

**Actividad 6 – Entrega final del proyecto: Sistema de agendamiento digital para  
consultorios médicos pequeños**

Jhonatan David Becerra Donado

Proyecto de Software

Tatiana Cabrera

Facultad de Ingeniería

Corporación Universitaria Iberoamericana

Programa de Ingeniería de Software

06 de junio de 2025

**Introducción**

El presente documento constituye la segunda entrega del proyecto "Sistema de Gestión para Consultorio Médico", cuyo objetivo es avanzar hacia la construcción de un prototipo funcional que dé respuesta a las necesidades identificadas en la fase inicial de formulación.

Este proyecto busca brindar una solución tecnológica eficiente para optimizar la gestión administrativa y operativa de un consultorio médico, permitiendo el manejo adecuado de pacientes, citas médicas, historiales clínicos y facturación.

En esta fase se abordarán las actividades correspondientes a las etapas de diseño, construcción de prototipo y pruebas, conforme al ciclo de vida del desarrollo de software. Se presentarán los requisitos funcionales y no funcionales, las historias de usuario, los diagramas de modelado, los prototipos de baja y alta fidelidad, las pruebas realizadas y las consideraciones futuras para la evolución del sistema.

La implementación cuidadosa de cada una de estas etapas permitirá avanzar hacia un producto de software sólido, aplicando buenas prácticas de desarrollo que garanticen calidad, funcionalidad y usabilidad para los usuarios finales.

## **Requisitos funcionales y no funcionales**

A continuación, se especifican los requisitos funcionales y no funcionales del Sistema de Gestión para Consultorio Médico, identificados a partir del análisis de las necesidades del consultorio.

### ***Requisitos funcionales (RQF)***

<b>Código</b>	<b>Requisito funcional</b>	<b>Descripción</b>
RQF01	Registro de pacientes	Permitir el registro, edición y eliminación de la información personal de los pacientes.
RQF02	Agendamiento de citas	Permitir la programación, edición y cancelación de citas médicas.
RQF03	Gestión de historial médico	Permitir la creación, consulta y actualización del historial clínico de cada paciente.
RQF04	Facturación de servicios	Generar facturas por las consultas médicas y otros servicios prestados.
RQF05	Gestión de usuarios	Permitir la creación, autenticación y administración de usuarios con diferentes roles (médico, asistente, administrador).

RQF06	Generación de reportes	Permitir la generación de reportes de pacientes atendidos, citas agendadas y facturación.
-------	------------------------	---

***Requisitos no funcionales (RQNF)***

Código	Requisito no funcional	Descripción
RQNF01	Seguridad	El sistema deberá garantizar la confidencialidad de los datos de los pacientes mediante autenticación y autorización de usuarios.
RQNF02	Usabilidad	La interfaz del sistema deberá ser intuitiva y amigable para facilitar su uso por el personal del consultorio.
RQNF03	Rendimiento	Las operaciones más comunes (registro de pacientes, agendamiento de citas, etc.) deberán ejecutarse en un tiempo menor a 3 segundos.
RQNF04	Disponibilidad	El sistema deberá estar disponible al menos el 95% del tiempo durante las horas laborales del consultorio.

RQNF05	Escalabilidad	El sistema deberá permitir agregar nuevas funcionalidades en el futuro sin afectar las existentes.
RQNF06	Compatibilidad	El sistema deberá ser compatible con navegadores web modernos y dispositivos móviles.

## **Historias de Usuario**

Este tablero corresponde al seguimiento y gestión de tareas para el desarrollo del **Sistema de gestión de consultorio médico**. El proyecto tiene como propósito implementar un software que facilite el registro de pacientes, la gestión de citas médicas, el manejo de historiales clínicos y la generación de reportes administrativos, con el fin de optimizar los procesos de atención y administración en un consultorio.

### **Objetivo general:**

Desarrollar un prototipo funcional de un sistema de gestión para un consultorio médico que permita optimizar la administración de pacientes, citas, historiales clínicos y facturación.

### **Metodología utilizada:**

Se emplea una metodología ágil basada en tableros Kanban para la gestión de tareas e historias de usuario. El ciclo de vida aplicado corresponde al **modelo de prototipado** (recolección de requisitos, diseño rápido, construcción, evaluación, refinamiento y entrega).

