Sistema de Ingreso y Control de Pacientes:

MediCor –Entrega de Final

Por:

Sarah Montes Jarava

Docente:

Leidy Johanna Gonzales Ballesteros

Universidad Cooperativa de Colombia

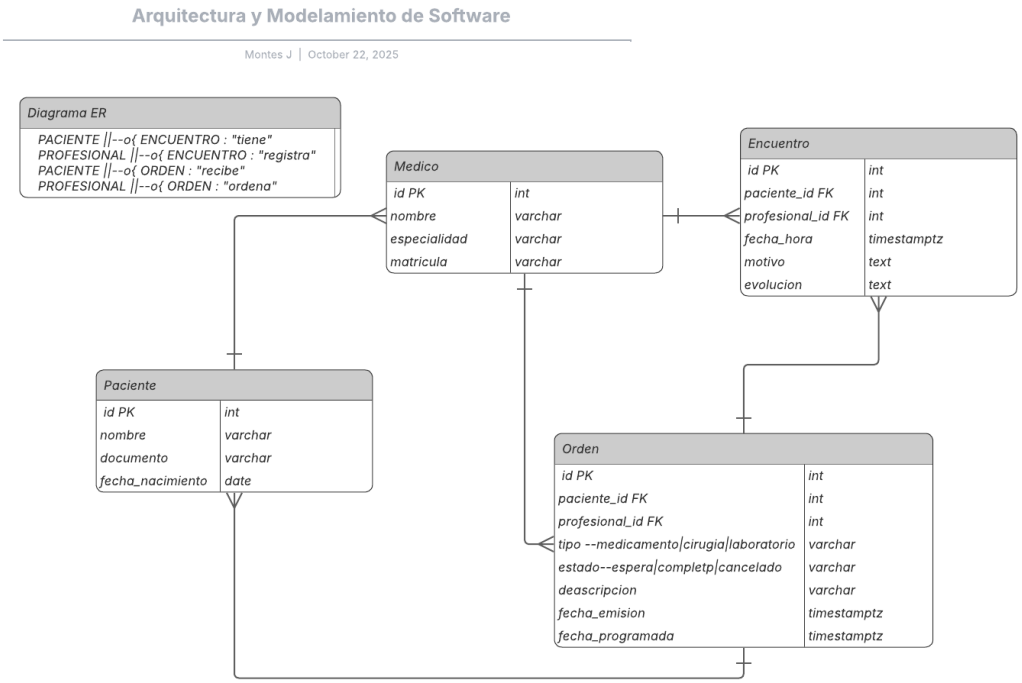
Arquitectura y Modelamiento de Software

Montería, Córdoba

2025, Octubre

1. Modelo de dominio y datos

1.1. Diagrama ER – Conceptual



1.2. Diccionario de datos: entidad, atributo, tipo, restricción.

Diccionario de Datos

PACIENTE

- **id — bigserial —** PK **—** autoincrement
- **nombre — varchar(150) —** NOT NULL
- **documento — varchar(50) —** NOT NULL, UNIQUE (nro cédula)
- **fecha_nacimiento — date —** NULLABLE
- **telefono — varchar(30) —** NULLABLE
- **created_at — timestamptz —** default now()

PROFESIONAL

- **id — bigserial —** PK
- **nombre — varchar(150) —** NOT NULL
- **especialidad — varchar(100) —** NULLABLE
- **matricula — varchar(80) —** UNIQUE, NOT NULL
- **created_at — timestamptz —** default now()

ENCUENTRO (evolución / nota clínica)

- **id — bigserial —** PK
- **paciente_id — bigint —** FK → paciente(id) **—** NOT NULL
- **profesional_id — bigint — FK** → profesional(id) — NOT NULL
- **fecha_hora — timestamptz —** NOT NULL
- **motivo — text —** motivo de consulta/ingreso
- **evolucion — text —** registro clínico (examen, plan, observaciones)
- **created_at — timestamptz —** default now()

ORDEN

- **id — bigserial —** PK
- **paciente_id — bigint —** FK → paciente(id) **—** NOT NULL
- **profesional_id — bigint —** FK → profesional(id) **—** NOT NULL
- **tipo — varchar(20) —** NOT NULL (values: medication,surgery,lab)
- **descripcion — text —** NOT NULL (p. ej. pauta farmacológica o detalle de cirugía)
- **estado — varchar(20) —** NOT NULL default pending (values: pending,executed,cancelled)
- **fecha_emision — timestamptz —** NOT NULL default now()
- **fecha_programada — timestamptz —** NULLABLE (Para laboratorios y cirugías programadas)
- **created_at — timestamptz —** default now()

1.3. Descripción de relaciones (1–N, N–M).

Las relaciones están incluidas en el diagrama ER y en las claves foráneas (FK), para esta entrega se redactarán en texto, con una breve descripción 1–N o N–M como se indicó en la guía de entrega de trabajo.

