*3Retail S.A*

**Análisis y Diseño del Sistema - Tienda de Instrumentos Musicales**

*Iniciación y pasos a seguir para el proyecto grupal de ciberseguridad primer semestre.*

Versión 1.0

Historia de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 9/05/2025 | 1.0 | Primera versión del documento para realizar el proyecto del profesor Daniel Tijerino | Melany Poveda  Jurgen Altamirano |

Contenido

[1. Introducción 3](#_Toc75411093)

[1.1. iniciacion 3](#_Toc75411094)

[1.2. alcance 3](#_Toc75411095)

[1.3. Definiciones, siglas y abreviaturas. 3](#_Toc75411096)

[1.4. Referencias 3](#_Toc75411097)

[1.5. Visión general 3](#_Toc75411098)

[2. Descripción general 4](#_Toc75411099)

[2.1. Estudio del Modelo de Casos de Uso 4](#_Toc75411100)

[2.2. Perspectiva del producto 4](#_Toc75411101)

[2.2.1. Interfases de usuario 4](#_Toc75411102)

[2.2.2. Interfases con hardware 5](#_Toc75411103)

[2.2.3. Interfases con software 5](#_Toc75411104)

[2.2.4. Interfases de comunicación 5](#_Toc75411105)

[2.2.5. Restricciones de memoria 5](#_Toc75411106)

[2.2.6. Requerimientos de adecuación al entorno 5](#_Toc75411107)

[2.3. Funciones del producto 5](#_Toc75411108)

[2.4. Características de los usuarios 5](#_Toc75411109)

[2.5. Restricciones de diseño 6](#_Toc75411110)

[2.5.1. [Restricción de diseño 1] 6](#_Toc75411111)

[2.6. Supuestos y dependencias 6](#_Toc75411112)

[3. Requerimientos específicos 6](#_Toc75411113)

[3.1. Modelo de Casos de Uso 6](#_Toc75411114)

[3.2. Requerimientos Suplementarios 6](#_Toc75411115)

[4. Requerimientos de documentación 6](#_Toc75411116)

[4.1. Manual de Usuario 6](#_Toc75411117)

[4.2. Ayuda en línea 7](#_Toc75411118)

[4.3. Guías de instalación, configuración y archivo Léame. 7](#_Toc75411119)

[4.4. Etiquetado y empaquetado 7](#_Toc75411120)

[5. [Información de soporte (opcional) ] 7](#_Toc75411121)

**1. Descripción General del Sistema**

El sistema permite gestionar:

* Usuarios con roles y permisos diferenciados (administrador, cliente).
* Información detallada de clientes.
* Catálogo de instrumentos musicales con stock y precios.
* Registro de pedidos con detalle de productos y cantidades.
* Procedimientos almacenados para asegurar integridad en operaciones críticas.
* API REST para interacción con frontend y aplicaciones móviles.

**2. Estructura y Módulos del Sistema**

**2.1 Usuarios y Roles**

* **Función:** Controlar accesos y privilegios.
* **Tablas:** Usuarios, Roles.
* **Seguridad:** Contraseñas almacenadas con hash bcrypt.
* **Roles definidos:** Administrador y Cliente.

**2.2 Gestión de Clientes**

* **Función:** Registrar clientes con datos básicos y dirección.
* **Validaciones:** Email único y formato válido.
* **Tabla:** Clientes.

**2.3 Catálogo de Productos**

* **Función:** Control de instrumentos disponibles para venta.
* **Validaciones:** Precio y stock no negativos; nombre único.
* **Tabla:** Productos.

**2.4 Pedidos y Detalles**

* **Función:** Registro de ventas con productos y cantidades.
* **Tablas:** Pedidos, DetallesPedido.
* **Integridad:** Validación de stock al crear pedido

**3. Procedimientos Almacenados**

**Procedimiento CrearPedido**

* **Parámetros:** ID cliente, lista de productos en formato JSON.
* **Pasos:**
  1. Inicia transacción.
  2. Inserta nuevo pedido.
  3. Recorre productos:
     + Verifica stock.
     + Inserta detalle.
     + Actualiza stock.
  4. Calcula total y actualiza pedido.
  5. Confirma o revierte transacción según resultado.

**4. API REST con Flask**

**Tecnologías:**

* Python
* Flask y Flask-CORS
* MySQL Connector
* Bcrypt para seguridad

**Endpoints principales:**

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Ruta | Descripción | Rol requerido |
| GET | / | Mensaje bienvenida | Público |
| POST | /login | Autenticación con email y contraseña | Público |
| GET | /productos | Listar productos | Público |
| POST | /productos | Agregar producto | Admin o Vendedor |
| PUT | /productos/<id> | Editar producto | Admin o Vendedor |
| DELETE | /productos/<id> | Eliminar producto | Admin |
| GET | /usuarios/<id> | Datos usuario por ID | Admin |
| POST | /pedidos | Crear nuevo pedido | Cliente |
| GET | /pedidos/<id> | Detalle de pedido | Cliente o Admin |
| GET | /clientes | Listar clientes | Admin |
| POST | /clientes | Crear cliente | Admin |
| PUT | /clientes/<id> | Actualizar cliente | Admin |
| DELETE | /clientes/<id> | Eliminar cliente | Admin |

**Seguridad:**

* Hash bcrypt para contraseñas.
* Validación de roles para autorización.
* Validaciones de integridad (stock, datos).

