Proyecto: Implementación del Sistema de Gestión de Ventas

1. Objetivo del Proyecto

El objetivo de este proyecto es que los estudiantes demuestren su capacidad para desarrollar una aplicación web completa y robusta utilizando el ecosistema Spring Boot, integrando conceptos clave de persistencia de datos, validación y seguridad. Deberán transformar los requisitos definidos en el ERS proporcionado en un sistema funcional de gestión de ventas.

2. Descripción de la Tarea

Basándose en el Documento de Especificación de Requisitos de Software (ERS) para el Sistema de Gestión de Ventas, los estudiantes deberán implementar una aplicación web que cubra los siguientes módulos y requisitos:

Módulos a Implementar:

1. Gestión de Clientes (RF1):
    - Implementar todas las operaciones CRUD (Crear, Consultar, Actualizar, Eliminar) para la entidad Cliente.
    - Integrar la seguridad para la autenticación: El campo correo del cliente servirá como username y se deberá almacenar la contraseña codificada con BCrypt.
    - Asignar un rol (ROLE_USER o ROLE_ADMIN) a cada cliente para la autorización.
    - Aplicar todas las reglas de validación (RF1.1.1) en el formulario de registro y edición.
    - Considerar la restricción de clave foránea al eliminar clientes (RF1.4).
2. Gestión de Productos (RF2):
    - Implementar todas las operaciones CRUD (Crear, Consultar, Actualizar, Eliminar) para la entidad Producto.
    - Aplicar todas las reglas de validación (RF2.1.1) en el formulario de registro y edición.
    - Considerar la restricción de clave foránea al eliminar productos (RF2.4).
3. Gestión de Ventas (RF3):
    - Implementar las operaciones de Registro de Venta (RF3.1):
        - Permitir seleccionar un cliente existente.
        - Permitir añadir múltiples productos a una misma venta con sus respectivas cantidades (usando un enfoque dinámico en el formulario, como botones "Añadir/Eliminar Producto").
        - Asegurar que la lógica de stock se maneje correctamente (verificar disponibilidad y reducir stock al guardar).

        - Registrar el precioUnitario del producto al momento de la venta en el detalle.
        - Calcular y almacenar el total de la venta.
        - Asegurar que la operación de guardar venta sea transaccional (si el stock no es suficiente para un producto, toda la venta debe revertirse).
    - Implementar la operación de Consulta de Ventas (RF3.2):
        - Mostrar la lista de ventas, incluyendo cliente y total.
        - Al visualizar una venta específica (edición/detalle), mostrar todos sus DetalleVenta asociados.
    - Implementar la operación de Eliminar Venta (RF3.4):
        - Al eliminar una venta, el stock de los productos involucrados debe ser revertido.
        - La operación de eliminación debe ser transaccional.

3. Requisitos No Funcionales (RNF) Clave a Demostrar:

- Seguridad (RNF4, RNF5, RNF6, RNF7, RNF8):
    - Configurar Spring Security para exigir autenticación.
    - Hashear contraseñas con BCryptPasswordEncoder.
    - Implementar autenticación basada en base de datos (UserDetailsService personalizado).
    - Proteger URLs según los roles (ADMIN, USER) como se especifica en el ERS (ej. /productos/registro solo para ADMIN).
    - Asegurar la protección CSRF.
    - Manejar el acceso denegado.
- Usabilidad (RNF9, RNF10, RNF11, RNF12):
    - La interfaz debe ser intuitiva y el diseño responsivo utilizando Tailwind CSS.
    - Los mensajes de validación deben ser claros y visibles en el formulario.
- Mantenibilidad (RNF13, RNF14):
    - El código debe seguir una arquitectura en capas (MVC, Servicio, Repositorio, Modelo).
    - Uso de Lombok para reducir código repetitivo.
    - Código limpio, legible y con comentarios adecuados donde sea necesario.
- Fiabilidad (RNF18, RNF19):
    - Correcto uso de validaciones (Jakarta Bean Validation) a nivel de aplicación.
    - Manejo de restricciones de base de datos (UNIQUE, FOREIGN KEY, NOT NULL).

- Implementación de transacciones en operaciones que modifiquen múltiples entidades (ej. guardarVenta, eliminarVenta).

4. Tecnologías y Herramientas

- Backend: Spring Boot 3+, Spring Web, Spring Data JPA, Jakarta Bean Validation, Lombok, Spring Security.
- Base de Datos: MySQL 8+.
- Frontend: Thymeleaf (HTML), Tailwind CSS.
- Gestión de Dependencias: Apache Maven.
- IDE: NetBeans (o similar).

5. Entregables

1. Repositorio Git (GitHub/GitLab): Conteniendo el código fuente completo del proyecto.
   - Debe incluir un archivo README.md detallado con:
     - Instrucciones claras para clonar, construir y ejecutar el proyecto.
     - Credenciales de usuarios de prueba (ej. user@example.com/password, admin@example.com/adminpass).
     - Comandos SQL para crear la base de datos y un par de usuarios iniciales.
2. Demostración (En Clase): Estar preparado para una demostración funcional del sistema, mostrando las funcionalidades implementadas y la aplicación de los requisitos de seguridad y validación.

