName	String	Vista a renderizar.
idName	String	Nombre de la columna que actúa como identificador.

Flujo:

1. Extrae id desde req.params.
2. Llama a productosService.getDataById().
3. Si el resultado existe, renderiza la vista con el registro.

4. Si no existe, devuelve 404.

4. updateById(tableName, redirectPath, idName = 'id')

Descripción: Actualiza un registro por su identificador.

Parámetros:

Nombre	Tipo	Descripción
tableName	String	Nombre de la tabla.
redirectPath	String	Ruta a redirigir tras la actualización.
idName	String	Nombre de la columna identificadora.

Flujo:

1. Extrae id desde req.params.
2. Valida req.body.
3. Llama a productosService.updateDataById().
4. Redirige si es exitoso, o devuelve 500 en caso de error.

Routes

Descripción general: Este módulo define las rutas principales de la aplicación Express, conectando endpoints HTTP con los controladores que gestionan inventario, entradas, salidas y el dashboard.

Dependencias

- express: Framework para crear el servidor y manejar rutas.
- productosController: Controlador con funciones genéricas CRUD para diferentes tablas.
- indexController: Controlador que maneja la carga de datos para el dashboard.

Listado de rutas

Inventario

Método	Ruta	Descripción
GET	/inventario	Muestra todos los productos del inventario.
POST	/agregar	Inserta un nuevo producto.
GET	/productos/editar/:id_producto	Muestra el formulario para editar un producto específico.
POST	/agregar/:id_producto	Actualiza un producto existente.

Entradas

Método	Ruta	Descripción
GET	/entradas	Muestra todas las entradas de inventario.
POST	/agregarEntrada	Registra una nueva entrada de producto.

Salidas

Método	Ruta	Descripción
GET	/salidas	Muestra todas las salidas de inventario.
POST	/agregarSalida	Registra una nueva salida de producto.

Rutas de vistas estáticas

Método	Ruta	Descripción
GET	/agregar	Muestra formulario para agregar producto.
GET	/agregarSalida	Muestra formulario para agregar salida.
GET	/agregarEntrada	Muestra formulario para agregar entrada.

Dashboard

Método	Ruta	Descripción
GET	/	Muestra el panel principal con métricas de ventas, inventario y productos más vendidos.

5. Descripción de Funcionalidades

El sistema ofrece las siguientes funcionalidades principales:

- Registro de productos disponibles.

- Registro de entradas y salidas de inventario.

- Visualización de productos con bajo stock.

- Visualización de productos con mayores ventas.

- Visualización del número total de categorías y productos.

- Cálculo de ventas correspondientes al mes.

- Panel principal (dashboard) con acceso a todas las funciones.

6. Configuración e Instalación

Para ejecutar el sistema localmente, siga los siguientes pasos:

1. Instale Node.js y npm si no están instalados.

2. Instale las dependencias del proyecto ejecutando `npm install` en el directorio raíz.

3. Configure el archivo `config/bd.js` con los datos correctos de su base de datos local.

4. Ejecute el servidor con el comando `node server.js`.

5. Acceda al sistema desde su navegador en `http://localhost:puerto`.

7. Licencia

Este proyecto está licenciado bajo la Licencia MIT.