**5. Validaciones y Restricciones**

* Email único con formato válido.
* Precio y stock ≥ 0.
* Cantidad en pedido > 0 y stock suficiente.
* Control de acceso basado en roles.
  1. **Datos de Prueba (SQL)**

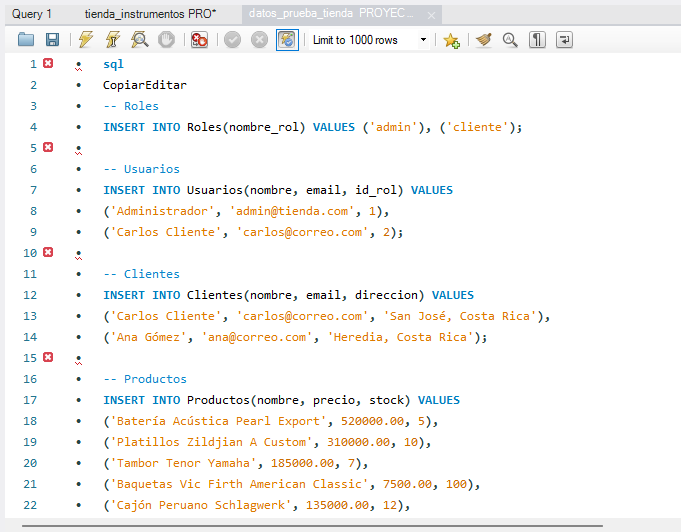
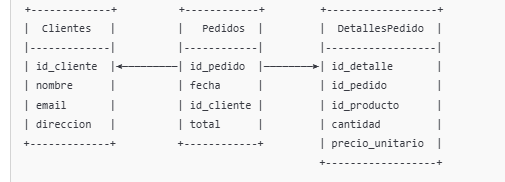


Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

**7. Diagramas**

* 1. **Diagrama Entidad-Relación (ER)**



* **Clientes**: contiene información de cada persona que hace compras (nombre, email, dirección).
* **Pedidos**: representa una compra realizada por un cliente en una fecha específica.
* **Relación**: un cliente puede tener muchos pedidos (**relación 1 a N**).
* **Detalles Pedido**: representa los productos específicos comprados en cada pedido.
  + Incluye cantidad y precio unitario en ese momento (importante si los precios cambian con el tiempo).
  + Está relacionado con Pedidos y Productos

**7.2 Usuarios – Roles diagrama**

Diagrama

El contenido generado por IA puede ser incorrecto.

* **Usuarios**: personas que acceden al sistema (como administradores o empleados).
* **Roles**: define el nivel de permisos del usuario (ej: admin, cliente, vendedor).
* **Relación**: un usuario tiene un único rol (**relación N a 1**).

**7.3. Productos diagrama**

**Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.**

* Contiene todos los productos disponibles en la tienda: nombre, precio y stock actual.
* Se relaciona con DetallesPedido, ya que cada detalle de pedido corresponde a un producto específico.
* Esta tabla permite gestionar el inventario.

**9. Código Ejemplo para API (Flask)**

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Entidades principales:**

1. **Roles**
   * id\_rol (PK)
   * nombre\_rol
2. **Usuarios**
   * id\_usuario (PK)
   * nombre
   * email
   * password\_hash
   * id\_rol (FK → Roles)
3. **Clientes**
   * id\_cliente (PK)
   * nombre
   * email
   * Direccion
4. **Productos**
   * id\_producto (PK)
   * nombre
   * precio
   * stock
   * imagen\_url
   * Caracteristicas
5. **Pedidos**
   * id\_pedido (PK)
   * id\_cliente (FK → Clientes)
   * fecha
   * total
6. **DetallesPedido**
   * id\_detalle (PK)
   * id\_pedido (FK → Pedidos)
   * id\_producto (FK → Productos)
   * cantidad
   * precio\_unitario

**Relaciones clave:**

* Un **rol** puede tener muchos **usuarios** → Relación 1:N
* Un **cliente** puede hacer muchos **pedidos** → Relación 1:N
* Un **pedido** tiene muchos **detalles de pedido** → Relación 1:N
* Un **producto** puede estar en muchos **detalles de pedido** → Relación 1:N
* La tabla **DetallesPedido** actúa como entidad intermedia para una relación N:M entre **Pedidos** y **Productos**

