Manual Técnico: Sistema Web de Inventario para Tienda

1. Introducción

Propósito del Manual

Este manual técnico está diseñado para proporcionar una guía detallada sobre la instalación, configuración, arquitectura, mantenimiento y seguridad de una aplicación web de inventario para una tienda. Está dirigido a desarrolladores, administradores de sistemas y personal técnico que necesiten implementar o dar soporte a la aplicación.

Alcance del Software

La aplicación permite la gestión de productos, usuarios, citas, solicitudes PQRs y el manejo de un carrito de compras. Incluye los siguientes módulos:

- Registro y login de usuarios con permisos de administrador.
- Registro y visualización de productos con CRUD completo (GET, POST, PUT, DELETE).
- Gestión de PQRS (Petición, Queja, Reclamo, Sugerencia).
- Registro de citas.
- Carrito de compras con catálogo de productos e imágenes.
- Envío automático de factura al correo del cliente tras realizar una compra.
- Implementación de API RESTful para permitir la comunicación entre el frontend y el backend, facilitando operaciones como crear, consultar, actualizar y eliminar recursos (productos, usuarios, citas, etc.).

Versiones del Documento

- Versión 1.0 - Junio 2025 - Primera publicación.

2. Requisitos del Sistema

Hardware Necesario

- Procesador: Intel i5 o superior
- Memoria RAM: 8 GB o superior
- Almacenamiento: 500 MB libres (base de datos y archivos estáticos)

Software y Dependencias

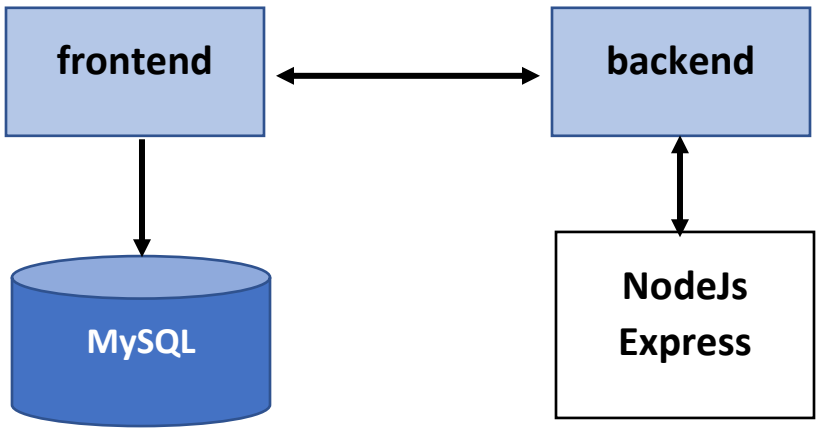
- Node.js v18 o superior
- MySQL Server v8+
- NPM
- Navegador actualizado (Chrome, Firefox, Edge)

Configuraciones Recomendadas

- Sistema operativo: Windows 10/11, Ubuntu 20.04+
- Editor de código: Visual Studio Code
- Puertos: 3000 (frontend/backend), 3306 (MySQL)

3. Arquitectura del Software

Diseño General y Componentes



- **Frontend**: HTML, CSS, JavaScript.
- **Backend**: Node.js con Express.
- **Base de datos**: MySQL.
- **Correo**: Nodemailer para envío de facturas.

Diagramas de Arquitectura (Imagen sugerida: Diagrama tipo MVC con rutas, controladores, modelos y vistas)

Cliente
  |
Frontend (HTML, CSS, JS)
  |
Backend (Express.js)
  |
Base de Datos (MySQL)

Relaciones entre Módulos

- El módulo de usuarios controla accesos.
- Productos se gestionan mediante usuarios administradores.
- PQRs pueden ser enviadas por cualquier usuario registrado.
- Citas solo pueden ser agendadas por usuarios logueados.
- Carrito y facturación interactúan con los productos y usuarios.

API RESTful
La aplicación expone una API RESTful diseñada bajo los principios de arquitectura REST, permitiendo a clientes (como el frontend web o futuras apps móviles) interactuar con los datos del sistema.
Cada recurso (usuarios, productos, citas, pqrs, etc.) cuenta con rutas estructuradas que permiten realizar operaciones CRUD de forma clara y organizada.

