
	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 1 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación:	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Pruebas de integración

Presentado por


Darwin steven Gomez
Constanza Quira
Liliana Perez

centro de teleinformatica y producción industrial
análisis y desarrollo de software
sena regional cauca
popayan cauca
2025

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 2 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Contenido

1. Ficha tecnica	¡Error! Marcador no definido.
2. Pruebas de integracion _modulo carrito.....	¡Error! Marcador no definido.
3. Pruebas de integracion_Modulo categoria de Procedimientos	¡Error! Marcador no definido.
4. Prueba de integracion_Modulo de chat	¡Error! Marcador no definido.
5. Pruebas de integracion_Modulo de citas	¡Error! Marcador no definido.
6. Pruebas de integracion_ Modulo de consentimientos...	¡Error! Marcador no definido.
7. Pruebas de integración_Modulo de contacto.....	5
8. Pruebas de integración_Modulo de Exámenes.....	5
9. Pruebas de integración_Modulo historial medico.....	5
10. Pruebas de integración_Modulo ordenes.....	5
11. Pruebas de integración orden procedimiento.....	5
12. Pruebas de integración_Modulo procedimientos.....	5
13. Pruebas de integración_Modulo de usuarios.....	5

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 3 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

1. Ficha técnica de prueba de integración

Módulo: Cambio de Contraseña

1.1 Propósito de la prueba

Validar integralmente el funcionamiento del **módulo de Cambio de Contraseña**, asegurando que las rutas definidas en el router de Express (CambiarContrasenaRouters) respondan correctamente y se integren con el **middleware de autorización** y los **servicios de lógica de negocio** (CambiarContrasenaServices).


1.2 Alcance y Cobertura

Estas pruebas de integración verifican el flujo completo entre los componentes: Router → Middleware (Authorization) → Servicios (CambiarContrasenaServices) → Respuesta HTTP

Método	Endpoint	Descripción funcional	Respuesta esperada
POST	/apicambiarcontrasena/cambiarcontrasena	Permite al usuario autenticado cambiar su contraseña actual por una nueva.	HTTP 200 con { success: true }
POST	/apicambiarcontrasena/olvidocontrasena	Envía un correo de recuperación de contraseña al usuario.	HTTP 200 con { sent: true }
POST	/apicambiarcontrasena/resetearcontrasena/:token	Permite restablecer la contraseña mediante un token de seguridad.	HTTP 200 con { ok: true }

1.3 Componentes involucrados

Componente	Función dentro de la prueba
Express.js	Monta la aplicación y gestiona las rutas reales del backend.
Supertest	Simula peticiones HTTP a los endpoints para verificar respuestas.
Middleware Authorization (mockeado)	Inyecta un usuario simulado en el objeto req ({ id: 3 }) para probar la autenticación.
Servicio CambiarContrasenaServices (mockeado)	Contiene la lógica de negocio (cambiar, solicitar y resetear contraseñas).

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 4 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Componente	Función dentro de la prueba
Router CambiarContrasenaRouters	Define los endpoints que se integran con el middleware y los servicios.

1.4 Escenarios de prueba

Nº Escenario	Descripción	Resultado Esperado
1 Cambiar contraseña	Se envía una petición POST a /apicambiarcontrasena/cambiarcontrasena con los campos actual y nueva.	Responde 200 OK con { success: true }.
2 Solicitar restablecimiento	Se envía una petición POST a /apicambiarcontrasena/olvidocontrasena con el correo del usuario.	Responde 200 OK con { sent: true }.
3 Resetear contraseña	Se envía una petición POST a /apicambiarcontrasena/resetearcontrasena/:token con la nueva contraseña.	Responde 200 OK con { ok: true }.



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área
Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
jest.mock('../middleware/Authorization', () => ({
  authorization: (req, _res, next) => { req.usuario = { id: 3 }; next(); },
}));

const mockService = {
  cambiarcontrasena: jest.fn(),
  solicitarReset: jest.fn(),
  resetearPassword: jest.fn(),
};
jest.mock('../services/CambiarContrasenaServices', () => mockService);

const Router = require('../routers/CambiarContrasenaRouters');

describe('Routers de Cambio de Contraseña - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apicambiarcontrasena', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('POST /cambiarcontrasena devuelve resultado', async () => {
    mockService.cambiarcontrasena.mockResolvedValueOnce({ success: true });
    const res = await request(app).post('/apicambiarcontrasena/cambiarcontrasena').send({ actual: 'a', nueva: 'b' });
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('success', true);
  });


  test('POST /olvidocontrasena devuelve resultado', async () => {
    mockService.solicitarReset.mockResolvedValueOnce({ sent: true });
    const res = await request(app).post('/apicambiarcontrasena/olvidocontrasena').send({ correo: 'x@x.com' });
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('sent', true);
  });

  test('POST /resetearcontrasena/:token devuelve resultado', async () => {
    mockService.resetearPassword.mockResolvedValueOnce({ ok: true });
    const res = await request(app).post('/apicambiarcontrasena/resetearcontrasena/123').send({ nueva: 'c' });
```

```
dstevegnz@stevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- --_tests_ /CambiarContrasenaRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_ /CambiarContrasenaRouters.full.int.test.js

