

# **Especificación de Requisitos del Sistema (ERS) - Colmotica Backend**

Versión: 1.0 Fecha: 2025-10-20 Autor: Julián Alexander Gutiérrez Mercado

## **1. Propósito**

Este documento define la primera versión de la Especificación de Requisitos del Sistema (ERS) para el backend del proyecto Colmotica. Su objetivo es describir el comportamiento esperado del sistema, su alcance, las funcionalidades principales, reglas de negocio, requisitos funcionales y no funcionales, así como el modelo de datos lógico básico y un glosario de términos.

## **2. Alcance del producto**

El backend de Colmotica provee una API REST (montada bajo el prefijo /colmotica) que gestiona: - Registro y gestión de usuarios. - Autenticación y verificación de cuentas. - Gestión de administradores. - Gestión de manuales (creación, listado) y solicitudes de acceso a manuales por parte de usuarios. - Envío de correos electrónicos para verificación, recuperación y notificaciones relacionadas a manuales.

El sistema no incluye: interfaz de usuario (frontend), migraciones SQL automáticas ni despliegue automático (estos pueden añadirse en próximas versiones).

## **3. Funcionalidades del producto (resumen)**

- Gestión de usuarios: registro, listado, actualización parcial, eliminación.
- Autenticación: login y verificación de credenciales.
- Recuperación de contraseña mediante código enviado por correo.
- Emisión y verificación de códigos (tipo: VERIFICACION, RECUPERACION).
- Gestión de manuales: crear manuales y listar manuales disponibles.
- Solicitudes de manuales: crear solicitudes por usuario, listar pendientes, aprobar/rechazar.
- Gestión de administradores: crear y listar administradores.
- Envío de notificaciones por correo en eventos relevantes (aprobación/rechazo, verificación, recuperación).

## 4. Clases y características de usuarios

Se reconocen las siguientes clases de usuario en el sistema (según ID\_ROL):

- Usuario estándar (rol por defecto: 10003)
  - Permisos: registrarse, solicitar manuales, recuperar contraseña, actualizar su perfil.
  - Atributos principales: ID\_USERS (UUID), EMAIL, NAME, TEL, PAIS, PASS\_HASH, VERIFIED (0/1).
- Administrador (rol = 10002)
  - Permisos: crear manuales (según endpoints expuestos), aprobar o rechazar solicitudes de manuales.
  - Atributos iguales a usuario con rol distinto y privilegios mayores.
- Super Administrador (rol = 10001)
  - Permisos: Todos los permisos del rol 10002, pero gestiona a los administradores; agrega, elimina y consulta la lista de administradores.

Características de credenciales: - Contraseña: almacenada como hash (bcrypt o servicio sColmoticaService). Reglas de complejidad aplicadas en validación (min 8 caracteres, al menos una mayúscula, una minúscula y un número).

## 5. Entorno operativo

- Plataforma: Node.js (el repositorio usa TypeScript).
- Framework web: Express 5.
- Base de datos: MariaDB.
- Puerto por defecto: `process.env.PORT` o 1234.
- Dependencias relevantes: `express`, `cors`, `mariadb`, `bcrypt`, `jsonwebtoken`, `nodemailer`, `zod`.
- Desarrollo: `ts-node-dev` para hot reload.

## 6. Requerimientos funcionales y no funcionales (con códigos)

Nota: los códigos siguen el patrón RF-### para funcionales y RNF-### para no funcionales.

### 6.1 Requerimientos funcionales (RF)

- RF-001: Registro de usuario
  - Descripción: Permitir que un nuevo usuario se registre con EMAIL, NAME, TEL, PAIS, PASS\_HASH (contraseña) y opcionalmente ID\_ROL.
  - Endpoint: `POST /colmotica/users`
  - Validación: según `userSchema` (Zod): email válido, pass con complejidad, campos con longitudes máximas.