### **Estructura del tablero:**

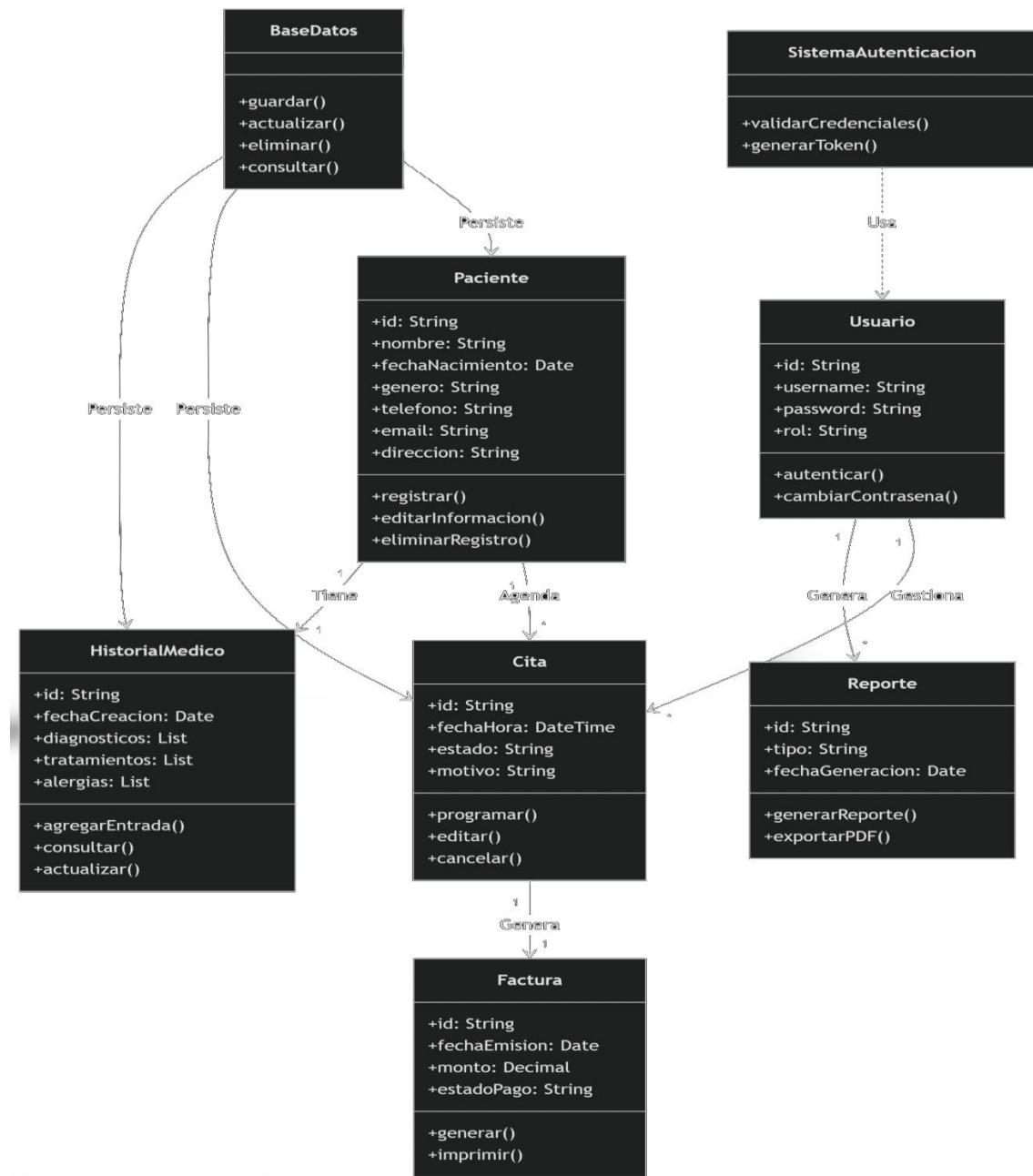
- **Backlog:** Tareas e historias por priorizar y planear.
- **To Do:** Tareas listas para iniciar.
- **In Progress:** Tareas actualmente en desarrollo.
- **Done:** Tareas finalizadas y validadas.

Enlace tablero de la metodología ágil: [Planner - Proyecto de Software: Sistema de agendamiento digital para consultorios médicos pequeños](#)