6. Consideraciones Adicionales

- Se valorará la originalidad en el diseño UI/UX (dentro de las limitaciones de Thymeleaf y Tailwind) y la robustez del manejo de errores.
- La prioridad en el desarrollo debe ser la funcionalidad correcta y la calidad del código.

Rúbrica de Evaluación del Proyecto

Criterio	Peso (%)	Excelente (4 puntos)	Notable (3 puntos)	Suficiente (2 puntos)	Insuficiente (1 punto)
1. Implementación Requisitos Funcionales (40%)					
1.1 Gestión de Clientes (CRUD y Auth)	15%	Funcionalidad CRUD completa, robusta y con autenticación integrada. Contraseñas hasheadas y roles gestionados correctamente.	Funcionalidad CRUD completa y autenticación implementada. Pequeñas omisiones o detalles mejorables en el manejo de contraseñas/roles.	Funcionalidad CRUD parcial o con errores. Autenticación básica pero incompleta o con fallos.	Funcionalidad CRUD o autenticación no implementada o con errores graves.
1.2 Gestión de Productos (CRUD)	10%	Funcionalidad CRUD completa y robusta. Todas las validaciones y restricciones de DB consideradas.	Funcionalidad CRUD completa con la mayoría de validaciones. Pequeñas deficiencias.	Funcionalidad CRUD parcial o con errores. Validaciones incompletas o ineficaces.	Funcionalidad CRUD no implementada o con errores graves.
1.3 Gestión de Ventas (Registro, Consulta, Eliminación con Stock)	15%	Registro de ventas robusto (múltiples productos, stock, total, transaccionalidad). Consulta y eliminación (con reversión de stock) impecables.	Registro de ventas funcional, con manejo de stock y total. Consulta y eliminación operativas. Pequeños detalles a pulir.	Registro de ventas básico, manejo de stock con fallos o sin transaccionalidad. Consulta o eliminación parciales.	Módulo de ventas no implementado o con errores críticos.
2. Implementación de Seguridad (25%)					
2.1 Autenticación (BCrypt, UserDetailsService)	10%	Autenticación basada en DB con UserDetailsService y BCryptPasswordEncoder correctamente configurados y funcionales.	Autenticación funcional, pero con detalles menores en la implementación del UserDetailsService o PasswordEncoder.	Autenticación implementada pero con fallos, o uso incorrecto de hashing/UserDetailsService.	Autenticación no funcional o ausente.
2.2 Autorización (Roles, Protección URLs)	10%	Reglas de autorización por rol (ADMIN/USER) correctamente aplicadas a todas las URLs según el ERS. Manejo de acceso denegado funcional.	La mayoría de reglas de autorización implementadas. Pequeñas fallas en el mapeo de roles o URLs.	Autorización parcial o con errores. Algunas URLs no protegidas o roles no asignados correctamente.	Autorización no implementada o completamente ineficaz.

2.3 Protección CSRF	5%	Protección CSRF habilitada y funcional, sin errores de interacción.	Protección CSRF habilitada y funcional con pequeños problemas de formulario.	Protección CSRF habilitada pero con problemas que afectan la usabilidad.	Protección CSRF ausente o deshabilitada.
3. Calidad de Código y Arquitectura (20%)					
3.1 Separación de Capas (MVC, Servicio, Repositorio)	10%	Excelente separación de capas, lógica bien distribuida entre Controlador, Servicio, Repositorio y Modelo.	Buena separación de capas, con algunas responsabilidades ligeramente difusas.	Separación de capas deficiente, con lógica mezclada o responsabilidades confusas.	Sin separación de capas clara o arquitectura caótica.
3.2 Uso de Tecnologías y Buenas Prácticas	10%	Uso óptimo de JPA/Spring Data JPA, Jakarta Bean Validation, Lombok. Código limpio, legible, con nombres significativos y comentarios claros.	Buen uso de las tecnologías, código legible. Algunas convenciones o prácticas podrían mejorarse.	Uso básico de las tecnologías, código con baja legibilidad, nombres inconsistentes o falta de comentarios.	Mal uso de las tecnologías, código ilegible, o plagio evidente.
4. Cumplimiento de RNF y Robustez (10%)					
4.1 Manejo de Errores y Excepciones	5%	Manejo de errores y excepciones robusto, con mensajes informativos para el usuario y logs adecuados.	Manejo de errores presente, pero podría ser más granular o informativo.	Manejo de errores básico, con posibles excepciones no controladas o mensajes poco claros.	Sin manejo de errores o la aplicación falla inesperadamente.
4.2 Restricciones DB y Transaccionalidad	5%	Todas las restricciones de DB aplicadas correctamente (UNIQUE, FK, NOT NULL). Transacciones para operaciones críticas garantizadas.	La mayoría de restricciones DB aplicadas. Transacciones para operaciones críticas correctas.	Restricciones DB incompletas o incorrectas. Transaccionalidad ausente o con fallos.	Restricciones DB no aplicadas o fallos en la integridad de los datos.
5. Documentación y Entrega (5%)					
5.1 README.md y Repositorio	5%	README.md completo y claro con instrucciones de configuración y ejecución. Repositorio bien organizado y con historial de commits significativo.	README.md con instrucciones suficientes. Repositorio organizado.	README.md incompleto o confuso. Repositorio con organización deficiente.	README.md ausente. Repositorio desorganizado o no funcional.
Puntuación Total	100%				