

Requerimientos No Funcionales

RNF-01: Rendimiento

RNF-01.1 El sistema debe responder a búsquedas de validación en menos de 2 segundos

- Búsqueda por placas, folio, CURP o escaneo de QR
- Medido desde envío de request hasta recepción de respuesta

RNF-01.2 El registro de entrada/salida debe completarse en menos de 5 segundos

- Tiempo total desde búsqueda hasta confirmación guardada en BD

RNF-01.3 La generación de pases (PDF + QR + envío de correo) debe completarse en menos de 15 segundos

- Generación asíncrona para no bloquear UI

RNF-01.4 El sistema debe soportar al menos 50 usuarios concurrentes sin degradación

RNF-01.5 La carga inicial de cualquier página no debe exceder 3 segundos

- Medido con conexión de red estándar (10 Mbps)
- First Contentful Paint (FCP) < 1.5s
- Time to Interactive (TTI) < 3s

RNF-01.6 El mapa de cajones de motos debe actualizar su visualización en menos de 500ms

- Tras cualquier cambio de estado de cajón
- Uso de WebSocket preferido para actualizaciones en tiempo real

RNF-01.7 Las consultas a la base de datos deben estar optimizadas

- Uso de índices en campos de búsqueda frecuente (placas, CURP, folio)
- Queries complejas deben ejecutarse en menos de 1 segundo

RNF-01.8 Las imágenes subidas deben comprimirse automáticamente

- Sin pérdida significativa de calidad
- Tamaño reducido en al menos 40% para optimizar almacenamiento y carga

RNF-02: Seguridad

RNF-02.1 Las contraseñas deben almacenarse encriptadas con bcrypt

- Factor de costo (rounds): 12
- Salt generado automáticamente por usuario

RNF-02.2 El sistema debe implementar autenticación basada en JWT

- Tokens firmados con algoritmo HS256 o RS256
- Payload incluye: id_usuario, rol, tipo_usuario, iat, exp

- Vigencia del token: 24 horas
- Refresh token con vigencia de 7 días (opcional)

RNF-02.3 El sistema debe implementar protección contra inyección SQL (No es tan necesario)

- Uso obligatorio de ORM (Sequelize) con queries parametrizadas
- Validación de inputs en backend
- Nunca concatenar strings para formar queries

RNF-02.4 El sistema debe implementar validación de datos en cliente y servidor

- Doble validación para prevenir manipulación de requests
- Backend como última línea de defensa (nunca confiar solo en frontend)

RNF-02.5 Los folios de pases deben ser únicos y no predecibles

- Generación con componente aleatorio criptográficamente seguro
- Verificación de unicidad antes de almacenar en BD

RNF-02.6 El sistema debe cerrar sesiones automáticamente después de 30 minutos de inactividad

- Implementado mediante expiración de token JWT
- Renovación automática al detectar actividad (refresh token)

RNF-02.7 El sistema debe registrar y limitar intentos fallidos de inicio de sesión

- Máximo 5 intentos fallidos por cuenta
- Bloqueo temporal de 15 minutos tras exceder el límite
- Registro en log de auditoría con nivel WARNING
- Notificación por correo al usuario sobre intentos sospechosos

RNF-02.8 Los archivos subidos deben validarse estrictamente

- Validación de tipo MIME real (no solo extensión)
- Tipos permitidos: image/jpeg, image/png
- Tamaño máximo: 5 MB por archivo
- Escaneo antivirus recomendado en producción

RNF-02.9 El acceso a funciones administrativas debe requerir rol de admin/guardia

- Middleware de autorización en cada ruta protegida
- Verificación de rol en el token JWT
- Respuesta 403 Forbidden si no tiene permisos

RNF-03: Usabilidad

RNF-03.1 La interfaz debe ser intuitiva y no requerir más de 10 minutos de capacitación

- Para usuarios finales (registro, generación de pases, envío de reportes)
- Tooltips y ayuda contextual disponibles

RNF-03.2 La interfaz del administrador/guardia debe ser eficiente para uso intensivo

- Accesos directos con teclado (keyboard shortcuts)
- Flujo optimizado para tareas repetitivas (registrar entrada/salida)
- Botones grandes y claros con feedback visual inmediato

RNF-03.3 El sistema debe ser completamente responsive

- Breakpoints: Mobile (320px+), Tablet (768px+), Desktop (1024px+)
- Diseño Mobile-First
- Touch-friendly en dispositivos táctiles (botones >44px)

RNF-03.4 Los mensajes de error deben ser claros y orientar al usuario

- Formato: "Qué salió mal + Cómo corregirlo"
- Ejemplo: "Las placas ABC-123 ya están registradas. Verifica e intenta con otras placas."
- Evitar mensajes técnicos (códigos de error internos ocultos al usuario)

RNF-03.5 El sistema debe proporcionar retroalimentación visual inmediata

- Loaders/spinners durante procesos (>500ms)
- Toasts/notificaciones para confirmaciones (ej: "Pase generado exitosamente")
- Animaciones sutiles para transiciones (max 300ms)
- Estados de botones: Normal, Hover, Active, Disabled

RNF-03.7 La interfaz debe usar terminología familiar para usuarios mexicanos

- "Placas" en lugar de "matrícula"
- "Boleta" para estudiantes, "Número de empleado" o "RFC" para personal
- "INE" o "Credencial de elector" en lugar de "ID"
- "CURP" (término oficial mexicano)

RNF-04: Disponibilidad y Confiabilidad

RNF-04.1 El sistema debe manejar errores de forma elegante

- Sin exponer información técnica sensible al usuario (stack traces, DB errors)
- Mensajes user-friendly en frontend
- Errores técnicos registrados en logs del servidor
- Páginas de error personalizadas (404, 500, 503)

RNF-04.2 El sistema debe implementar logging

- Niveles: DEBUG, INFO, WARNING, ERROR, CRITICAL
- Logs estructurados (JSON format)
- Rotación automática de archivos de log (máximo 50 MB por archivo)
- Retención de logs: 90 días

RNF-05: Mantenibilidad

RNF-05.1 El código backend debe seguir estándares de JavaScript/Node.js

- Estilo: ESLint con configuración Airbnb o Standard
- Uso de ES6+ features (arrow functions, async/await, destructuring)
- Comentarios JSDoc en funciones complejas

RNF-05.2 El código frontend debe seguir estándares de React

- Hooks preferidos sobre class components
- Componentes funcionales puros cuando sea posible
- PropTypes o TypeScript para type checking
- ESLint con plugin de React

RNF-05.3 El sistema debe seguir arquitectura modular y escalable

- Backend: Separación clara entre rutas, controladores, servicios, modelos
- Frontend: Componentes reutilizables, hooks personalizados, context para estado global
- Principio de responsabilidad única (Single Responsibility Principle)

RNF-05.4 Debe existir documentación técnica completa

- README principal con:
 - Descripción del proyecto
 - Instrucciones de instalación
 - Variables de entorno necesarias
 - Scripts disponibles
 - Estructura del proyecto
- README por carpeta (frontend/backend) con detalles específicos
- Documentación de API
- Diagramas de arquitectura y base de datos

RNF-06: Portabilidad

RNF-06.1 El frontend debe funcionar correctamente en dispositivos móviles

RNF-06.2 El sistema debe soportar variables de entorno para configuración

- Uso de archivo .env para desarrollo
- Variables de entorno del sistema operativo para producción
- Nunca hardcodear credenciales o URLs en código (solo cuando aún no se conecta back y front)