PASS _tests_ /CambiarContrasenaRouters.full.int.test.js
  Routers de Cambio de Contraseña - Integración completa
    - POST /cambiarcontrasena devuelve resultado (85 ms)
    - POST /olvidocontrasena devuelve resultado (8 ms)
    - POST /resetearcontrasena/:token devuelve resultado (7 ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 1.319 s
Ran all test suites matching /_tests_ /CambiarContrasenaRouters.full.int.test.js/i.
dstevegnz@stevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$
```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 6 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

2. Pruebas de Integración — Módulo de Carrito

2.1 Propósito de la prueba

Validar integralmente el funcionamiento de las rutas del **módulo de Carrito**, verificando que las operaciones de listado, adición, eliminación y limpieza del carrito de compras funcionen correctamente para un **usuario autenticado**, y que las respuestas HTTP sean las esperadas según cada escenario.

2.2 Alcance y Cobertura

Estas pruebas comprueban la interacción entre las capas:

Router → Middleware (Authorization) → Servicios (CarritoServices) → Respuesta HTTP.

Método	Endpoint	Descripción funcional	Respuesta esperada
GET	/apicarrito/listarmicarrito	Lista los elementos del carrito del usuario autenticado.	HTTP 200 con arreglo de objetos.
POST	/apicarrito/agregaramicarrito	Agrega un procedimiento al carrito.	HTTP 201 con { id, id_procedimiento }.
DELETE	/apicarrito/eliminardecarrito/:id	Elimina un procedimiento específico del carrito.	HTTP 200 con { mensaje: string }.
DELETE	/apicarrito/limpiarmicarrito	Vacía completamente el carrito del usuario autenticado.	HTTP 200 con { mensaje: string }.

2.3 Componentes involucrados

Componente	Descripción
Express.js	Framework que gestiona las rutas y peticiones HTTP.
Supertest	Librería que simula solicitudes HTTP para pruebas de API.
Middleware Authorization (mockeado)	Injecta el usuario autenticado (req.usuario = { id, rol }).
Servicio CarritoServices (mockeado)	Lógica de negocio para listar, agregar y limpiar el carrito.
Router CarritoRouters	Define las rutas HTTP bajo /apicarrito.

2.4 Escenarios de prueba



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

Nº Escenario	Descripción	Resultado Esperado
1 Listar carrito	Solicitud GET con usuario autenticado.	200 OK con [{ id: 1 }].
2 Agregar al carrito	Envío de POST con { id_procedimiento: 3 }.	201 Created con { id: 5, id_procedimiento: 3 }.
3 Eliminar del carrito	DELETE /eliminardecarrito/2.	200 OK con { mensaje: '...' }.
4 Limpiar carrito	DELETE /limpiarmicarrito.	200 OK con { mensaje: '...' }.

```
describe('Routers de Carrito - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apicarrito', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('GET /listarmicarrito devuelve lista para usuario autenticado', async () => {
    mockService.listarCarritoPorUsuario.mockResolvedValueOnce([ { id: 1 } ]);
    const res = await request(app).get('/apicarrito/listarmicarrito').set('x-user-id', '9');
    expect(res.status).toBe(200);
    expect(res.body).toEqual([ { id: 1 } ]);
    expect(mockService.listarCarritoPorUsuario).toHaveBeenCalledWith(9);
  });


  test('POST /agregaramicarrito devuelve 201 con nuevo elemento', async () => {
    mockService.agregarAlCarrito.mockResolvedValueOnce({ id: 5, id_procedimiento: 3 });
    const res = await request(app).post('/apicarrito/agregaramicarrito').send({ id_procedimiento: 3 });
    expect(res.status).toBe(201);
    expect(res.body).toHaveProperty('id', 5);
  });

  test('DELETE /eliminardecarrito/:id devuelve mensaje', async () => {
    mockService.eliminarDelCarrito.mockResolvedValueOnce(true);
    const res = await request(app).delete('/apicarrito/eliminardecarrito/2');
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('mensaje');
  });
});

dstevegnz@dstevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$ npm test -- --_tests_/CarritoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_/CarritoRouters.full.int.test.js

PASS _tests_/CarritoRouters.full.int.test.js
Routers de Carrito - integración completa
  GET /listarmicarrito devuelve lista para usuario autenticado (38 ms)
  POST /agregaramicarrito devuelve 201 con nuevo elemento (24 ms)
  DELETE /eliminardecarrito/:id devuelve mensaje (7 ms)
  DELETE /limpiarmicarrito devuelve ok (6 ms)

Test Suites: 1 passed, 1 total
Tests: 4 passed, 4 total
Snapshots: 0 total
Time: 1.087 s
Ran all test suites matching /_tests_/CarritoRouters.full.int.test.js/i.
dstevegnz@dstevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$
```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 8 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

3. Pruebas de Integración — Módulo Categoría de Procedimientos

3.1 Propósito de la prueba

Validar de forma integral el correcto funcionamiento de las rutas pertenecientes al **módulo de Categorías de Procedimientos**, comprobando que las operaciones CRUD respondan con los estados y estructuras esperadas cuando se integran el **router**, los **middlewares** y los **servicios** del backend.

3.2 Alcance y cobertura

Las pruebas verifican la integración completa del flujo: Router → Middleware (Authorization, PrimeraMayusculaCategoria) → Servicios → Respuesta HTTP.


Las pruebas verifican la integración completa del flujo:

Router → Middleware (Authorization, PrimeraMayusculaCategoria) → Servicios → Respuesta HTTP.

Método	Endpoint	Descripción funcional	Respuesta esperada
GET	/apicategoriaprocedimientos/listarcategorias	Lista todas las categorías registradas.	HTTP 200 con arreglo de objetos [{ id, nombre }].
GET	/apicategoriaprocedimientos/buscarcategoria/:id	Busca una categoría por ID.	HTTP 200 si existe, 404 si no existe.
POST	/apicategoriaprocedimientos/crearcategoria	Crea una nueva categoría y convierte valores de texto a booleanos.	HTTP 201 con { id, nombre, estado }.
PATCH	/apicategoriaprocedimientos/editarcategoria/:id	Edita los datos de una categoría existente.	HTTP 400 si el ID es inválido, 404 si no existe.
DELETE	/apicategoriaprocedimientos/eliminarcategoria/:id	Elimina una categoría.	HTTP 404 si la categoría no existe.

3.3 Componentes involucrados

Componente	Descripción
Express.js	Framework de servidor para el manejo

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 9 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Componente	Descripción
Supertest	de rutas y peticiones. Herramienta para simular solicitudes HTTP durante las pruebas.
Middleware Authorization (mockeado)	Autoriza la ejecución de las rutas sin validar tokens reales.
Middleware PrimeraMayusculaCategoria (mockeado)	Asegura la normalización de texto en el campo nombre.
Servicio CategoriaProcedimientosServices (mockeado)	Gestiona la lógica CRUD de categorías.
Router CategoriaProcedimientosRouters	Define las rutas y las conecta con los servicios.