1.3.1. Descripción de relaciones entre entidades

1. Paciente – Encuentro (1–N)

Un paciente puede tener muchos encuentros (consultas o evoluciones clínicas).

Cada encuentro pertenece a un solo y único paciente.

Implementación: *encuentro.paciente_id → FK a paciente.id.*

Ejemplo: El paciente “Samuel Rodríguez” tiene 3 encuentros registrados (consulta inicial-ingreso, control, postoperatorio).

2. Profesional – Encuentro (1–N)

Un profesional de salud puede registrar muchos encuentros (evoluciones de distintos pacientes).

Cada encuentro es atendido por un único profesional.

Implementación: encuentro.profesional_id → FK a profesional.id.

Ejemplo: La doctora “Mariana López” atendió 10 encuentros diferentes.

3. Paciente – Orden (1–N)

Un paciente puede tener varias órdenes médicas (medicamentos, cirugías, laboratorios).

Cada orden pertenece a un solo y único paciente.

Implementación: orden.paciente_id → FK a paciente.id.

Ejemplo: El paciente “Samuel Rodríguez” tiene dos órdenes: una de medicación y una de cirugía programada.

4. Profesional – Orden (1–N)

Un profesional puede emitir múltiples órdenes (para distintos pacientes).

Cada orden es emitida por un único profesional.

Implementación: *orden.profesional_id → FK a profesional.id.*

Ejemplo: El doctor “Juan Pérez” ha emitido 25 órdenes de medicamentos y 3 cirugías.

5. Relaciones N–M indirectas (implícitas)

Un profesional puede ordenar tratamientos o cirugías a varios pacientes → N–M representada mediante orden.

2. Diseño de API (Contratos)

Autenticación: Authorization: Bearer <JWT> en endpoints protegidos.

Fechas: Formato ISO8601 con zona (p. ej. 2025-11-05T09:00:00-05:00). Se almacena en DB como timestamptz (UTC).

Recurso / Acción	Método	Ruta	Body / Query	Respuesta (200/201)
Registro usuario (paciente/prof)	POST	/api/v1/auth/register	body: `{ name, email, password, role: "patient"	"doctor" }`
Login	POST	/api/v1/auth/login	body: { email, password }	200 { access_token, token_type:"Bearer", expires_in, user: {id,name,email,role} }
Obtener perfil	GET	/api/v1/auth/me	header Auth	200 { id,name,email,role,meta ... }
Lista pacientes (filtros)	GET	/api/v1/patients	query: ?q=&page=&per_page=&age_min=&age_max=	200 { items:[{id,name,age,email}], total, page, per_page }
Crear paciente	POST	/api/v1/patients	body: { name, dob, phone, email, gender, address }	201 { id, name, dob, ... }
Obtener paciente	GET	/api/v1/patients/{patient_id}	path param	200 { id,name,dob,contacts,medical_history:[] }
Actualizar paciente	PUT / PATCH	/api/v1/patients/{patient_id}	body: fields editables	200 { id,... }
Eliminar paciente	DELETE	/api/v1/patients/{patient_id}	auth role=admin/doctor	204 No Content
Lista doctores	GET	/api/v1/doctors	query: ?specialty=&q=	200 { items:[{id,name,specialty,available}], ... }