- Criterio de aceptación: devuelve 201 y el objeto creado (ID\_USERS generado) en caso de éxito; 422 para datos inválidos; 409 o null cuando el correo ya existe.
- RF-002: Listar usuarios
  - Descripción: Obtener la lista de usuarios registrados.
  - Endpoint: GET /colmotica/users
  - Criterio de aceptación: devuelve 200 con array de usuarios.
- RF-003: Login / Autenticación
  - Descripción: Validar credenciales (EMAIL + PASS\_HASH), comparar el hash y devolver éxito o fallo de autenticación.
  - Endpoint: POST /colmotica/users/login
  - Criterio de aceptación: 201 con datos de sesión (por ahora devuelve success true y datos enviados) o 401 en credenciales incorrectas.
- RF-004: Envío de código de verificación
  - Descripción: Insertar un código de verificación o recuperación y enviar por correo.
  - Endpoint: POST /colmotica/users/sendcode (y otros métodos de sMailService utilizados internamente)
  - Criterio de aceptación: 200 si correo enviado; 400 si no se genera código.
- RF-005: Verificación de código
  - Descripción: Verificar un código para activar/verificar cuenta.
  - Endpoint: POST /colmotica/users/verify-code
  - Criterio de aceptación: 200 si el código es válido; 400 si inválido.
- RF-006: Recuperación y actualización de contraseña
  - Descripción: Cambiar la contraseña a partir de email y código/flujo de recuperación.
  - Endpoint: POST /colmotica/users/recover-pass
  - Criterio de aceptación: 200 con mensaje de éxito al actualizar contraseña.
- RF-007: Crear manual
  - Descripción: Registrar un manual con ID\_MANUALS, ID\_ROL y NAME.
  - Endpoint: POST /colmotica/users/manuals
  - Criterio de aceptación: 201 con objeto del manual creado.
- RF-008: Listar manuales
  - Descripción: Obtener todos los manuales disponibles.
  - Endpoint: GET /colmotica/users/manuals
  - Criterio de aceptación: 200 con array de manuales.
- RF-009: Solicitar acceso a un manual
  - Descripción: Un usuario solicita un manual; el sistema registra la solicitud y limita solicitudes pendientes por usuario.
  - Endpoint: POST /colmotica/manuals/req

- Criterio de aceptación: 201 si la solicitud es aceptada; 429 si el usuario alcanzó el límite de solicitudes pendientes.
- RF-010: Listar solicitudes pendientes
  - Descripción: Obtener solicitudes con estado 'PENDIENTE'.
  - Endpoint: GET /colmotica/manuals/req/pendiente
  - Criterio de aceptación: 200 con lista de solicitudes pendientes.
- RF-011: Aprobar o rechazar solicitud de manual
  - Descripción: Cambiar el estado de MANUALS\_VS\_USERS a APROVADO o RECHAZADO, establecer DATE\_APPROVED y notificar por correo.
  - Endpoint: PATCH /colmotica/manuals/req/aprobado PATCH /colmotica/manuals/req/rechazado
  - Criterio de aceptación: 201 y envío de correo de notificación.
- RF-012: Gestión de administradores (crear/listar)
  - Descripción: Crear administrador y listar administradores (usuarios con ID\_ROL = 10002).
  - Endpoint: POST /colmotica/users/admin, GET /colmotica/users/admin
  - Criterio de aceptación: 201 al crear y 200 al listar.

## 6.2 Requerimientos no funcionales (RNF)

- RNF-001: Seguridad de contraseñas
  - Descripción: Las contraseñas deben almacenarse como hash. Requisito implementado por sColmoticaService / bcrypt.
  - Criterio: ninguna contraseña se almacena en texto plano.
- RNF-002: Validación de entrada
  - Descripción: Todas las entradas públicas deben validarse con Zod (esquemas definidos en schemas.ts).
  - Criterio: peticiones con datos inválidos devuelven 422.
- RNF-003: Rendimiento básico
  - Descripción: Responder peticiones REST típicas en menos de 500ms bajo carga baja (puede requerir pruebas de carga).
- RNF-004: Disponibilidad y tolerancia a fallos
  - Descripción: El servicio debe manejar errores internos y devolver códigos HTTP apropiados (500 para errores de servidor) y registrar errores en consola.
- RNF-005: Compatibilidad
  - Descripción: Ejecutable en Node.js LTS, compatible con MariaDB 10.3+.

## **7. Funcionalidades Generales:**

### **Gestión de usuarios**

Descripción extendida: Permite a los usuarios registrarse, autenticarse, recuperar contraseña y actualizar su perfil. Validaciones estrictas en contraseñas y emails. Registro genera ID\_USERS (UUID) y almacena contraseña hasheada.

Entradas/Salidas principales: - Entrada: JSON con campos según userSchema. - Salida: JSON con success y datos o error.

Requisitos relacionados: RF-001, RF-002, RF-003, RF-004, RF-005, RF-006.

### **Gestión de manuales y solicitudes**

Descripción extendida: CRUD básico para manuales (actualmente solo crear y listar) y gestión de solicitudes por usuario. Límite de solicitudes pendientes por usuario: 5.

Entradas/Salidas principales: - Entrada: JSON con ID\_MANUALS y ID\_USERS al solicitar un manual. - Salida: confirmación de solicitud o error (429 si límite alcanzado).

