**Ecoluz – Especificación de Requisitos de Software (ERS)**

**Proyecto:** Sistema de Monitoreo y Gestión de Luminarias Inteligentes  
**Acrónimo:** *Ecoluz*  
**Fecha:** 10/08/2025  
**Equipo:** Luis Iván Márquez Azuara; Brayn Kalid Reyes Silva; Aldo Tolentino Domingo; Ángel David Reyes Téllez

**1. Introducción**

**1.1 Propósito**

Este documento especifica, de forma completa y verificable, los requisitos del software para *Ecoluz*. Sirve como contrato técnico entre el equipo desarrollador y las partes interesadas académicas. A partir de esta ERS se deberán planear, construir, probar y aceptar las funcionalidades del sistema.

**1.2 Alcance**

*Ecoluz* es una plataforma para monitoreo de luminarias públicas, compuesta por:

* **Aplicación web administrativa** (consulta, gestión).
* **API REST** (Node.js) que integra datos provenientes de sensores en luminarias y expone servicios a los clientes.
* **App para wearable (Wear OS)** de sólo consulta para técnicos de campo (lectura de estado y ubicación).
* **Base de datos NoSQL (MongoDB)** para persistencia.

El sistema permite: visualización en mapa, panel de alertas, gestión de usuarios/roles y estadísticas de consumo.

**1.3 Definiciones, acrónimos y abreviaturas**

* **API**: Application Programming Interface.
* **IoT**: Internet of Things.
* **JWT**: JSON Web Token (autenticación).
* **TLS**: Transport Layer Security.
* **RNF**: Requisito No Funcional.
* **RF**: Requisito Funcional.
* **Wear OS**: Sistema operativo de smartwatches de Google.

**1.4 Referencias**

* Documento de Requisitos del Sistema (Ecoluz)
* Modelo del Sistema (Ecoluz), 2025.
* IEEE 830/29148 – Guía de especificación de requisitos (referencia de estructura).

**1.5 Visión general del documento**

La sección 2 describe el producto en su contexto. La sección 3 establece los requisitos funcionales numerados. La sección 4 consolida requisitos no funcionales. La sección 5 define datos e interfaces. La sección 6 lista casos de uso resumidos. La sección 7 incluye trazabilidad, criterios de aceptación y validación. La sección 8 presenta anexos técnicos.

**2. Descripción general**

**2.1 Perspectiva del producto**

*Ecoluz* se ubica en una arquitectura cliente-servidor: sensores de luminarias reportan estado y métricas a la **API REST (Node.js)**, los datos se almacenan en **MongoDB** y se consumen desde la **web** y el **wearable**.

**Componentes y relaciones (alto nivel):**

1. Sensores en luminarias → API REST (ingesta).
2. API REST ↔ MongoDB (persistencia/consulta).
3. Web Admin ↔ API REST (gestión, mapas).
4. Wear OS ↔ API REST (consulta cercana por ubicación).

**2.2 Funciones del sistema (resumen)**

* Autenticación/autorización con roles.
* Gestión de luminarias y sensores asociados.
* Visualización geográfica (mapa) y estado operativo.
* Recepción de datos y detección de fallas/alertas.
* Consulta en wearable (sólo lectura).

**2.3 Características de usuario**

* **Administrador (Gestor municipal):** configuración, gestión total.
* **Operador:** monitoreo, consultas.
* **Técnico de campo:** consulta de estado/ubicación desde wearable.

**2.4 Restricciones y supuestos**

* Conectividad a Internet requerida para todos los clientes.
* Navegadores modernos (Chrome, Firefox, Edge) soportados.
* Los sensores reportan vía **HTTP/REST** (mínimo).
* El wearable es **sólo lectura** (no envía cambios).

**2.5 Suposiciones y dependencias**

* Disponibilidad de credenciales de mapa (por ejemplo, Leaflet con proveedor de tiles público).
* Permisos municipales para geolocalización y mantenimiento de inventario.
* Políticas de respaldo/retención definidas por el administrador institucional.

**3. Requisitos funcionales (RF)**

**Nota:** Todos los RF deben ser trazables a casos de uso, tener criterios de aceptación y prioridad.

**RF-01 Autenticación y sesiones (JWT).**

* El sistema debe permitir iniciar sesión y emitir **JWT** firmados.
* Criterio: Un usuario válido recibe un JWT y puede acceder a endpoints protegidos.

**RF-02 Gestión de usuarios y roles.**

* Crear/editar/eliminar usuarios; asignar roles: *Administrador, Operador, Técnico*.
* Criterio: Accesos diferenciados por rol en UI y API.