Ejemplos de Endpoints

- GET /api/productos → Obtener lista de productos
- POST /api/productos → Crear nuevo producto
- PUT /api/productos/:id → Actualizar producto existente
- DELETE /api/productos/:id → Eliminar producto
- POST /api/login → Autenticación del usuario
- GET /api/usuarios/:id → Obtener datos del usuario
- GET /api/pqrs → Consultar peticiones recibidas
- POST /api/citas → Registrar nueva cita

Comunicación Cliente-Servidor

El frontend consume estos endpoints mediante fetch() en JavaScript para cargar datos dinámicamente en la página sin necesidad de recargarla, proporcionando una experiencia fluida tipo SPA (Single Page Application).

Seguridad en la API

- Validación de datos en cada endpoint
- Middleware para restringir acceso según roles (admin, usuario)
- En futuras versiones: autenticación por token JWT

4. Instalación y Configuración

Paso a Paso para la Instalación

1. Clonar el repositorio:

```
git clone https://github.com/usuario/proyecto-inventario.git
```

2. Instalar dependencias:

```
cd backend
npm init -y
npm install expres, mysql2, cors, bcrypt, nodemon, nodemailer
```

3. Configurar base de datos en config/db.js

4. Ejecutar scripts SQL en MySQL Workbench:

```
CREATE DATABASE inventario_tienda;
-- Importar tablas desde /sql/schema.sql
```

5. Iniciar servidor:

```
node server.js
npm run dev
```

6. Abrir en navegador: http://localhost:3000

Configuración Inicial

- Crear primer usuario administrador desde register.html.
- Ingresar al sistema con credenciales.

Configuraciones Avanzadas

- Configurar servicio de correo con credenciales seguras.

5. Mantenimiento y Actualización

Procedimientos de Mantenimiento

- Respaldar la base de datos semanalmente.
- Revisar logs en logs/server.log.

¿Qué son los logs?

Los logs (o registros) son mensajes que el sistema guarda automáticamente para indicar lo que está ocurriendo durante su ejecución. Pueden contener información como:

- Inicio o detención del servidor.
- Errores de conexión o procesos fallidos.
- Actividades del usuario (login, compras, etc.).

¿Dónde se encuentran?

En este sistema, los logs están guardados en el archivo logs/server.log. Puedes abrirlo con un editor de texto para revisar lo que ha sucedido.

Ejemplo de log:
[2025-06-17 12:15:02] INFO: Usuario admin inició sesión correctamente.
[2025-06-17 12:16:10] ERROR: No se pudo conectar a la base de datos.

¿Cómo se generan?

Con Node.js se pueden generar con console.log() o librerías como winston. Ejemplo:

fs.appendFileSync('logs/server.log', '[INFO] Servidor iniciado correctamente\n');

Actualizaciones y Parches

- Realizar pull del repositorio y reiniciar servidor.

quiere decir:
Descarga los últimos cambios del proyecto desde GitHub.

Aplica esos cambios reiniciando tu servidor local para que funcionen correctamente.

Solución de Problemas Comunes

- **Error de conexión a BD**: Verifica credenciales en database.js.
- **Servidor no inicia**: Verifica dependencias o conflictos de puertos.
- **Correo no enviado**: Revisar configuración SMTP.

6. Seguridad y Protección

Medidas de Seguridad

- Contraseñas cifradas con bcrypt.
- Validación de datos en backend y frontend.

Configuraciones de Seguridad Recomendadas

- Uso de HTTPS.

¿Qué significa “Uso de HTTPS”?

HTTPS (Hypertext Transfer Protocol Secure) es la **versión segura de HTTP**, el protocolo que usan los navegadores para comunicarse con los sitios web.

Cuando ves una página que empieza con https:// (por ejemplo, https://www.google.com), significa que:

- La información que se envía entre el navegador del usuario y el servidor **está cifrada**.
- Nadie (ni hackers, ni atacantes en redes WiFi públicas) puede leer fácilmente los datos que se envían o reciben.

¿Qué protege HTTPS?