Requisitos relacionados: RF-007, RF-008, RF-009, RF-010, RF-011.

### **Notificaciones por correo**

Descripción: Envío de correos para verificación, recuperación y notificar aprobación/rechazo de solicitudes. Servicio responsable: sMailService.

Requisitos relacionados: RF-004, RF-005, RF-011.

## 8. Reglas de negocio

- RN-BUS-001: Un usuario no puede tener más de 5 solicitudes de manual en estado PENDIENTE (implementado en `mManuals.mreqManual`).
- RN-BUS-002: No se permiten registros con el mismo EMAIL (comprobación antes de insertar en `mUser.mpostRegistro`).
- RN-BUS-003: Los códigos de verificación y recuperación se insertan con STATUS = 1 y se obtienen ordenando por CREATED\_AT descendente para obtener el último código activo.
- RN-BUS-004: Solo usuarios con ID\_ROL = 10002 se consideran administradores en listados (`mAdmins.mgetAdmins`).
- RN-BUS-005: Las contraseñas deben cumplir la política de complejidad definida en `schemas.ts`.

## 9. Requerimientos no funcionales

- RNF-006: Logging mínimo
  - Descripción: El backend debe registrar en consola los errores y eventos importantes (ya hay logs en consola en controladores y modelos).
- RNF-007: Formato de API
  - Descripción: Todas las respuestas JSON deben seguir el patrón: { success: boolean, result?: any, message?: string, error?: any } cuando aplique.
- RNF-008: Internacionalización y localización
  - Descripción: Mensajes básicos retornados en español (actual comportamiento). En versiones futuras se puede parametrizar.

## 10. Otros requerimientos

- OR-001: Variables de entorno
  - El sistema depende de variables de entorno para configuración de puerto, credenciales DB y servicios de correo (archivo `.env`).
- OR-002: Migraciones
  - Actualmente no hay migraciones automáticas. Se asume que la base de datos contiene las tablas descritas en el modelo lógico.

## 11. Modelo lógico de datos (resumen y entidades principales)

Tablas principales (campo : tipo aproximado):

- USERS
  - ID\_USERS : UUID (PK)

- ID\_ROL : INT
- EMAIL : VARCHAR(255) (UNIQUE)
- PAIS : VARCHAR(255)
- TEL : VARCHAR(20)
- NAME : VARCHAR(255)
- PASS\_HASH : VARCHAR(255)
- VERIFIED : TINYINT (0/1)
- MANUALS
  - ID\_MANUALS : UUID (PK)
  - ID\_ROL : INT (rol requerido para acceder o crear)
  - NAME : VARCHAR(255)
  - CREATE\_AT : DATETIME
- MANUALS\_VS\_USERS (solicitudes)
  - ID\_MANUALS\_VS\_USERS : BIGINT (PK, generado con COALESCE(MAX+1,1))
  - ID\_MANUALS : UUID (FK -> MANUALS)
  - ID\_USERS : UUID (FK -> USERS)
  - STATE : VARCHAR (ej. 'PENDIENTE','APROVADO','RECHAZADO')
  - DATE\_REQ : DATETIME
  - DATE\_APPROVED : DATETIME
- LOG\_MANUAL
  - ID\_LOG : UUID (PK)
  - ID\_USERS : UUID
  - ID\_MANUALS : UUID
  - CREATE\_AT : DATETIME
- CODE\_VERIFICATION
  - ID\_CODE : UUID (PK)
  - ID\_USERS : UUID (FK -> USERS)
  - CONTENT : INT (código)
  - CREATED\_AT : DATETIME
  - STATUS : INT (1 activo, 0 inactivo)
  - TYPE : VARCHAR (ej. 'VERIFICACION','RECUPERACION')

Relaciones principales: - USERS 1..\* MANUALS\_VS\_USERS (un usuario puede hacer muchas solicitudes) - MANUALS 1..\* MANUALS\_VS\_USERS (un manual puede tener muchas solicitudes)

## 12. Glosario

- API: Application Programming Interface, conjunto de endpoints REST del backend.

- Backend: Lógica y persistencia del sistema.
- CODE\_VERIFICATION: tabla que almacena códigos enviados a usuarios para verificación o recuperación.
- MANUALS\_VS\_USERS: tabla intermedia que registra solicitudes de usuarios para acceder a manuales.
- sColmoticaService: servicio interno que gestiona hashing y validaciones de seguridad.
- sMailService: servicio responsable del envío de correos y verificación de códigos.