**RF-03 Registro y gestión de luminarias.**

* Alta/edición/baja de luminarias con: código, ubicación (lat/lon), estado, consumo estimado, identificador de sensor.
* Criterio: CRUD operativo y validaciones (coordenadas válidas, campos obligatorios).

**RF-04 Asociación de sensores.**

* Asociar un sensor IoT a cada luminaria con su identificador único.
* Criterio: El sistema rechaza asociaciones duplicadas/inválidas.

**RF-05 Ingesta de datos de sensores (HTTP/REST mínimo).**

* Recepción de mediciones (estado ON/OFF, consumo, errores) y almacenamiento en MongoDB.
* Criterio: Datos persistidos y consultables por rango de tiempo.

**RF-06 Visualización en mapa.**

* Mostrar luminarias en mapa interactivo con estado (colores/íconos).
* Criterio: Al menos *zoom, clic* con detalle de cada luminaria.

**RF-07 Panel de alertas por fallas/anomalías.**

* Generación y listado de alertas con severidad (alta/media/baja).
* Criterio: Al dispararse una falla, aparece en el panel con metadatos.

**RF-08 Historial de mantenimiento.**

* Registro de intervenciones (fecha, tipo, técnico, notas) por luminaria.
* Criterio: Consulta filtrada por fecha y exportable.

**RF-09 Filtros y búsquedas.**

* Filtrar por fecha, zona, estado, ID de luminaria/sensor.
* Criterio: Respuesta con paginación y ordenamiento.

**RF-11 Consulta de estado operativo.**

* Mostrar si la luminaria está encendida, apagada o con falla.
* Criterio: Actualización reflejada en mapa y ficha de detalle.

**RF-12 Wearable – consulta cercana.**

* App en Wear OS que **sólo lee** y muestra luminarias cercanas y su estado.

**RF-18 API de consulta pública limitada (opcional).**

* Endpoint para exponer estadísticas agregadas sin datos sensibles.
* Criterio: Sin identificación de activos individuales.

**RF-19 Notificaciones (futuro/optativo).**

* Canal para enviar avisos (email/push) ante alertas críticas.
* Criterio: Configurable por rol/zona. (Marcado como *futuro*.)

**RF-20 Backup/restore operativo.**

* Programar respaldos y restauración controlada.
* Criterio: Al menos respaldo semanal disponible.

**4. Requisitos no funcionales (RNF)**

**Rendimiento y escalabilidad**

* **RNF-01** Tiempo de respuesta API < **500 ms** en consultas estándar.
* **RNF-02** Dashboard carga total < **2 s** en red estable.
* **RNF-03** Concurrencia mínima: **10,000** sensores conectados (ingesta/consulta).
* **RNF-04** Escalabilidad horizontal del backend (balanceo/replicas).

**Disponibilidad y confiabilidad**

* **RNF-05** Disponibilidad mensual ≥ **99%**.
* **RNF-06** Tolerancia a fallos: caídas de módulos no afectan el sistema completo.
* **RNF-07** Respaldos automáticos semanales; restauración documentada.

**Seguridad**

* **RNF-08** Comunicación cifrada con **TLS 1.3**.
* **RNF-09** Control de acceso por roles/permisos.
* **RNF-10** Prácticas OWASP (validaciones, sanitización, protección CSRF/XSS).
* **RNF-11** Logs de actividad para auditoría (acciones, fecha/hora, usuario, IP).

**Compatibilidad y usabilidad**

* **RNF-12** Web compatible con últimas versiones de Chrome/Firefox/Edge.
* **RNF-13** Android **10+** (si aplica app móvil).
* **RNF-14** Wear OS **3+**.
* **RNF-15** UI responsiva y accesible (WCAG AA).

**Mantenibilidad y proceso**

* **RNF-16** Integración continua (GitHub Actions); pruebas automáticas.
* **RNF-17** Documentación técnica viva (por ejemplo, en Notion).
* **RNF-18** Versionado SemVer y convenciones de *commits*.

1. **Datos, interfaces y arquitectura**

**5.1 Modelo de datos**

**Usuario**

**Colección:** usuario  
**Campos (según respuestas reales de la API):**

* \_id: ObjectId
* identificador: string (ej. ADM-018)
* nombre: string
* apellido: string
* telefono: string
* rol: string (admin, supervisor, usuario)
* correo: string | null
* createdAt / updatedAt:

**Notas:**

* Las **credenciales** viven en otra colección (autenticación) y se crean con POST /api/usuarios/completo.
* La API de **login** responde un **JWT** (no se expone hash en Usuario).