```

dsteven@steven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- --_tests_/CategoriaProcedimientosRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_/CategoriaProcedimientosRouters.full.int.test.js

PASS Tests /CategoriaProcedimientosRouters.full.int.test.js
  Routers de Categoria de Procedimientos - Integración completa
    ✓ GET /listarcategorias devuelve lista (43 ms)
    ✓ GET /buscarcategoria/:id devuelve 404 cuando no existe (18 ms)
    ✓ POST /crearcategoria devuelve 201 y normaliza estado string (23 ms)
    ✓ PATCH /editarcategoria/:id valida id y devuelve 400 si inválido (7 ms)
    ✓ PATCH /editarcategoria/:id devuelve 404 cuando no existe (7 ms)
    ✓ DELETE /eliminarcategoria/:id devuelve 404 cuando no existe (6 ms)

Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 1.13 s
Run all test suites matching /_tests_/CategoriaProcedimientosRouters.full.int.test.js/i.
dsteven@steven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$


test('GET /buscarcategoria/:id devuelve 404 cuando no existe', async () => {
  mockService.buscarLaCategoria.mockResolvedValueOnce(null);
  const res = await request(app).get('/apicategoriaprocedimientos/buscarcategoria/99');
  expect(res.status).toBe(404);
});

test('POST /crearcategoria devuelve 201 y normaliza estado string', async () => {
  mockService.crearLaCategoria.mockImplementation(async (payload) => ({ id: 9, ...payload }));
  const res = await request(app).post('/apicategoriaprocedimientos/crearcategoria').send({ nombre: 'Corporal', estado: 'true' });
  expect(res.status).toBe(201);
  expect(res.body).toHaveProperty('id');
  expect(mockService.crearLaCategoria).toHaveBeenCalledWith({ nombre: 'Corporal', estado: true });
});

test('PATCH /editarcategoria/:id valida id y devuelve 400 si inválido', async () => {
  const res = await request(app).patch('/apicategoriaprocedimientos/editarcategoria/abc').send({});
  expect(res.status).toBe(400);
});

test('PATCH /editarcategoria/:id devuelve 404 cuando no existe', async () => {
  mockService.buscarLaCategoria.mockResolvedValueOnce(null);

```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 10 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

4. Pruebas de Integración — Módulo de Chat

4.1 Propósito de la prueba

Verificar la correcta integración del **Router de Chat**, asegurando que el endpoint principal /consultar procese peticiones POST correctamente, responda con el formato JSON esperado y devuelva un estado HTTP 200 sin errores.}


4.2 Alcance y cobertura

Estas pruebas validan la integración del flujo completo: Router → Controlador (mockeado) → Respuesta HTTP.

Método	Endpoint	Descripción	Resultado esperado
POST	/apichat/consultar	Consulta mensajes o respuestas del chatbot.	HTTP 200 con { ok: true }.

4.2 Componentes involucrados

Componente	Descripción
Express.js	Framework que gestiona el enrutamiento y la comunicación HTTP.
Supertest	Simula peticiones HTTP reales para pruebas de integración.
Controlador ChatControllers (mockeado)	Simula la función consultarchat, retornando un JSON con { ok: true }.
Router ChatRouters	Define la ruta /consultar y la asocia con el controlador.

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 11 de 36	
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario			

```

const express = require('express');
const request = require('supertest');

const mockCtrl = { consultarchat: jest.fn(async (_req, res) => res.json({ ok: true })) };
jest.mock('../controllers/ChatControllers', () => mockCtrl);

const Router = require('../routers/ChatRouters');

describe('Routers de Chat - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apichat', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('POST /consultar devuelve ok', async () => {
    const res = await request(app).post('/apichat/consultar').send({ q: 'hola' });
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('ok', true);
  });
});

```


```

dstevengsz@dstevengsz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- __tests__/ChatRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests __tests__/ChatRouters.full.int.test.js

PASS __tests__/ChatRouters.full.int.test.js
  Routers de Chat - integración completa
    ✓ POST /consultar devuelve ok (89 ms)

Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 1.013 s
Ran all test suites matching /__tests__/ChatRouters.full.int.test.js/i.
dstevengsz@dstevengsz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$

```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 12 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

5 Pruebas de integración — módulo de citas


5.1 Propósito de prueba

Validar integralmente el correcto funcionamiento del **router de Citas** (/apicitas) junto con su controlador y los servicios asociados (mockeados), comprobando que todas las rutas gestionen adecuadamente las solicitudes, los estados HTTP y las reglas de negocio.

5.2 Alcance y cobertura

Estas pruebas cubren todo el ciclo de vida de las citas médicas dentro del sistema:

Funcionalidad	Endpoint	Método	Casos probados	Resultado esperado
Listado de citas por rol	/apicitas/listarcitas	GET	Doctor / Usuario con validaciones	200 / 400
Búsqueda por ID	/apicitas/buscarcitas/:id	GET	Cita existente / inexistente	200 / 404
Creación de cita	/apicitas/crearcitas	POST	Usuario / Doctor con tipo válido	201 / 400
Edición de cita	/apicitas/editarcitausuario/:id	PATCH	Actualización y no encontrada	200 / 404
Edición por doctor	/apicitas/editarcita-doctor/:id	PATCH	Con requerimientos y notificaciones	200
Cancelación	/apicitas/cancelarcita/:id	PATCH	Existente / inexistente	200 / 404
Consultar horarios	/apicitas/horarios/:fecha	GET	Mapea duraciones según tipo	200
Cambio de estado	/apicitas/editarestadocita/:id	PATCH	Valida id, 404 / 200	400 / 404 / 200
Generar PDF órdenes	/apicitas/orden-examen/pdf/:id y /apicitas/orden-medicamentos/pdf/:id	GET	404 / 400 / 200	200 (PDF)
Marcar exámenes subidos	/apicitas/marcar-examen-subidos/:id	PATCH	Valida rol (usuario/doct or)	403 / 200

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 13 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Funcionalidad	Endpoint	Método	Casos probados	Resultado esperado
Ver mis citas	/apicitas/miscitas	GET	Usuario autenticado	200
Crear requerimientos	/apicitas/requerimientos	POST	Doctor	201
Reagendar cita	/apicitas/reagendarcita/:id	PATCH	Usuario / Doctor	200
Consultar todas las citas (asistente)	/apicitas/listarcitastodosusuariodesdeasistente	GET	Rol asistente	200

5.3 Componentes involucrados

Componente	Rol en la prueba
Express.js	Crea la app y monta el router de citas (/apicitas).
Supertest	Ejecuta peticiones HTTP simuladas y valida respuestas.
Authorization Middleware (mock)	Inyecta un usuario simulado con rol (doctor, usuario, asistente).
CitasServices (mock)	Simula la lógica de negocio (listar, crear, editar, notificar, etc.).
Router CitasRouters	Define todas las rutas y conecta con los controladores reales.



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
test("GET /listarcitas como doctor llama al servicio con doctorId y devuelve lista", async () => {
  mockCitasService.listarLasCitas.mockResolvedValueOnce([{ id: 1 }]);
  const res = await request(app)
    .get("/apicitas/listarcitas")
    .set("x-role", "doctor")
    .set("x-user-id", "42");
  expect(res.status).toBe(200);
  expect(res.body).toEqual([{ id: 1 }]);
  expect(mockCitasService.listarLasCitas).toHaveBeenCalled();
});

test("GET /listarcitas como usuario requiere doctorId en query o devuelve []", async () => {
  const res = await request(app)
    .get("/apicitas/listarcitas")
    .set("x-role", "usuario")
    .set("x-user-id", "7");
  expect(res.status).toBe(200);
  expect(res.body).toEqual([]);
  expect(mockCitasService.listarLasCitas).not.toHaveBeenCalled();
});

test("GET /listarcitas doctorId inválido para usuario devuelve 400", async () => {
  const res = await request(app)
    .get("/apicitas/listarcitas?doctorId=abc")
    .set("x-role", "usuario");
  expect(res.status).toBe(400);
  expect(res.body).toHaveProperty("error", "doctorId inválido");
});

test("GET /buscarchitas/:id devuelve 200 cuando existe, 404 cuando no", async () => {
  mockCitasService.buscarLasCitas.mockResolvedValueOnce({ id: 10 });
  let res = await request(app)
    .get("/apicitas/buscarchitas/10")
    .set("x-role", "doctor");
  expect(res.status).toBe(200);
  expect(res.body).toEqual({ id: 10 });

  mockCitasService.buscarLasCitas.mockResolvedValueOnce(null);
  res = await request(app)
    .get("/apicitas/buscarchitas/999")
    .set("x-role", "doctor");
  expect(res.status).toBe(404);
});
```




Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
test("POST /crearcitas por usuario sin tipo pone por defecto evaluacion", async () => {
  mockCitasService.crearLasCitas.mockImplementation(async (payload) => ({
    ...payload,
    id: 1,
    id_doctor: 3,
    tipo: payload.tipo,
  }));
  const body = { fecha: "2025-01-01T10:00:00Z" };
  const res = await request(app)
    .post("/apicitas/crearcitas")
    .set("x-role", "usuario")
    .send(body);
  expect(res.status).toBe(201);
  expect(mockCitasService.crearLasCitas).toHaveBeenCalled();
  const calledPayload = mockCitasService.crearLasCitas.mock.calls[0][0];
  expect(calledPayload.tipo).toBe("evaluacion");
  expect(calledPayload._rol_creador).toBe("usuario");
});

test("POST /crearcitas tipo inválido para doctor devuelve 400", async () => {
  const res = await request(app)
    .post("/apicitas/crearcitas")
    .set("x-role", "doctor")
    .send({ tipo: "otra", fecha: "2025-01-01T10:00:00Z" });
  expect(res.status).toBe(400);
  expect(res.body).toHaveProperty("message", "Tipo de cita inválido.");
});

test("PATCH /editarcitausuario/:id devuelve 200 cuando actualiza, 404 cuando no", async () => {
  mockCitasService.actualizarLasCitas.mockResolvedValueOnce([1]);
  let res = await request(app)
    .patch("/apicitas/editarcitausuario/5")
    .set("x-role", "usuario")
    .send({ fecha: "2025-01-02T12:00:00Z" });
  expect(res.status).toBe(200);
  expect(res.body).toHaveProperty("mensaje");

  mockCitasService.actualizarLasCitas.mockResolvedValueOnce([0]);
  res = await request(app)
    .patch("/apicitas/editarcitausuario/5")
    .set("x-role", "usuario")
    .send({ fecha: "2025-01-02T12:00:00Z" });
  expect(res.status).toBe(404);
});
```



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
describe("Routers de Citas - integración completa (controller + router)", () => {
  test("PATCH /cancelarcita/:id devuelve 404 si no existe, 200 si cancela", async () => {
    mockCitasService.buscarLasCitas.mockResolvedValueOnce(null);
    let res = await request(app)
      .patch("/apicitas/cancelarcita/123")
      .set("x-role", "usuario");
    expect(res.status).toBe(404);

    mockCitasService.buscarLasCitas.mockResolvedValueOnce({
      id: 11,
      id_doctor: 5,
    });
    mockCitasService.actualizarLasCitas.mockResolvedValueOnce([1]);
    res = await request(app)
      .patch("/apicitas/cancelarcita/11")
      .set("x-role", "usuario");
    expect(res.status).toBe(200);
    expect(mockCitasService.notificarTotalesMes).toHaveBeenCalledWith(5);
  });

  test("GET /horarios/:fecha mapea citas a duraciones", async () => {
    mockCitasService.obtenerCitasPorFecha.mockResolvedValueOnce([
      { id: 1, fecha: "2025-01-01T10:00:00Z", tipo: "evaluacion" },
      { id: 2, fecha: "2025-01-01T13:00:00Z", tipo: "procedimiento" },
    ]);
    const res = await request(app)
      .get("/apicitas/horarios/2025-01-01")
      .set("x-role", "doctor");
    expect(res.status).toBe(200);
    expect(res.body).toEqual([
      {
        id: 1,
        fecha: "2025-01-01T10:00:00Z",
        tipo: "evaluacion",
        duracion: 30,
      },
      {
        id: 2,

