

Documento de Requisitos Funcionales (RF)

1. Definición

Los **Requisitos Funcionales (RF)** definen las funcionalidades y el comportamiento que el sistema debe ejecutar. Cada requisito ha sido catalogado basándose en los **Endpoints de la API** y la interfaz de usuario implementada, asegurando la trazabilidad.

2. Requisitos Funcionales por Categoría

2.1. Categoría 1: Autenticación y Acceso

RF ID	Descripción Detallada	Referencia de Implementación	Actor Principal
RF-01	Registro de Nuevo Usuario: Permitir el registro de nuevos usuarios, asignando por defecto el rol de paciente .	POST /api/auth/register ^{1/} Lógica en backend/src/controllers/authController.js ² .	Todos
RF-02	Inicio de Sesión Seguro: Autenticar al usuario mediante email/contraseña, verificando credenciales y generando un token JWT .	POST /api/auth/login ^{3/} Route: backend/src/routes/auth.routes.js ⁴ .	Todos
RF-03	Obtención de Perfil de Sesión: Obtener los datos del usuario autenticado (Nombre, Rol, Email) para la personalización de la interfaz.	GET /api/auth/me ⁵ . Usado por authAPI.getMe() en web/js/api.js ⁶ .	Todos
RF-04	Redirección por Rol y Protección: Redirigir al <i>Dashboard</i> correspondiente y proteger todas las rutas privadas mediante verificación de token y rol.	Función redirectToDashboard() y protectPage() en web/js/api.js ⁷ .	Todos
RF-05	Restablecimiento de Contraseña: Permitir al usuario solicitar un restablecimiento de contraseña mediante email o tarjeta sanitaria.	Interfaz implementada en web/password_recovery.html.	Todos

2.2. Categoría 2: Gestión de Citas y Disponibilidad

RF ID	Descripción Detallada	Referencia de Implementación	Actor Principal
RF-06	Búsqueda de Disponibilidad: Permitir la búsqueda por Especialidad, Centro Médico y Médico para la reserva.	Interfaz en web/book_new_appointment.html. Se apoya en: GET /api/centers ⁸ .	Paciente
RF-07	Listado de Médicos Filtrado: Listar <i>Médicos</i> disponibles mediante filtro de especialidad.	GET /api/users/doctors?specialty=X ⁹ . Función: usersAPI.listDoctors en web/js/api.js ¹⁰ .	Paciente
RF-08	Creación de Cita: Crear una nueva cita, registrando todos los datos necesarios (profesional, centro, fecha, hora).	POST /api/appointments ¹¹ . Función: appointmentsAPI.create en web/js/api.js ¹² .	Paciente
RF-09	Listado de Citas (Paciente): Listar todas las citas asociadas al <i>Paciente</i> autenticado.	GET /api/appointments ¹³ (filtrado por <i>patientId</i>). Lógica <i>frontend</i> en web/js/modules/patient-dashboard.js.	Paciente
RF-10	Cancelación de Cita (Paciente): Permitir al <i>Paciente</i> eliminar una cita pendiente desde su <i>Dashboard</i> .	DELETE /api/appointments/:id ¹⁴ . Función: appointmentsAPI.cancel en web/js/api.js ¹⁵ .	Paciente
RF-11	Cierre de Cita (Médico): Permitir al <i>Médico</i> cambiar el estado de una cita a completada .	PATCH /api/appointments/:id/status ¹⁶ . Lógica en web/js/modules/doctor-dashboard.js.	Médico

2.3. Categoría 3: Administración y Mantenimiento (CRUD)

RF ID	Descripción Detallada	Referencia de Implementación	Actor Principal
RF-12	Listado de Usuarios (Admin): Mostrar una tabla con todos los usuarios del sistema para su gestión.	GET /api/users ¹⁷ . Módulo: web/js/modules/administrator-dashboard.js.	Administrador
RF-13	Creación de Usuarios (Admin): Permitir crear nuevos usuarios con asignación de roles y especialidades .	POST /api/users ¹⁸ . Lógica en web/js/modules/administrator-dashboard.js.	Administrador
RF-14	Edición de Usuarios (Admin): Modificar los datos de cualquier usuario (nombre, email, rol, etc.).	PUT /api/users/:id ¹⁹ . Función: usersAPI.update en web/js/api.js ²⁰ .	Administrador
RF-15	Eliminación de Usuarios (Admin): Eliminar usuarios del sistema.	DELETE /api/users/:id ²¹ . Función: usersAPI.delete en web/js/api.js ²² .	Administrador
RF-16	CRUD de Centros (Admin): Crear, Editar y Eliminar centros médicos.	POST, PUT, DELETE /api/centers/:id ²³ .	Administrador
RF-17	Toggle de Estado (Centros): Cambiar el estado de un <i>Centro Médico</i> a <i>Activo</i> o <i>Inactivo</i> .	PATCH /api/centers/:id/status ²⁴ . Función: centersAPI.toggleStatus en web/js/api.js ²⁵ .	Administrador
RF-18	Búsqueda en Tablas (Frontend): Las tablas de <i>Usuarios</i> y <i>Centros</i> deben permitir la búsqueda en tiempo real por nombre, email o rol.	Lógica de filtrado y renderizado en web/js/modules/administrator-dashboard.js.	Administrador

2.4. Categoría 4: Auxiliares y Experiencia de Usuario (RF-19 a RF-25)

RF ID	Descripción Detallada	Referencia de Implementación	Actor Principal
RF-19	Gestión de Órdenes Médicas: El <i>Médico</i> debe poder crear y listar órdenes médicas para sus pacientes.	Interfaz implementada en web/doctor_dashboard.html (Panel "Órdenes Médicas").	Médico
RF-20	Mensajería Interna (Chat): Proporcionar una interfaz de chat para la comunicación <i>Paciente-Médico</i> .	Interfaz en web/_chat.html y web/doctor_dashboard.html.	Paciente, Médico
RF-21	Centro de Notificaciones: Mostrar un centro de notificaciones con información sobre citas, recordatorios y alertas.	Interfaz implementada en web/notification_center.html.	Todos
RF-22	Gestión de Pagos Online: Procesar los pagos online de las citas mediante diversos métodos (Tarjeta, PayPal, Transferencia).	Interfaz implementada en web/online_payment_screen.html.	Paciente
RF-23	Cálculo de Pagos Detallado: Mostrar el resumen de pagos, incluyendo subtotal, IVA (21%), y la aplicación de descuentos por códigos y puntos de fidelidad.	Lógica de cálculo y descuentos en web/online_payment_screen.html.	Paciente
RF-24	Sistema de Puntos y Recompensas: Mostrar el nivel de fidelidad y puntos disponibles del <i>Paciente</i> para canje por descuentos.	Lógica <i>frontend</i> en web/js/modules/patient-dashboard.js.	Paciente
RF-25	Panel de Métricas (Analytics): Acceso a un panel de <i>Analytics</i> que muestre estadísticas clave del sistema.	Interfaz implementada en web/healthcare_analytics_dashboard.html.	Administrador