**Luminaria**

**Colección:** luminaria  
**Campos**

* \_id: ObjectId
* identificador: string (ej. LUM-001-MTY)
* tipo\_luminaria: string (LED, etc.)
* pais: string
* estado: string
* ciudad: string
* region: string
* coordenadas: { lat: number, lng: number }
* fecha\_instalacion: ISODate
* activo: boolean

**Consumo (registro de sensores)**

**Colección:** consumo  
**Campos**

* \_id: ObjectId
* luminaria\_id: ObjectId
* timestamp: ISODate
* consumo: number
* lumenes: number
* encendida: boolean

**Autenticación (credenciales)**

*(colección separada, no se expone en respuestas públicas)*

* \_id, usuario\_id, correo, password\_hash, activo, createdAt, updatedAt

**5.2 Especificación de API REST**

**Autenticación**

* **POST** /api/auth/login  
  **Body:** { "correo": string, "password": string }  
  **200:** { "token": "JWT", ...claims }  
  Requerido para endpoints protegidos (ej. GET /api/usuarios).

**Usuarios**

* **POST** /api/usuarios/completo  
  Crea **usuario + credenciales** en una sola operación.  
  **201:** { success, message, usuario\_id, auth\_id }
* **GET** /api/usuarios?incluirInactivos=false  
  Lista usuarios (protegidA con JWT).  
  **200:** { success, usuarios: [...], total }
* **GET** /api/usuarios/{id}  
  Devuelve un usuario por \_id.  
  **200:** { success, usuario } • **404** si no existe
* **GET** /api/usuarios/identificador/{identificador}  
  Devuelve un usuario por identificador.  
  **200:** { ...usuario } • **404** si no existe
* **GET** /api/usuarios/rol/{rol}  
  Devuelve usuarios por rol (admin|supervisor|usuario).  
  **200:** [ ...usuarios ]
* **PUT** /api/usuarios/{id}  
  Actualiza datos del usuario.  
  **200:** { success, message, usuario: {...} } • **404** si no existe
* **DELETE** /api/usuarios/{id}  
  Baja (lógica o definitiva, según implementación).  
  **200:** { success, message, usuarioEliminado: {...} } • **404** si no existe

**Consumo**

* **GET** /api/consumo?limite=100  
  Lista registros (por defecto limite=100).  
  **200:** [ { \_id, luminaria\_id, timestamp, consumo, lumenes, encendida } ]
* **POST** /api/consumo  
  Inserta **un** registro de sensor.  
  **201:** { message: "Registro de sensor creado exitosamente", id }  
  **400** datos inválidos.
* **GET** /api/consumo/{id}  
  Devuelve un registro por \_id.  
  **200:** { \_id, luminaria\_id, timestamp, consumo, lumenes, encendida }  
  **404** si no existe.
* **POST** /api/consumo/bulk  
  Inserción **masiva** de registros.  
  **201:** { message: "Registros creados exitosamente", insertados, ids: [...] }  
  **400** si el body no es válido.
* **GET** /api/consumo/estadisticas/{luminaria\_id}?fecha\_inicio=ISO&fecha\_fin=ISO  
  Estadísticas agregadas de consumo.  
  **200:** { \_id: luminaria\_id, consumoTotal, consumoPromedio, lumenesTotal, lumenesPromedio, tiempoEncendida, totalRegistros, consumoMaximo, consumoMinimo }  
  **404** sin datos.
* **DELETE** /api/consumo/limpieza/antiguos?fecha\_limite=ISO  
  Limpieza de datos previos a fecha\_limite.  
  **200:** { message: "Registros antiguos eliminados exitosamente", eliminados }  
  **400** si falta fecha\_limite.

**5.3 Formatos de datos (ejemplos)**

**5.3.1 Login**

**POST** /api/auth/login

{

"correo": "admin@eurbana.com",

"password": "Admin123!"

}

**200**

{

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6..."

}

**5.3.2 Crear usuario completo**

**POST** /api/usuarios/completo

{

"identificador": "ADM-018",

"nombre": "Angel",

"apellido": "Reyes",

"telefono": "+52-81-5555-1235",

"rol": "admin",

"correo": "angel.lopez@eurbana.com",

"password": "Admin123!"

}

**201**

{

"success": true,

"message": "Usuario y credenciales creados exitosamente",

"usuario\_id": "68a216b78442c2285baa94d0",

"auth\_id": "68a216b78442c2285baa94d1"

}

**5.3.3 Listar usuarios (JWT)**

**GET** /api/usuarios?incluirInactivos=false  
**Header:** Authorization: Bearer <JWT>  
**200**

{

"success": true,

"usuarios": [

{

"\_id": "68a216b78442c2285baa94d0",

"identificador": "ADM-018",

"nombre": "Angel",

"apellido": "Reyes",

"telefono": "+52-81-5555-1235",

"correo": "admin"