Crear cita (appointment)	**POST**	**/api/v1/appointments**	**body: { patient_id, doctor_id, start_at, end_at, reason }**	**201 { id, patient_id, doctor_id, start_at, end_at, status }**
Obtener / listar citas	**GET**	**/api/v1/appointments**	**query: ?patient_id=&doctor_id=&date=&status=&page=**	**200 { items:[...], total }**
Actualizarbcita (confirmar /cancelar)	**PATCH**	**/api/v1/appointments/{id}**	**body: `{ status:"confirmed"**	**"cancelled"**
Registros médicos (file metadata) - subir	**POST**	**/api/v1/patients/{id}/records**	**multipart/form-data (file) + JSON metadata {type, notes}**	**201 { record_id, filename, url, uploaded_at }**
Listar registros	**GET**	**/api/v1/patients/{id}/records**	**query: ?type=**	**200 { items:[{id,type,filename, url,uploaded_at}] }**
Obtener registro	**GET**	**/api/v1/patients/{id}/records/{record_id}**	**-**	**200 file stream / or metadata JSON**
Prescripciones	**POST**	**/api/v1/patients/{id}/prescriptions**	**body: { doctor_id, meds:[{name,dose,frequency,duration}], notes }**	**201 { prescription_id, created_at, meds }**
Buscar (global)	**GET**	**/api/v1/search**	**query: `?q=&type=patients**	**doctors**
Health check	**GET**	**/api/v1/health**	**-**	**200 { status:"ok", time }**

2.1. Request (POST /appointments):

{

"patientName": "Carlos Pérez",

"professionalId": 3,

"startAt": "2025-10-26T10:00:00Z",

"endAt": "2025-10-26T10:30:00Z"

}

Response:

{

"id": 17,

"patientName": "Carlos Pérez",

"professionalId": 3,

"startAt": "2025-10-26T10:00:00Z",

"endAt": "2025-10-26T10:30:00Z",

"status": "confirmed" **}**

2.2. Lista de Errores

Código	Causa	Ejemplo
400	Datos inválidos	Faltan campos o formato incorrecto
401	No autenticado	Token JWT no válido o ausente
404	No encontrado	Cita o usuario inexistente
409	Conflicto	Cita duplicada o solapada
500	Error del servidor	Fallo interno en la API

3. Decisiones arquitectónicas (ADRs)

- **ADR-001:** Base de Datos relacional (SQLite3)

Fecha: 2025/11/25

Contexto:

- Se necesita una base de datos para manejar relaciones entre usuarios, doctores y citas médicas.

Decisión:

- Usar SQLite como sistema de base de datos relacional, ligero, simple de usar y adaptado a la escala actual y proyectada del trabajo.

Alternativas consideradas:

- MongoDB (flexible, pero sin relaciones)
- PostgreSQL (robusto y relacional)

Efectos:

- Facilidad de consultas
- Disponibilidad de scripts e implementación rápida.
- Menor configuración inicial

Estado: Implementada.

- **ADR-002: Autenticación con JWT**

Contexto:

El sistema necesita autenticar usuarios (administradores y doctores) de forma segura.

Decisión:

Usar JSON Web Tokens (JWT) con expiración de 1 hora**.**

Efectos:

- Ligero y sin estado en el servidor
- Requiere renovación del token

Estado: Implementada

4. Seguridad y Roles

4.1. Modelo de autenticación

- JWT (JSON Web Token): cada usuario recibe un token válido por 1 hora.
- Renovación automática si el usuario sigue activo.
- Tokens firmados con una clave restringida guardada en .env.

Recurso	Administrador	Profesional
Crear cita	PERMITIDO	PERMITIDO
Agendar cita	PERMITIDO	NO AUTORIZADO
Editar cita	PERMITIDO	NO AUTORIZADO
Ver citas	PERMITIDO	PERMITIDO (SOLO ASIGNADAS)
Eliminar cita	PERMITIDO	NO AUTORIZADO (Solicitud a Admin)
Crear usuario	PERMITIDO	NO AUTORIZADO

4.2. Protección de datos

- Contraseñas cifradas con bcrypt
- Variables sensibles en .env (JWT_SECRET, DB_PASSWORD)
- Datos personales (correo, nombre) protegidos bajo Ley 1581 de 2012 (Colombia)

5. Implementación mínima (MVP)

Usuarios: Recepcionista / Médico

Funcionalidades mínimas:

Autenticación: register / login (JWT)

Pacientes: crear, listar, ver, editar, eliminar

Citas: crear, listar, ver, editar, eliminar; relacionar cita con paciente

Frontend mínimo:

Login Page

Dashboard (resumen)

Pacientes: lista + formulario crear/borrar + detalle

Doctor: lista + formulario crear/borrar

Citas: lista + formulario crear/ borrar

Persistencia: archivos JSON (data/) para desarrollo, actualmente se ya se completo y quedo hecha la migración a Bases de Datos SQLite.

API - Medicor

Base URL (desarrollo)

http://localhost:4000

Autenticación

- **Método:** Authorization header
- **Formato:** Authorization: Bearer <token>
- Endpoints de auth no requieren token.