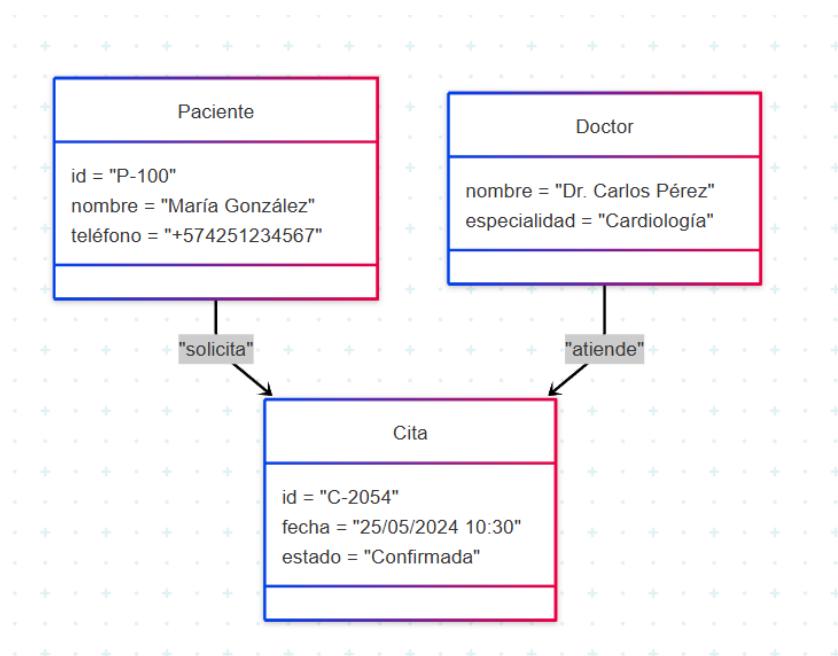
## Modelamiento

### Conceptual:

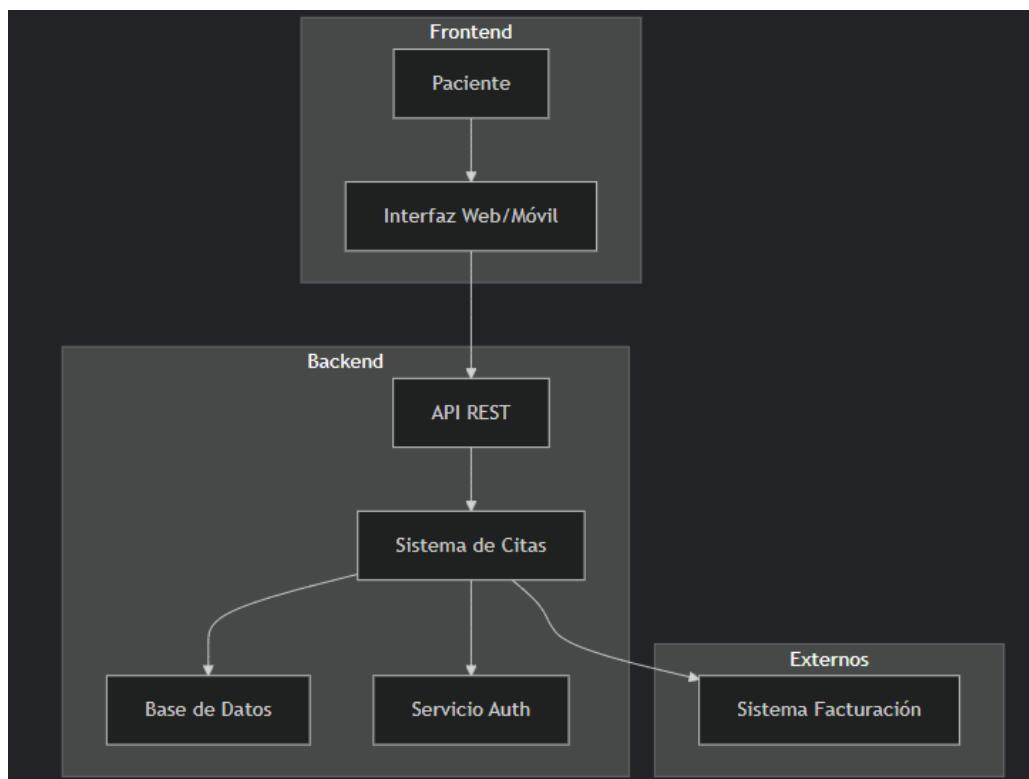
- Diagrama de clases



- Diagrama de objetos

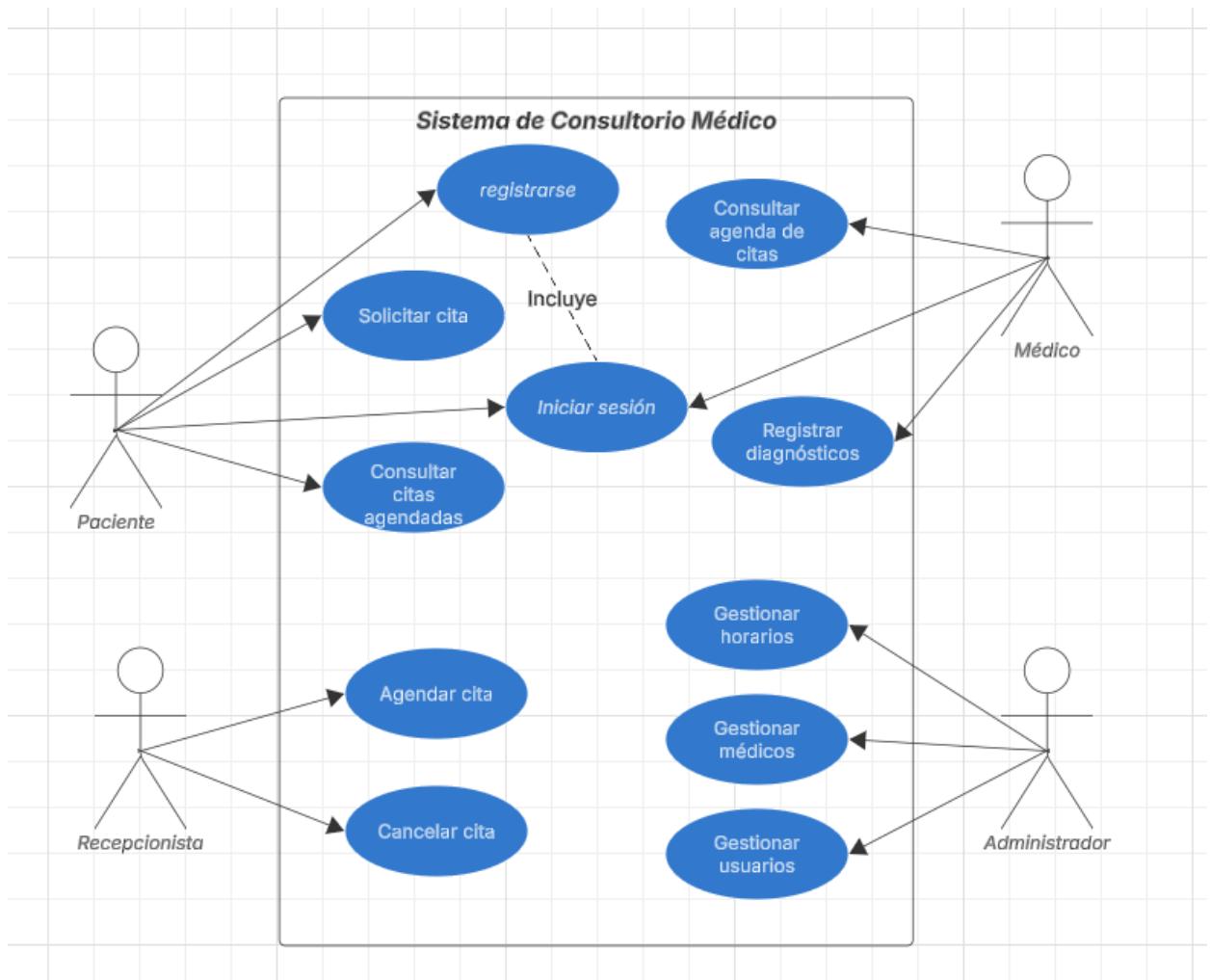


- *Diagrama de componentes*

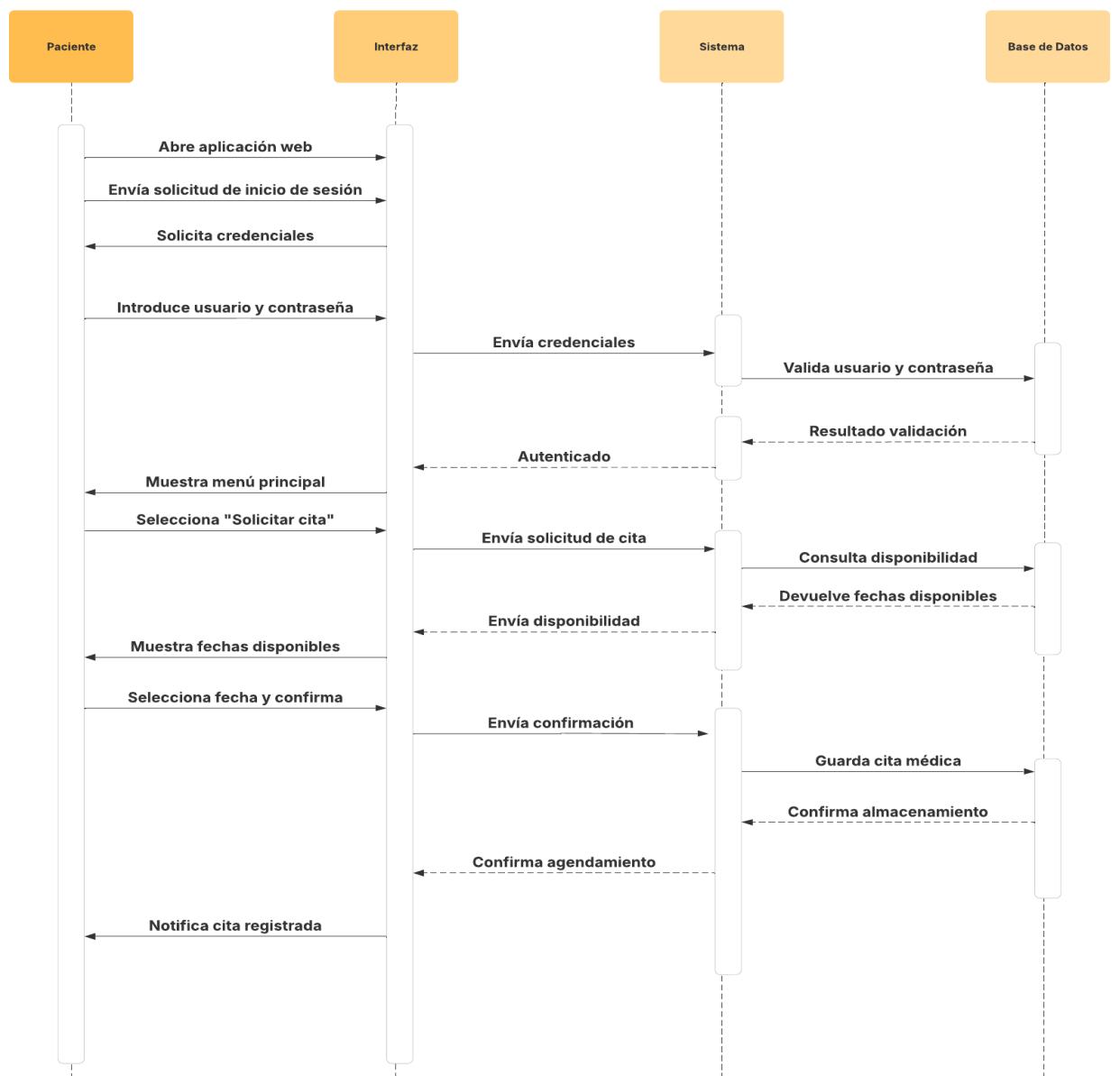