}

],

"total": 5

}

**5.3.4 Registro de sensor (individual)**

**POST** /api/consumo

{

"luminaria\_id": "507f1f77bcf86cd799439012",

"timestamp": "2024-01-15T20:30:00.000Z",

"consumo": 85.5,

"lumenes": 3200,

"encendida": true

}

**201**

{

"message": "Registro de sensor creado exitosamente",

"id": "507f1f77bcf86cd799439011"

}

**5.3.5 Registro de sensor (bulk)**

**POST** /api/consumo/bulk

{

"registros": [

{

"luminaria\_id": "507f1f77bcf86cd799439012",

"timestamp": "2024-01-15T20:30:00.000Z",

"consumo": 85.5,

"lumenes": 3200,

"encendida": true

},

{

"luminaria\_id": "507f1f77bcf86cd799439012",

"timestamp": "2024-01-15T20:29:00.000Z",

"consumo": 82.1,

"lumenes": 3150,

"encendida": true

}

]

}

**201**

{

"message": "Registros creados exitosamente",

"insertados": 25,

"ids": ["68a2...", "68a3..."]

}

**5.3.6 Estadísticas de consumo**

**GET** /api/consumo/estadisticas/507f1f77bcf86cd799439012?fecha\_inicio=2024-01-01T00:00:00Z&fecha\_fin=2024-01-31T23:59:59Z  
**200**

{

"\_id": "507f1f77bcf86cd799439012",

"consumoTotal": 2560.5,

"consumoPromedio": 85.35,

"lumenesTotal": 96000,

"lumenesPromedio": 3200,

"tiempoEncendida": 25,

"totalRegistros": 30,

"consumoMaximo": 95.2,

"consumoMinimo": 0

}

**5.3.7 Limpieza de registros antiguos**

**DELETE** /api/consumo/limpieza/antiguos?fecha\_limite=2024-01-01T00:00:00Z  
**200**

{

"message": "Registros antiguos eliminados exitosamente",

"eliminados": 1250

}

**5.4 Reglas de negocio**

* Severidad **alta** si *estado = FALLA* o *consumo > umbralZona*.
* Una luminaria **no puede** tener más de un sensor activo asociado.
* Sólo **admin** crea/borra usuarios; **operador** no elimina activos.

**5.5 Arquitectura y tecnologías**

* **Backend:** Node.js (JavaScript), framework web (por ejemplo, Express).
* **Base de datos:** MongoDB (colecciones: luminarias, sensores, alertas, mantenimientos, usuarios).
* **Frontend web:** React
* **Wearable:** Wear OS (Kotlin), **sólo lectura**

**6. Casos de uso (resumen)**

**CU-01 Iniciar sesión** – Un usuario válido accede y obtiene JWT.  
**Pre:** Usuario registrado. **Post:** Sesión activa. **Excepciones:** 401 credenciales inválidas.

**CU-02 Alta de luminaria** – Admin registra una luminaria con ubicación y sensor.  
**Pre:** Sensor disponible. **Post:** Luminaria visible en mapa. **Ex:** 400 datos inválidos; 409 sensor ya asociado.

**CU-03 Monitorear mapa** – Operador visualiza luminarias y estados.  
**Pre:** Datos recientes. **Post:** Estado actualizado. **Ex:** 503 datos no disponibles.

**CU-04 Gestionar alerta** – Operador revisa y marca atendida/cerrada.  
**Pre:** Alerta activa. **Post:** Cambio registrado. **Ex:** 403 sin permisos.

**CU-05 Registrar mantenimiento** – Técnico carga intervención con notas/evidencias.  
**Pre:** Orden de trabajo. **Post:** Historial actualizado. **Ex:** 400 campos faltantes.

**CU-06 Consultar desde wearable** – Técnico ve luminarias cercanas.  
**Pre:** GPS activo y red. **Post:** Lista ordenada por proximidad. **Ex:** 408 tiempo de espera.

**7. Trazabilidad, aceptación y validación**

**7.1 Matriz de trazabilidad (extracto)**

| **Objetivo** | **RF** | **RNF** |
| --- | --- | --- |
| Monitoreo continuo | RF-05, RF-06, RF-11 | RNF-01, RNF-05 |
| Gestión operativa | RF-03, RF-08, RF-10 | RNF-12, RNF-15 |
| Seguridad | RF-01, RF-02, RF-14 | RNF-08, RNF-09, RNF-10 |
| Movilidad técnica | RF-12 | RNF-14 |

**9. Anexos**

**9.1 Catálogos**

* **Severidad de alerta:** alta, media, baja.
* **Tipos de mantenimiento:** preventivo, correctivo, sustitución, inspección.

**9.2 Códigos de error (API)**

* **400** Datos inválidos
* **401** No autenticado
* **403** Prohibido
* **404** No encontrado
* **409** Conflicto (duplicado/asociación existente)
* **500** Error interno