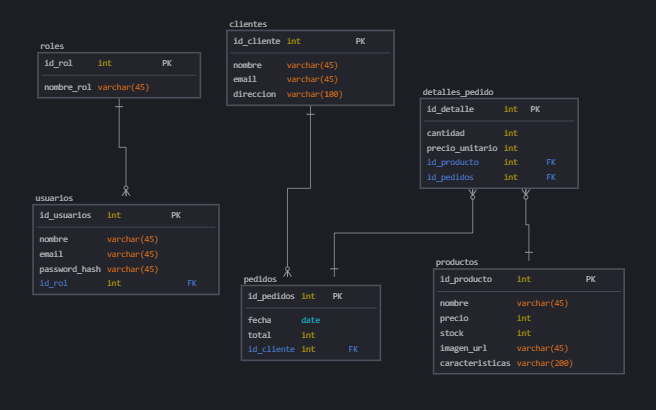
**Resumen visual (estructura simplificada)**

scss

Copiar código

Roles (1) ────< (N) Usuarios

Clientes (1) ────< (N) Pedidos (1) ────< (N) DetallesPedido >──── (1) Productos



# Versión 2.0 - Documento Ampliado

**Fecha: 27/05/2025**

**Descripción: Documento ampliado con información técnica detallada sobre las mejoras realizadas al sistema, incluyendo cambios en la base de datos, interfaz, seguridad, pruebas y arquitectura del software.**

## Resumen de Cambios Principales

- Inclusión del rol de Vendedor para permitir gestión compartida de productos.  
- Interfaces HTML rediseñadas con validaciones de entrada en el frontend.  
- Campo 'imagen\_url' en la base de datos para productos con representación visual.  
- Triggers para auditoría de cambios sensibles en usuarios y productos.  
- Funciones SQL para cálculos automáticos y validaciones.  
- Validaciones de integridad cruzadas entre pedidos y stock en base de datos.  
- Control de acceso ampliado en la API con autenticación basada en roles.  
- Manejo de errores detallado en el backend para respuestas HTTP claras.  
- Pruebas de seguridad: validación de entradas, protección contra inyección SQL.

## Interfaces HTML Desarrolladas

1. login.html:  
 - Formulario de acceso con validación de correo electrónico y contraseña.  
 - Integración con backend vía fetch y token de sesión.  
2. crear\_usuario.html:  
 - Formulario con campos nombre, email, rol y contraseña.  
 - Validaciones de formato y restricciones por rol.  
3. home.html:  
 - Vista general después del inicio de sesión.  
 - Contenido dinámico basado en el tipo de usuario.  
4. pagina\_Tienda\_instrumentos\_modificada.html:  
 - Visualización de catálogo con imágenes, nombre y precio.  
 - Botones de compra o edición según permisos del usuario.

## Cambios en la Base de Datos

- Campo 'imagen\_url' en la tabla Productos.  
- Triggers para registrar inserciones y eliminaciones en tabla Usuarios.  
- Función calcular\_total\_pedido(pedido\_id) para automatizar el cálculo total.  
- Validaciones de stock al momento de registrar un pedido (control desde stored procedure).  
- Script 'datos\_prueba\_tienda.sql' para poblar la base con datos realistas de productos y usuarios.

## Seguridad y Validaciones Implementadas

- Hashing de contraseñas con bcrypt.  
- Validación de correo único y formato correcto.  
- Control de accesos basado en roles desde el backend.  
- Manejo de errores con códigos HTTP adecuados y mensajes informativos.  
- Sanitización de entradas para prevenir ataques XSS y SQL Injection.

## Pruebas de Seguridad y Funcionalidad

- Pruebas de inyección SQL en formularios de login y creación de usuario.  
- Pruebas de acceso a rutas protegidas sin autenticación.  
- Validación de campos vacíos, incorrectos o duplicados.  
- Carga de datos de prueba verificada con INSERTs desde SQL.

## Recomendaciones Futuras

- Incluir cifrado HTTPS en servidor Flask para seguridad de transporte.  
- Implementar sesiones con JWT para mayor robustez.  
- Sistema de recuperación de contraseña y verificación por email.  
- Módulo de reportes para administración avanzada.  
- Sistema de logs y dashboard de estadísticas de uso.

**Conclusiones**

Este sistema permite gestionar de manera eficiente y segura todos los aspectos relevantes para una tienda de instrumentos musicales, desde la administración de usuarios y roles, hasta el control de productos, clientes y pedidos. La implementación de procedimientos almacenados asegura la integridad de las transacciones, mientras que la API REST ofrece un punto de acceso flexible y seguro para aplicaciones frontend o móviles.

El diseño presentado cumple con buenas prácticas de desarrollo y modelado, garantizando escalabilidad, seguridad y mantenibilidad, aspectos fundamentales para un sistema que debe crecer y adaptarse a futuras necesidades.