```

```
datevengnz@datevengnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estatica-Frontend-Backend/Backend$ npm test -- __tests__/CitasRouters.full.int.test.js
```

```
> clinestetica@1.0.0 test
```

```
> jest --passWithNoTests __tests__/CitasRouters.full.int.test.js
```


```
PASS   tests_/_CitasRouters.full.int.test.js
  Routers de Citas - integración completa (controller + router)
    ✓ GET /listarcitas como doctor llama al servicio con doctorId y devuelve lista (44 ms)
    ✓ GET /listarcitas como usuario requiere doctorId en query o devuelve [] (0 ms)
    ✓ GET /listarcitas doctorId inválido para usuario devuelve 400 (6 ms)
    ✓ GET /buscarcitas/:id devuelve 200 cuando existe, 404 cuando no (11 ms)
    ✓ POST /crearcitas por usuario sin tipo pone por defecto evaluacion (21 ms)
    ✓ POST /crearcitas tipo inválido para doctor devuelve 400 (7 ms)
    ✓ PATCH /editararcitausuario/:id devuelve 200 cuando actualiza, 404 cuando no (10 ms)
    ✓ PATCH /editararcita-doctor/:id actualiza y crea requerimientos (7 ms)
    ✓ PATCH /cancelarcita/:id devuelve 404 si no existe, 200 si cancela (10 ms)
    ✓ GET /horarios/:fecha mapea citas a duraciones (6 ms)
    ✓ PATCH /editararcita/:id valida id, 404 si no existe, 200 en caso contrario (14 ms)
    ✓ GET /orden-examenes/pdf/:id maneja 404, 400 y 200 (397 ms)
    ✓ GET /orden-medicamentos/pdf/:id maneja 404, 400 y 200 (210 ms)
    ✓ PATCH /marcar-examenes-subidos/:id devuelve 403 para no-usuario y 200 para usuario (9 ms)
    ✓ GET /miscitas devuelve lista para usuario autenticado (5 ms)
    ✓ POST /requerimientos crea requerimiento (doctor) (6 ms)
    ✓ PATCH /reagendarcita/:id actualiza cita usando service.reagendarCita (6 ms)
    ✓ GET /listarcitas todos usuarios de asistente devuelve lista para asistente (5 ms)
```

```
Test Suites: 1 passed, 1 total
Tests:       18 passed, 18 total
Snapshots:   0 total
Time:        2.742 s
```

```
nan all test suites matching /__tests_/_CitasRouters.full.int.test.js/i.
```

```
datevengnz@datevengnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estatica-Frontend-Backend/Backend$
```

17

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 17 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

6. Pruebas de integración — módulo de consentimientos

6.1 Propósito de prueba

Validar integralmente las rutas del módulo **Consentimientos** (/apiconsentimiento), incluyendo la obtención, creación y generación del PDF firmado.

El objetivo es asegurar que las rutas se comuniquen correctamente con los servicios, validen las entidades asociadas (usuario, cita, orden) y generen la URL firmada en Cloudinary.


6.2 Alcance y cobertura

Estas pruebas verifican la correcta comunicación entre el router Express, los servicios del consentimiento y las dependencias externas mockeadas (base de datos, Cloudinary).

Funcionalidad	Endpoint	Método	Casos probados	Resultado esperado
Listar consentimientos de un usuario autenticado	/apiconsentimiento/usuario	GET	Usuario con registros	200 OK + lista JSON
Crear un consentimiento	/apiconsentimiento/	POST	Cuerpo válido con id_cita y texto_terminos	201 Created
Generar PDF de consentimiento firmado	/apiconsentimiento/:id/pdf	GET	Verifica relaciones (cita, usuario, orden), genera hash y URL firmada	200 OK + { url: 'https://signed.url/file.pdf' }

6.3 Componentes involucrados

Componente	Rol en la prueba
Express.js	Crea el servidor y monta el router /apiconsentimiento.
Supertest	Ejecuta peticiones HTTP reales contra el servidor Express.
Authorization Middleware (mock)	Simula un usuario autenticado con rol (doctor, usuario).
ConsentimientoService	Simula la lógica de negocio (crear, listar, generar PDF).

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 18 de 36	
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario			

Componente	Rol en la prueba
(mock)	
Modelos (mock)	citás, ordenes, usuarios mockeados para validar relaciones.
Cloudinary (mock)	Genera una URL firmada temporal simulada para el PDF.

```
describe('Routers de Consentimientos - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apiconsentimiento', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('GET /usuario devuelve lista para el usuario', async () => {
    mockService.obtenerConsentimientosPorUsuario.mockResolvedValueOnce([
      { id: 1 }
    ]);
    const res = await request(app).get('/apiconsentimiento/usuario');
    expect(res.status).toBe(200);
    expect(res.body).toEqual([
      { id: 1 }
    ]);
  });


  test('POST / crea consentimiento y devuelve 201', async () => {
    mockService.crearConsentimiento.mockResolvedValueOnce({ id: 10 });
    const res = await request(app).post('/apiconsentimiento/').send({
      id_cita: 1, texto_terminos: 'ok'
    });
    expect(res.status).toBe(201);
  });

  test('GET /:id/pdf genera URL firmada', async () => {
    mockService.obtenerPorId.mockResolvedValueOnce({
      id: 1, id_cita: 2, id_usuario: 3
    });
    citas.findByPk.mockResolvedValueOnce({ id: 2, id_orden: 4 });
    usuarios.findByPk.mockResolvedValueOnce({ id: 3 });
    ordenes.findByPk = jest.fn().mockResolvedValueOnce({
      id: 4, procedimientos: []
    });
    mockService.generarConsentimientoPDF.mockResolvedValueOnce({
      publicId: 'abc', url: 'u', hash: 'h'
    });
    const res = await request(app).get('/apiconsentimiento/1/pdf').set(
      'x-role', 'doctor'
    );
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('url');
  });
});
```

```
dstevegnz@dstevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estatica-Frontend-Backend/Backend$ npm test -- -- __tests__/_ConsentimientoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests __tests__/_ConsentimientoRouters.full.int.test.js

PASS __tests__/_ConsentimientoRouters.full.int.test.js
  Routers de Consentimientos - Integración completa
    GET /usuario devuelve lista para el usuario (57 ms)
    POST / crea consentimiento y devuelve 201 (34 ms)
    GET /:id/pdf genera URL firmada (8 ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 1.16 s
Ran all test suites matching /__tests__/_ConsentimientoRouters.full.int.test.js/i.
dstevegnz@dstevegnz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estatica-Frontend-Backend/Backend$
```


