

Roles y Seguridad - Sistema Tiendario

Sistema de Seguridad Implementado

Tecnologías de Seguridad

- Spring Security 6.x
- JWT (JSON Web Tokens)
- BCrypt para encriptación de contraseñas
- RBAC (Role-Based Access Control)

Roles del Sistema

1. ADMINISTRADOR

Permisos: Acceso completo al sistema

- ☒ Gestión completa de usuarios (CRUD)
- ☒ Gestión completa de roles (CRUD)
- ☒ Gestión completa de productos (CRUD)
- ☒ Gestión completa de categorías (CRUD)
- ☒ Gestión completa de proveedores (CRUD)
- ☒ Gestión completa de movimientos de stock (CRUD)
- ☒ Acceso a todos los reportes y consultas

Endpoints exclusivos:

- GET/POST/PUT/DELETE /api/usuarios/**
- GET/POST/PUT/DELETE /api/roles/**

2. ENCARGADO_INVENTARIO

Permisos: Gestión del inventario y productos

- ☒ Gestión completa de productos (CRUD)
- ☒ Gestión completa de categorías (CRUD)
- ☒ Gestión completa de proveedores (CRUD)
- ☒ Gestión completa de movimientos de stock (CRUD)
- ☐ Sin acceso a gestión de usuarios
- ☐ Sin acceso a gestión de roles

Endpoints compartidos con ADMINISTRADOR:

- POST/PUT/DELETE /api/productos/**
- POST/PUT/DELETE /api/categorias/**
- GET/POST/PUT/DELETE /api/proveedores/**
- GET/POST/PUT/DELETE /api/movimientos/**

3. VENDEDOR

Permisos: Solo consulta de productos y categorías






- ☒ Consulta de productos (solo GET)
- ☒ Consulta de categorías (solo GET)
- ☐ Sin permisos de modificación
- ☐ Sin acceso a gestión de usuarios/roles
- ☐ Sin acceso a proveedores o movimientos

Endpoints de solo lectura:






- GET /api/productos/**
- GET /api/categorias/**

Flujo de Autenticación

Registro de Usuario

1. **Endpoint:** POST /register
2. **Proceso:**
 -  Usuario envía datos (nombre, correo, contraseña)
 -  Sistema valida unicidad de nombre y correo
 -  Contraseña se encripta con BCrypt
 -  Se asigna rol VENDEDOR por defecto
 -  Usuario queda registrado

Inicio de Sesión

1. **Endpoint:** POST /login
2. **Proceso:**
 -  Usuario envía credenciales (nombre, contraseña)
 -  Spring Security valida credenciales
 -  Si es válido, se genera token JWT
 -  Token incluye: username, rol, tiempo de expiración (1 hora)
 -  Se retorna token al cliente

Validación de Requests

1. **JWT Filter** intercepta cada request
2. **Extrae token** del header Authorization
3. **Valida token:** firma, expiración, formato
4. **Carga usuario** y establece contexto de seguridad
5. **Verifica permisos** según rol del usuario
6. **Permite o deniega** acceso al endpoint

Configuración de Seguridad

Endpoints Públicos

```
java

.requestMatchers("/register", "/login", "/validate").permitAll()

.requestMatchers("/swagger-ui/**", "/v3/api-docs/**").permitAll()
```

Endpoints Protegidos por Rol

```
java

// Solo ADMINISTRADOR

.requestMatchers("/api/usuarios/**").hasRole("ADMINISTRADOR")
.requestMatchers("/api/roles/**").hasRole("ADMINISTRADOR")

// ADMINISTRADOR y ENCARGADO_INVENTARIO

.requestMatchers(HttpMethod.POST, "/api/productos/**")
    .hasAnyRole("ADMINISTRADOR", "ENCARGADO_INVENTARIO")
.requestMatchers(HttpMethod.PUT, "/api/productos/**")
    .hasAnyRole("ADMINISTRADOR", "ENCARGADO_INVENTARIO")
.requestMatchers(HttpMethod.DELETE, "/api/productos/**")
    .hasAnyRole("ADMINISTRADOR", "ENCARGADO_INVENTARIO")

// Todos los roles autenticados (para GET)

.requestMatchers(HttpMethod.GET, "/api/productos/**")
    .hasAnyRole("ADMINISTRADOR", "ENCARGADO_INVENTARIO", "VENDEDOR")
```

Medidas de Seguridad Implementadas

Encriptación de Contraseñas

- **Algoritmo:** BCrypt con salt automático
- **Verificación:** En cada login se compara hash
- **Nunca** se almacenan contraseñas en texto plano

JWT Security

- **Firma secreta:** Clave privada para validar autenticidad
- **Expiración:** Tokens válidos por 1 hora
- **Claims personalizados:** Username y rol incluidos
- **Validación:** En cada request protegido

Protección CORS

- **Origen permitido:** http://localhost:3000
- **Métodos:** GET, POST, PUT, DELETE, OPTIONS
- **Headers:** Authorization, Content-Type permitidos
- **Credentials:** Habilitado para cookies/auth headers

Autorización Granular

- **Método específico:** Diferentes permisos por HTTP method
- **Recurso específico:** Control por endpoint
- **Rol específico:** Verificación de rol en cada request

Configuración JWT

Generación de Token

```
java
// Token incluye:
- Subject: username del usuario
- Claim "role": rol del usuario
- Issued At: timestamp de creación
- Expiration: 1 hora desde creación
- Signature: firmado con clave secreta
```

Validación de Token

```
java
// Validaciones aplicadas:
- Formato JWT válido
- Firma válida con clave secreta
- Token no expirado
- Username existe en sistema
- Rol válido
```

Manejo de Errores de Seguridad

401 Unauthorized

- Token inválido o expirado
- Credenciales incorrectas
- Token malformado

403 Forbidden

- Usuario autenticado pero sin permisos suficientes
- Rol no autorizado para el endpoint
- Intento de acceso a recurso restringido

Matriz de Permisos

Endpoint	ADMINISTRADOR	ENCARGADO_INVENTARIO	VENDEDOR
GET /api/usuarios	✓	×	×
POST /api/usuarios	✓	×	×
GET /api/productos	✓	✓	✓
POST /api/productos	✓	✓	×
PUT /api/productos	✓	✓	×
DELETE /api/productos	✓	✓	×
GET /api/categorias	✓	✓	✓
POST /api/categorias	✓	✓	×
GET /api/proveedores	✓	✓	×
POST /api/movimientos	✓	✓	×

Buenas Prácticas Implementadas

Seguridad del Token

- ☒ Tiempo de vida limitado (1 hora)
- ☒ Firma criptográfica segura
- ☒ No almacenamiento en servidor (stateless)
- ☒ Incluye información mínima necesaria

Gestión de Contraseñas

- ☒ Encriptación con BCrypt
- ☒ Validación de unicidad de usuarios
- ☒ No exposición en responses
- ☒ Verificación segura en login

Control de Acceso

- ☒ Principio de menor privilegio
- ☒ Separación de responsabilidades por rol
- ☒ Validación en cada request
- ☒ Mensajes de error seguros (sin información sensible)