### Comportamiento:

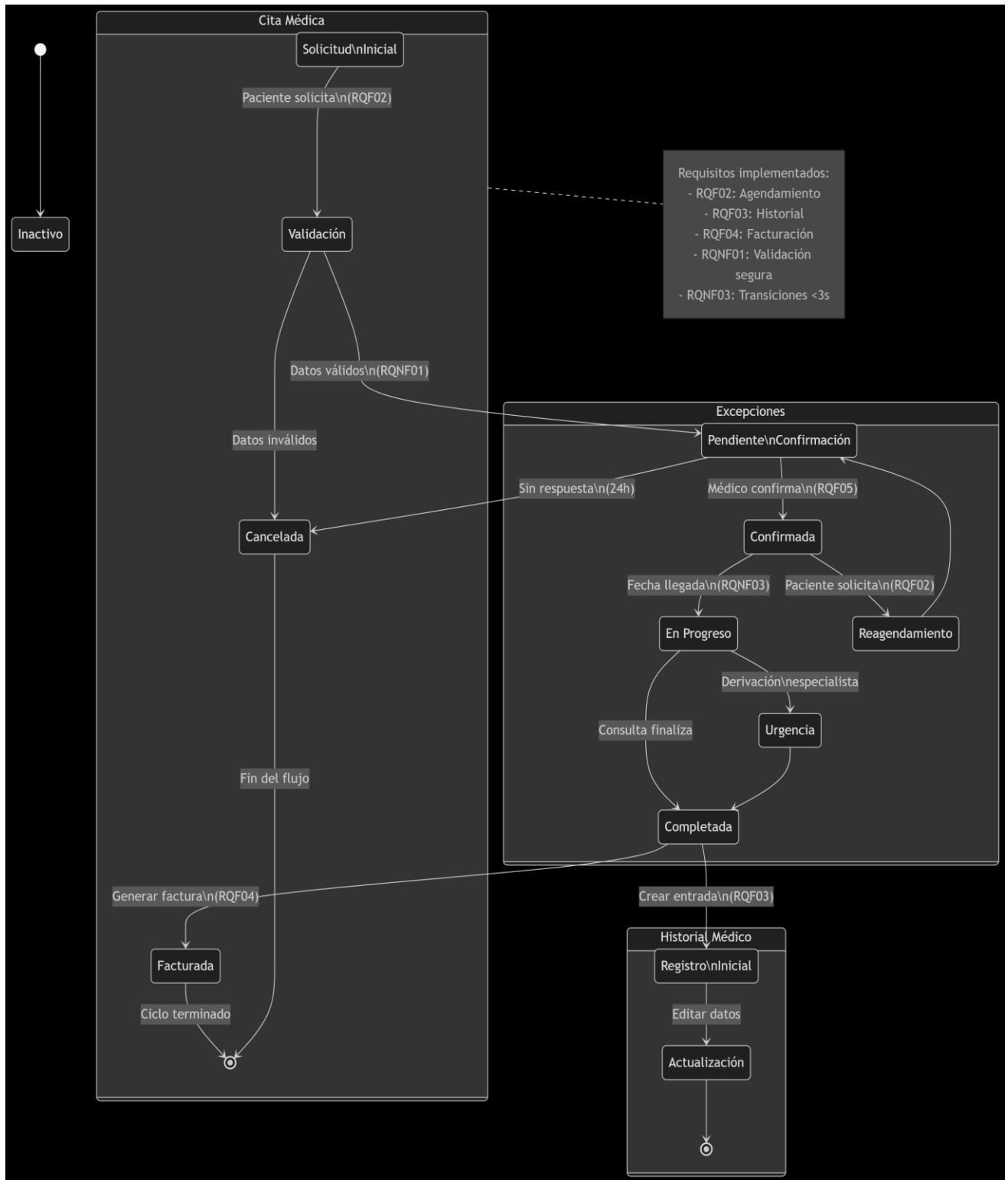
- *Diagrama de casos de uso*



- Diagrama de secuencia



- Diagrama de estados



## Diseño:

### - Prototipo de Baja fidelidad

*Dashboard doctor:*

**Welcome, Dr. Sarah**

Here's your schedule for today, Saturday, May 3

**Total Appointments: 4**

**Checked In: 2**

**Total Patients: 1,248**

**Current Patient:** Robert Wilson, 43 years old, Male  
Appointment Type: Consultation  
Time: 11:15 AM (25 min)  
Last Visit: Mar 10, 2025  
Patient ID: P-78934

**Today's Schedule:**

- 09:00 AM John Smith (Check-up)
- 10:30 AM Mary Johnson (Urgent) (Follow-up)
- 11:15 AM Robert Wilson (Now) (Consultation)

**Day Week Month**

**Dr. Sarah Smith Doctor**

**Logout**

*Inicio de Sesión:*

**Welcome back**

Please sign in to your account

Email address:

Password:

Remember me [Forgot your password?](#)