	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 19 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

7. Pruebas de integración — módulo de contacto

Validar integralmente la ruta del módulo **Contacto**, garantizando que las solicitudes HTTP hacia /apicontacto/contacto sean correctamente procesadas por el controlador correspondiente y que retornen la estructura esperada.

El objetivo es comprobar la conexión entre el **router Express**, el **controlador ContactoControllers** y la respuesta del endpoint sin necesidad de acceder a servicios externos reales (correo, base de datos, etc.).

7.1 Propósito de la prueba

Funcionalidad	Endpoint	Método	Casos probados	Resultado esperado
Envío de mensaje desde el formulario de contacto	/apicontacto/contacto	POST	Cuerpo con nombre, correo y mensaje válidos	200 OK + { ok: true }

7.2 Componentes Involucrados

Componente	Rol
Express.js	Configura el servidor y monta el router /apicontacto.
Supertest	Simula solicitudes HTTP reales al servidor Express.
ContactoControllers (mock)	Emula el controlador enviar, retornando una respuesta JSON simulada.
Router de Contacto	Define la ruta /contacto y conecta con el controlador.


```
const express = require('express');
const request = require('supertest');

const mockCtrl = { enviar: jest.fn(async (_req, res) => res.json({ ok: true })) };
jest.mock('../controllers/ContactoControllers', () => mockCtrl);

const Router = require('../routers/ContactoRouters');

describe('Routers de Contacto - integración completa', () => {
  let app;
  beforeEach(() => {
    app = express();
    app.use(express.json());
    app.use('/apicontacto', Router);
  });
  afterEach(() => jest.clearAllMocks());

  test('POST /contacto devuelve ok', async () => {
    const res = await request(app).post('/apicontacto/contacto').send({ nombre: 'a', correo: 'a@a', mensaje: '' });
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('ok', true);
  });
});
```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 20 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

```

dsteven@steven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$ npm test -- _tests_/ContactoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests _tests_/ContactoRouters.full.int.test.js

PASS _tests_/ContactoRouters.full.int.test.js
  Router de Contacto - integración completa
    POST /contacto devuelve url (100 ms)

Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 1.856 s
Run all test suites matching / _tests_/ContactoRouters.full.int.test.js/
dsteven@steven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$

```

8. Pruebas de integración — módulo de exámenes

8.1 Alcance de prueba

Validar integralmente el funcionamiento de las rutas del módulo **Exámenes**, comprobando la comunicación entre las capas de **router**, **middlewares de autorización/subida de archivos**, y **servicios**.


Se busca garantizar que el flujo de carga, consulta, eliminación y descarga de exámenes asociados a citas médicas funcione de forma correcta y segura.

8.2 Alcance y cobertura

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Subir archivos de exámenes	/apiexamenes/subir/:id_cita	POST	Carga exitosa de archivos asociados a una cita	201 Created + { examenes: [...] }
Listar exámenes por cita	/apiexamenes/cita/:id_cita	GET	Retorna lista de exámenes del paciente	200 OK + [{ id: ... }]
Eliminar examen	/apiexamenes/:id	DELETE	Maneja correctamente el caso inexistente	404 Not Found
Descargar examen (con permisos)	/apiexamenes/descargar/:id	GET	Devuelve URL firmada del archivo	200 OK + { url: 'https://pdf.signed' }

8.3 Componentes Involucrados

Componente	Descripción
Express.js	Servidor HTTP utilizado para montar el router /apiexamenes.
Supertest	Biblioteca utilizada para simular solicitudes HTTP reales.
Authorization Middleware	Simula la autenticación y roles del usuario (doctor/usuario).

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 21 de 36
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025
	Elaborado por: Instructores área Software			
Nombre del Documento:		Formato para la construcción del Manual de Usuario		

Componente	Descripción
(mock) MulterExamenes Middleware (mock)	Intercepta la carga de archivos sin procesar archivos reales.
ExamenServices (mock)	Emula los métodos de servicio: subirArchivos, listarPorCita, eliminar.
Cloudinary (mock)	Simula la generación de URL firmadas para descargas seguras.

```

jest.mock('cloudinary', () => ({
  v2: { utils: {}, url: jest.fn(() => 'https://image.signed'),
    utils: { private_download_url: jest.fn(() => 'https://pdf.signed') } },
}));

const Router = require('../routers/ExamenRouters');
const { examen, citas } = require('../models');

describe('Routers de Exámenes - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apiexamenes', Router);
  });
  beforeEach(() => jest.clearAllMocks());


  test('POST /subir/:id cita devuelve 201 con datos', async () => {
    mockService.subirArchivos.mockResolvedValueOnce({ id: 1 });
    const res = await request(app).post('/apiexamenes/subir/3');
    expect(res.status).toBe(201);
    expect(res.body).toHaveProperty('examenes');
  });

  test('GET /cita/:id cita devuelve lista', async () => {
    mockService.listarPorCita.mockResolvedValueOnce({ id: 2 });
    const res = await request(app).get('/apiexamenes/cita/9');
    expect(res.status).toBe(200);
  });

  test('DELETE /:id devuelve 404 si no existe', async () => {
    mockService.eliminar.mockResolvedValueOnce(false);
    const res = await request(app).delete('/apiexamenes/1');
    expect(res.status).toBe(404);
  });

  test('GET /descargar/:id devuelve URL firmada cuando está autorizado', async () => {
    examen.findByPk.mockResolvedValueOnce({ id: 1, id_cita: 5, archivo_examen: 'file.pdf' });
    citas.findByPk.mockResolvedValueOnce({ id: 5, id_doctor: 1, id_usuario: 2 });
    const res = await request(app).get('/apiexamenes/descargar/1').set('x-role', 'doctor');
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty('url');
  });
});

```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 22 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

9. Pruebas de integración — módulo de historial médico


9.1 Alcance de prueba

Verificar integralmente las rutas del módulo **Historial Médico**, asegurando que el flujo entre **router**, **middlewares de autorización**, y **servicios clínicos** funcione correctamente.