Endpoints principales

1) Auth

POST /api/auth/register

Body: { "username": "string", "password": "string" }

Response 201: { ok: true, data: { id, username, token } }

POST /api/auth/login

Body: { "username": "string", "password": "string" }

Response 200: { ok: true, data: { id, username, token } }

2) Pacientes (protegido)

GET /api/pacientes

Headers: Authorization

Response 200: { ok: true, data: [{ id, nombre, email, telefono }, ...] }

POST /api/pacientes

Body: { "nombre": "string", "email": "string", "telefono": "string" }

Response 201: { ok: true, data: { id, nombre, ... } }

GET /api/pacientes/:id

Response 200: { ok: true, data: { id, nombre, ... } }

PUT /api/pacientes/:id

Body: fields a actualizar

Response 200: { ok: true, data: { ... } }

DELETE /api/pacientes/:id

Response 204: No content

3) Citas (protegido)

GET /api/citas

- **Response 200:** { ok: true, data: [{ id, pacienteId, fecha, motivo }, ...] }

POST /api/citas

- Body: { "pacienteId": number, "fecha": "ISO_string", "motivo": "string" }
- Response 201: { ok: true, data: { id, pacienteId, fecha, motivo } }

GET /api/citas/:id

- Response 200: { ok: true, data: { ... } }

PUT /api/citas/:id

- Body: fields a actualizar
- Response 200: { ok: true, data: { ... } }

DELETE /api/citas/:id

- Response 204: No content

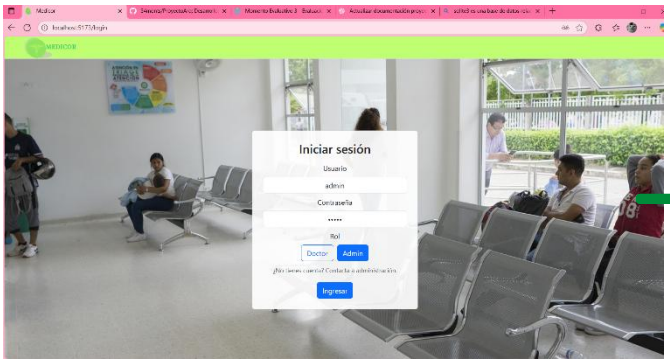
Errores comunes (estructura)

- 400 Bad Request: { ok: false, msg: "Faltan datos" }
- 401 Unauthorized: { ok: false, msg: "No token, acceso denegado" }
- 404 Not Found: { ok: false, msg: "No encontrado" }
- 500 Internal: { ok: false, msg: "Error interno del servidor" }

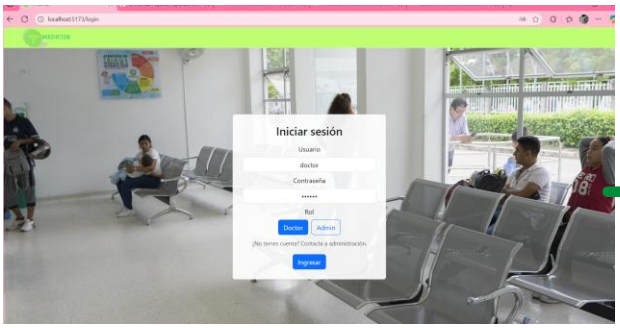
6. Pruebas y Validación

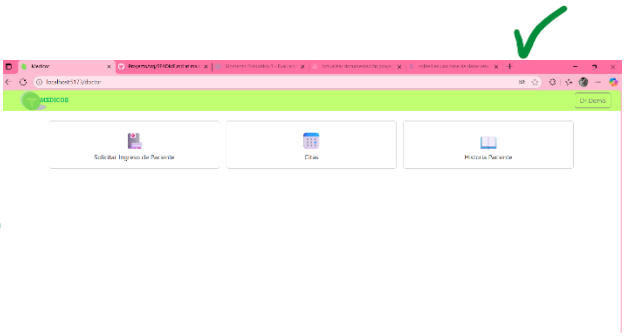
Caso	Entrada	Esperado
Crear cita válida	Horario libre	201 Created
Cita solapada	Mismo profesional y hora	409 Conflict
Sin token JWT	—	401 Unauthorized

Login (Administración)