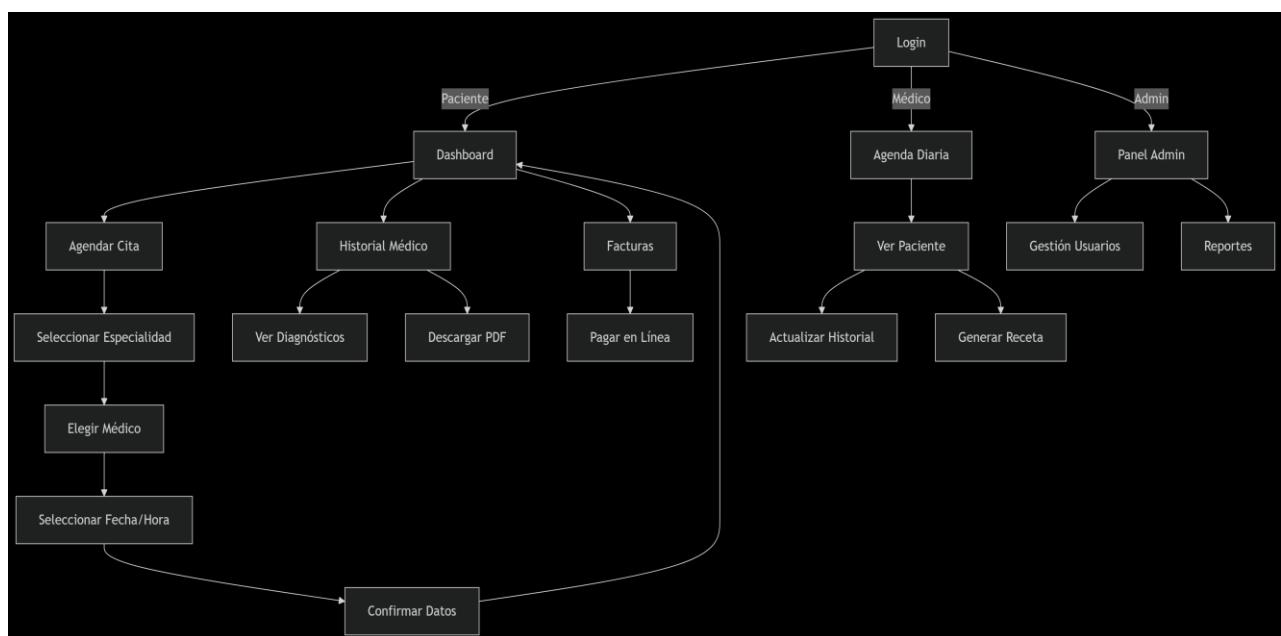
**Sign in**

[Don't have an account? Register now](#)

For demo purposes, you can use:  
 Patient: patient@example.com  
 Doctor: doctor@example.com  
 Password for all accounts: password

*Dashboard Paciente:*

## - Mapa de navegación



## Fase de Pruebas

En esta fase, se realizaron pruebas funcionales y de usabilidad sobre los prototipos desarrollados, con el objetivo de validar la correcta implementación de las funcionalidades propuestas y garantizar una experiencia de usuario coherente con los requisitos planteados.

Las pruebas se enfocaron en verificar aspectos como:

- La correcta navegación entre las diferentes vistas y componentes del sistema.
- El cumplimiento de los flujos definidos en los diagramas de casos de uso.
- La validación de entradas y manejo de errores.
- La consistencia del diseño y la alineación con los criterios de aceptación definidos previamente.

A continuación, se comparten los recursos de la fase de pruebas:

- **Repositorio del proyecto (GitHub):**

[Repositorio proyecto](#)

- **Enlace al prototipo:**

<https://transcendent-macaron-b9929f.netlify.app/>

- **Video demostrativo del funcionamiento del prototipo:**

[https://drive.google.com/file/d/1vpegFBcSS6NLgsPIvACT81G\\_Bn3u5wxj/view?usp=sharing](https://drive.google.com/file/d/1vpegFBcSS6NLgsPIvACT81G_Bn3u5wxj/view?usp=sharing)

Manual técnico

*Fragmento: Lógica de Agendamiento de Citas*

```
async function agendarCita(formData) {
  const res = await fetch('/api/citas', {
    method: 'POST',
    body: JSON.stringify(formData)
  });
  if (res.ok) toast.success('Cita agendada exitosamente');
  else toast.error('Error al agendar cita');
}
```

Este fragmento representa la lógica principal del frontend que permite enviar los datos del formulario al backend y brindar retroalimentación al usuario. Se emplea un fetch con validación del estado de respuesta y notificaciones mediante Solid Toast.

*Fragmento: Validación de Formulario*

```
function validarFormulario(cita) {
  if (!cita.fecha || !cita.hora || !cita.pacienteId || !cita.medicoId) {
    return { valido: false, mensaje: "Todos los campos son obligatorios" };
  }
  const fechaActual = new Date();
  if (new Date(cita.fecha) < fechaActual) {
    return { valido: false, mensaje: "La fecha no puede estar en el pasado" };
  }
  return { valido: true };
}
```

Este fragmento garantiza que el usuario no pueda enviar una cita sin llenar los campos requeridos o con una fecha inválida.

## Pruebas del Sistema

### *Pruebas Unitarias*

<b>Tipo</b>	<b>Caso de prueba</b>	<b>Resultado esperado</b>	<b>Resultado obtenido</b>	<b>Análisis</b>
Unitaria	Validar fecha pasada al agendar	Rechazo con mensaje de error	Rechazo correcto	Validación exitosa
Unitaria	Validar campos obligatorios vacíos	Alerta visual al usuario	Mensaje "Todos los campos son obligatorios"	Correcto
Unitaria	Validación de formato de correo en registro	Rechazo si el formato es incorrecto	Error mostrado en tiempo real	Funciona correctamente
Unitaria	Confirmación de contraseña	Las contraseñas deben coincidir	Se muestra alerta si no coinciden	Flujo de validación funcional
Unitaria	Longitud mínima de contraseña	Error si es menor a 8 caracteres	Mensaje de advertencia correcto	Correctamente gestionado

### *Pruebas de Integración*

<b>Tipo</b>	<b>Caso de prueba</b>	<b>Resultado esperado</b>	<b>Resultado obtenido</b>	<b>Análisis</b>

Integración	Crear paciente y agendar cita	Flujo sin errores	Datos conectados correctamente	Flujo funcional completo
Integración	Iniciar sesión y visualizar agenda	Acceso al panel y citas visibles	Citas correctamente listadas	Interacción correcta entre módulos
Integración	Eliminar cita y validar eliminación	Cita desaparece de la agenda	Eliminación exitosa	Flujo correcto
Integración	Cambiar contraseña y volver a iniciar sesión	Acceso con nueva clave	Cambio registrado y acceso exitoso	Funciona adecuadamente
Integración	Acceder sin sesión iniciada	Redirección al login	Protección de rutas activa	Comportamiento esperado

### ***Pruebas de Usabilidad***

Observaciones realizadas sobre usuarios de prueba no técnicos:

- La navegación fue intuitiva.
- Los colores y botones facilitaron la comprensión.
- El sistema notificó los errores en tiempo real.