Se validan los accesos por rol, la restricción al historial propio del usuario y las operaciones CRUD con sus respuestas esperadas.

9.2 Alacence y cobertura

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Listar historiales clínicos	/apihistorial/listarhistorialclinico	GET	Listado general para asistente o doctor	200 OK + [{ id: ... }]
Buscar historial por ID	/apihistorial/buscarhistorialclinico/:id	GET	Retorna historial existente o 404 si no existe	200 / 404
Buscar historial por usuario	/apihistorial/buscarhistorialclinicoporusuario/:id	GET	Retorna historial asociado a un usuario	200 / 404
Ver historial propio	/apihistorial/mihistorialclinico/:id	GET	Permite acceso solo si el usuario autenticado es el propietario	200 OK / 403 Forbidden
Crear historial clínico	/apihistorial/crearhistorialclinico	POST	Crea un nuevo historial (rol: doctor)	201 Created
Editar historial clínico	/apihistorial/editarhistorialclinico/:id	PATCH	Valida ID, retorna 400/404/200	400 / 404 / 200

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 23 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Eliminar historial clínico	/apihistorial/eliminarhistorialclinico/:id	DELETE	00 según caso Elimina registro existente	200 OK

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 24 de 36
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025
	Elaborado por: Instructores área Software			
Nombre del Documento:		Formato para la construcción del Manual de Usuario		


9.3 Componentes Involucrados

Componente	Descripción
Express.js	Framework utilizado para montar el router /apihistorial.
Supertest	Herramienta usada para simular peticiones HTTP reales.
Authorization Middleware (mock)	Simula los permisos según rol (doctor, asistente, usuario) y usuario autenticado.
HistorialMedicoServices (mock)	Emula las operaciones CRUD: listarLosHistorialesClinicos, buscarLosHistorialesClinicos, crearLosHistorialesClinicos, etc.
Jest	Framework de pruebas que permite aislar dependencias mediante mocks.

```

Proyecto-Sena-Clinica-Estetica-Frontend-Backend > Backend > __tests__ > HistorialMedicoRoutes.full.int.test.js > ...
39 describe("Routers de Historial Médico - integración completa", () => {
49   test("GET /listarhistorialclinico devuelve lista", async () => {
51     const res = await request(app)
52       .get("/apihistorial/listarhistorialclinico")
53       .set("x-role", "asistente");
54     expect(res.status).toBe(200);
55     expect(res.body).toEqual([ { id: 1 } ]);
56   });
57
58   test("GET /buscarhistorialclinico/:id devuelve 200/404", async () => {
59     mockHistService.buscarLosHistorialesClinicos.mockResolvedValueOnce({ id: 9 });
60     let res = await request(app)
61       .get("/apihistorial/buscarhistorialclinico/9")
62       .set("x-role", "doctor");
63     expect(res.status).toBe(200);
64
65     mockHistService.buscarLosHistorialesClinicos.mockResolvedValueOnce(null);
66     res = await request(app)
67       .get("/apihistorial/buscarhistorialclinico/9")
68       .set("x-role", "doctor");
69     expect(res.status).toBe(404);
70   });
71
72   test("GET /buscarhistorialclinicoporusuario/:id devuelve 200/404", async () => {
73     mockHistService.buscarLosHistorialesClinicosPorUsuario.mockResolvedValueOnce({ id: 4 });
74     let res = await request(app)
75       .get("/apihistorial/buscarhistorialclinicoporusuario/4")
76       .set("x-role", "doctor");
77     expect(res.status).toBe(200);
78
79     mockHistService.buscarLosHistorialesClinicosPorUsuario.mockResolvedValueOnce(null);
80     res = await request(app)
81       .get("/apihistorial/buscarhistorialclinicoporusuario/4")
82       .set("x-role", "doctor");
83     expect(res.status).toBe(404);
84   });
85

```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 25 de 36
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025
	Elaborado por: Instructores área Software			
Nombre del Documento:		Formato para la construcción del Manual de Usuario		

```

dstevengez@dstevengez:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$ npm test -- __tests__/HistorialMedicoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests __tests__/HistorialMedicoRouters.full.int.test.js

PASS __tests__/HistorialMedicoRouters.full.int.test.js
Routers de Historial Médico - integración completa
  ✓ GET /listarhistorialclinico devuelve lista (56 ms)
  ✓ GET /buscarnhistorialclinico/:id devuelve 200/404 (25 ms)
  ✓ GET /buscarnhistorialclinico/:id devuelve 200/404 (10 ms)
  ✓ GET /mihistorialclinico/:id exige propietario (10 ms)
  ✓ POST /crearhistorialclinico devuelve 201 (21 ms)
  ✓ PATCH /editarhistorialclinico/:id valida id y no encontrado (10 ms)
  ✓ DELETE /eliminarhistorialclinico/:id devuelve 200 (5 ms)

Test Suites: 1 passed, 1 total
Tests: 7 passed, 7 total
Snapshots: 0 total
Time: 1.189 s
Ran all test suites matching / __tests__/HistorialMedicoRouters.full.int.test.js/i.
dstevengez@dstevengez:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$

```

10. Pruebas de integración — módulo de órdenes

10.1 Proposito de prueba

Validar integralmente el comportamiento de las rutas del módulo **Órdenes**, comprobando la correcta comunicación entre el **Router Express**, los **middlewares de autenticación/autorización** y la capa de **servicios de órdenes** (OrdenServices).