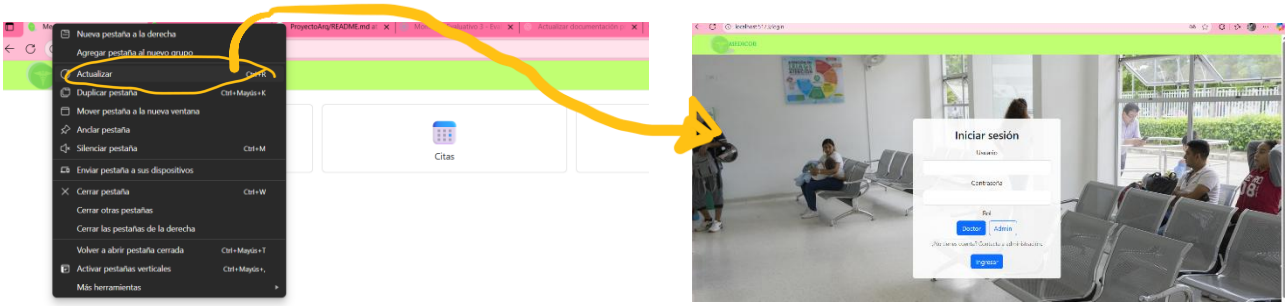
Login (Doctor)



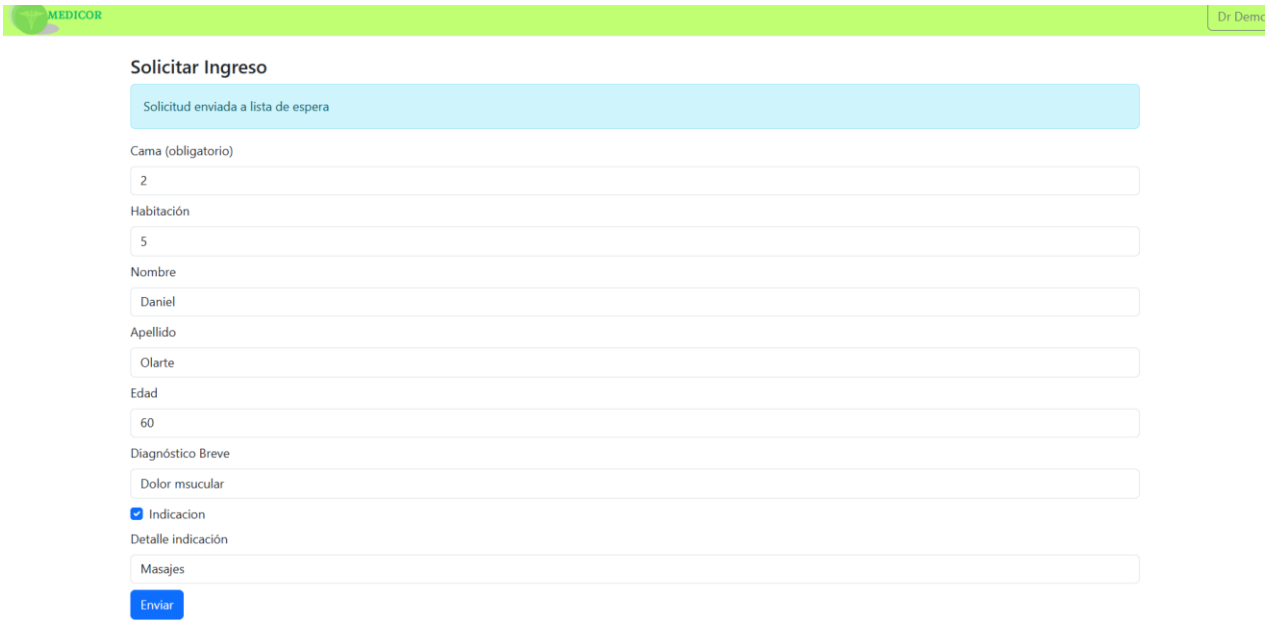


Detección de Usuario no Verificado

Se cierra la sesión en una de las ventanas y se actualiza la pagina



Registro de Pacientes



Carga Solicitud a lista de espera

Aprobación e ingreso a la lista principal

7. Guía de Despliegue y README

Requisitos

- Node.js + Nodemon
- REACT
- SQLite
- npm

8. Checklist de cierre

Ítem	Verificación
Diagrama y diccionario de datos completos	
Contrato de API documentado	
2–3 ADR nuevos creados	
Roles y seguridad definidos	
MVP funcional con 1 regla de negocio	
Pruebas y evidencias incluidas	
README y guías de ejecución