- **Datos sensibles**, como contraseñas, correos, nombres de usuario, datos del carrito de compras, etc.
- **La identidad del sitio web** (evita que alguien haga pasar su sitio por el tuyo).
- **La integridad de la información** (evita que alguien altere los datos mientras viajan por la red).

- Autenticación por token JWT (próxima versión).

¿Qué es JWT?

JWT significa **JSON Web Token**.
Es un **método moderno y seguro** para autenticar usuarios en una aplicación web.

En lugar de guardar la sesión en el servidor (como con session y cookie), con JWT se entrega un **token firmado** al usuario cuando se loguea, y ese token se envía en cada solicitud para demostrar que el usuario está autenticado.

¿Cómo funciona JWT paso a paso?

1. **Inicio de sesión**:
   - El usuario ingresa su correo y contraseña.
   - El servidor verifica las credenciales.
   - Si son correctas, el servidor **genera un token JWT** y lo envía al cliente (navegador).
2. **Almacenar el token**:
   - El navegador guarda el token en localStorage o sessionStorage.
3. **Usar el token**:
   - Cada vez que el usuario realiza una petición (por ejemplo, ver productos, añadir al carrito), el token se envía en la cabecera HTTP:

     Authorization: Bearer <token>

4. **El servidor valida el token**:
   - Si es válido, permite el acceso a la ruta protegida.
   - Si no, responde con error 401 (no autorizado).

- Roles de acceso: admin vs usuario.

Ejemplo de cuadro comparativo

Acción / Permiso	Administrador ☑	Usuario Normal 👤
Crear productos	☑	❌
Editar productos	☑	❌
Eliminar productos	☑	❌
Ver catálogo de productos	☑	☑
Usar carrito de compras	☑	☑
Enviar PQRs	☑	☑
Ver y gestionar PQRs	☑	❌
Registrar citas	☑	☑
Ver todas las citas	☑	❌ (solo propias)
Crear/gestionar usuarios	☑	❌
Ver facturas enviadas	☑	☑ (solo propias)

Gestión de Riesgos

- Monitoreo de actividad sospechosa.

Este apartado se refiere a mecanismos para detectar comportamientos anómalos que podrían indicar un intento de intrusión, abuso del sistema o uso indebido por parte de usuarios.

Contenido sugerido:

- Registro de **intentos fallidos de inicio de sesión**.
- Alerta en consola/logs si un usuario intenta acceder a rutas no autorizadas.
- Seguimiento de **cambios masivos** en productos o citas (por ejemplo, más de 10 eliminaciones seguidas).
- Registro de IPs desde las cuales se realiza el acceso.
- Posibilidad de bloquear cuentas tras múltiples intentos fallidos (en futuras versiones).
- Revisión manual periódica de los logs por parte del administrador del sistema.

Ejemplo de código básico para monitoreo:

```
// Middleware para detectar intentos a rutas protegidas sin autorización
app.use((req, res, next) => {
  if (!req.user && req.originalUrl.includes('/admin')) {
    console.warn(`Intento de acceso no autorizado: ${req.originalUrl}`);
  }
  next();
});
```

- Registros de auditoría básicos (logins, operaciones CRUD).

Los registros de auditoría permiten dejar trazabilidad de las acciones importantes dentro del sistema, especialmente aquellas relacionadas con **creación, modificación o eliminación de datos**.

Contenido sugerido:

- Registro de cada **inicio de sesión exitoso o fallido** con usuario, fecha y hora.
- Registro de operaciones **CRUD** en productos, usuarios, citas y PQRS:
    - Quién hizo la acción (usuario logueado)
    - Qué acción hizo (crear, editar, eliminar)
    - Cuándo la hizo
    - A qué recurso afectó (ej. producto ID 8 eliminado)

Ejemplo de formato de log:

[2025-06-17 08:22:15] Usuario admin@example.com inició sesión.
[2025-06-17 08:25:02] Usuario admin@example.com creó producto ID: 12.
[2025-06-17 08:30:40] Usuario usuario1@example.com eliminó cita ID: 33.

Cómo implementarlo:

- Puedes usar console.log() en producción (si se redirecciona a un archivo con morgan o winston).
- O guardar en una tabla logs en MySQL con columnas: usuario_id, accion, fecha_hora, descripcion.