### ***Prueba Automatizada***

Se implementó una prueba con Playwright para validar el flujo de inicio de sesión:

```
test('login success', async ({ page }) => {
  await page.goto('https://transcendent-macaron-b9929f.netlify.app/login');
  await page.fill('#email', 'usuario@demo.com');
  await page.fill('#password', '12345678');
  await page.click('button[type="submit"]');
  await expect(page).toHaveURL(/dashboard/);
});
```

Resultado: la prueba pasó correctamente simulando un login exitoso.

## Despliegue del Sistema

El sistema fue desplegado utilizando la plataforma **Netlify**, seleccionada por su compatibilidad con frameworks modernos y su integración directa con GitHub. El proceso de despliegue se realiza automáticamente cada vez que se detecta un nuevo push en la rama principal. La aplicación es accesible públicamente en el siguiente enlace:

🔗 <https://transcendent-macaron-b9929f.netlify.app/>

Netlify permite una experiencia de desarrollo fluida, rápida implementación y soporte de configuraciones personalizadas para redirecciones y rutas protegidas.

### **Lenguaje Orientado a Objetos Utilizado**

El desarrollo del sistema se realizó utilizando **TypeScript** junto con el framework **SolidJS**, ambos integrados dentro del entorno de **Astro**. Aunque TypeScript es un superset de JavaScript, permite aplicar conceptos de la **programación orientada a objetos (OOP)**, tales como:

- **Encapsulamiento:** Separando lógica en funciones y componentes reutilizables.
- **Abstracción:** Modularizando funcionalidades como validación de formularios o lógica de autenticación.
- **Polimorfismo y herencia (mediante interfaces y tipos):** Se definen tipos de datos y estructuras reutilizables, como las interfaces para usuarios, citas y pacientes.

SolidJS potencia esta estructura orientada a componentes (similar al enfoque de clases y objetos) y Astro permite integrarlos de forma eficiente en la arquitectura del sistema. La orientación a objetos se evidencia en el diseño modular, mantenable y reutilizable del sistema, clave para su escalabilidad y claridad.

## **Conclusiones**

La fase final del desarrollo del sistema de gestión de citas médicas permitió consolidar todos los conocimientos adquiridos en el proceso de planeación, diseño, construcción y validación de software. La implementación de funcionalidades clave, como el registro de usuarios, el agendamiento de citas y la administración de pacientes, demuestra la aplicabilidad real de las metodologías ágiles en contextos de necesidades concretas.

El uso de tecnologías modernas y patrones de diseño orientados a componentes contribuyó a una arquitectura clara, escalable y mantenible. Las pruebas realizadas evidencian que el sistema cumple con los requerimientos funcionales establecidos y responde de manera adecuada a las interacciones del usuario.

Finalmente, la experiencia adquirida en el ciclo completo de vida del software, desde los prototipos hasta el despliegue y la validación, refuerza las habilidades técnicas y profesionales necesarias para enfrentar proyectos reales en el ámbito del desarrollo de software.

## Bibliografía

- [1] M. Hernández, *Ciclo de vida de desarrollo ágil de software seguro*, Bogotá, Colombia: Fundación Universitaria Los Libertadores, 2020.
- [2] C. Zapata Jaramillo y F. Arango Isaza, *Unc-method: un método de desarrollo de software basado en problemas*, Medellín, Colombia: Universidad Nacional de Colombia, Facultad de Ingeniería, 2009.
- [3] G. Hernández, Á. Martínez, R. Jiménez y F. Jiménez, “Scrum y Peopleware: elementos clave para la gestión en la construcción de software,” *Revista Ibérica de Sistemas e Tecnologias de Informação*, no. E19, pp. 265–277, 2019.
- [4] J. Z. Gamboa, “Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software,” *INNOVA Research Journal*, vol. 3, no. 10, pp. 20–33, 2018.
- [5] K. W. Mutter, “Propiedad intelectual y desarrollo en Colombia,” *Estudios Socio-Jurídicos*, vol. 8, no. 2, pp. 85–101, 2006.