Se evalúan las operaciones principales de consulta y edición de órdenes médicas asociadas a usuarios autenticados.

10.2 Alcance y cobertura

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Listar órdenes por usuario autenticado	/apiordenes/misordenes	GET	Devuelve todas las órdenes del usuario logueado	200 OK + [{ id: ... }]
Listar órdenes elegibles (evaluaciones realizadas)	/apiordenes/elegibles/:usuarioid	GET	Valida parámetro usuarioid; devuelve 400 si no numérico y 200 si correcto	400 / 200 OK
Editar una orden	/apiordenes/editarordenes/:id	PATCH	Valida ID numérico y existencia de la orden (no actualiza = 404)	400 / 404

10.3 Componentes involucrados

Componente	Descripción
Express.js	Framework usado para montar el router /apiordenes.
Supertest	Simula peticiones HTTP reales para verificar los endpoints.
Authorization Middleware	Simula la autenticación del usuario (con rol "usuario" y id:7).

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 26 de 36
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025
	Elaborado por: Instructores área Software			
Nombre del Documento:		Formato para la construcción del Manual de Usuario		

Componente	Descripción
(mock)	
OrdenServices (mock)	Provee respuestas controladas para los métodos: listarOrdenesPorUsuario, listarOrdenesEvaluacionRealizadaPorUsuario, actualizarLasOrdenes.
Jest	Framework de testing que permite aislar dependencias y automatizar pruebas.


```
describe('Routers de Ordenes - integración completa', () => {
  let app;
  beforeEach(() => {
    app = express();
    app.use(express.json());
    app.use('/apiordenes', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('GET /misordenes devuelve órdenes del usuario', async () => {
    mockService.listarOrdenesPorUsuario.mockResolvedValueOnce([
      { id: 1 }
    ]);
    const res = await request(app).get('/apiordenes/misordenes');
    expect(res.status).toBe(200);
    expect(mockService.listarOrdenesPorUsuario).toHaveBeenCalled();
  });

  test('GET /elegibles/:usuarioId valida id', async () => {
    let res = await request(app).get('/apiordenes/elegibles/abc');
    expect(res.status).toBe(400);
    mockService.listarOrdenesEvaluacionRealizadaPorUsuario.mockResolvedValueOnce([
      { id: 2 }
    ]);
    res = await request(app).get('/apiordenes/elegibles/9');
    expect(res.status).toBe(200);
    expect(res.body).toEqual([
      { id: 2 }
    ]);
  });

  test('PATCH /editarordenes/:id valida id (400) y no encontrado (404)', async () => {
    // 400 por ID inválido
    let res = await request(app).patch('/apiordenes/editarordenes/abc').send({});
    expect(res.status).toBe(400);

    // 404 cuando service devuelve []
    mockService.actualizarLasOrdenes.mockResolvedValueOnce([
      {}
    ]);
    res = await request(app).patch('/apiordenes/editarordenes/3').send({});
    expect(res.status).toBe(404);
  });
});
```


	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 27 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

```

jsteven@jsteven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- --tests __tests__/OrdenesRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests __tests__/OrdenesRouters.full.int.test.js

PASS __tests__/OrdenesRouters.full.int.test.js
  Routers de Órdenes - integración completa
    ✓ GET /misordenes devuelve órdenes del usuario (43 ms)
    ✓ GET /elegibles/usuarioId valido id (14 ms)
    ✓ PATCH /editarordenes/:id valida id (400) y no encontrado (404) (29 ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 1.126 s
Ran all test suites matching /__tests__/OrdenesRouters.full.int.test.js/i
jsteven@jsteven:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$

```

11. Pruebas de integracion orden procedimiento

11.1 Proposito de prueba

Verificar el funcionamiento integral de las rutas del módulo **Órdenes de Procedimiento**, garantizando la correcta comunicación entre el **Router Express**, los **middlewares de autorización** y los métodos del **servicio de órdenes de procedimiento**.


El objetivo es asegurar que las operaciones de listado, búsqueda y edición respondan con los códigos HTTP adecuados y gestionen correctamente los casos de error.

11.2 Alcance y cobertura

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Listar todas las órdenes de procedimiento	/apiordenprocedimiento/listarordenes	GET	Devuelve la lista general de órdenes	200 OK
Buscar una orden específica	/apiordenprocedimiento/buscarordenes/:id	GET	Retorna 404 Not Found cuando la orden no existe	404 Not Found
Editar una orden (validación de ID)	/apiordenprocedimiento/editarordenes/:id	PATCH	Retorna 400 Bad Request cuando el ID es inválido (no numérico)	400 Bad Request

11.3 Componentes Involucrados

Componente	Descripción
Express.js	Framework base para el enrutamiento y manejo de peticiones.

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 28 de 36
	Número de Documento:	FS-DOC Usuario	Formato Manual de	Fecha de Creación: 1/02/2025
	Elaborado por: Instructores área Software			
Nombre del Documento:		Formato para la construcción del Manual de Usuario		

Componente	Descripción
Supertest	Simula solicitudes HTTP reales para verificar las rutas.
Authorization Middleware (mock)	Simula autenticación/autorización básica sin aplicar restricciones de rol.
OrdenProcedimientoServices (mock)	Servicio simulado que responde con datos de prueba para los métodos CRUD.
Jest	Framework de testing encargado de la ejecución de las pruebas e aislamiento de dependencias.


```
const mockService = {
  listarLasOrdenesProcedimientos: jest.fn(),
  buscarLasOrdenesProcedimientos: jest.fn(),
  crearLasOrdenesProcedimientos: jest.fn(),
  actualizarLasOrdenesProcedimientos: jest.fn(),
  eliminarLasOrdenesProcedimientos: jest.fn(),
};
jest.mock('../services/OrdenProcedimientoServices', () => mockService);
const Router = require('../routers/OrdenProcedimientoRouters');

describe('Routers de Órdenes de Procedimiento - integración completa', () => {
  let app;
  beforeAll(() => {
    app = express();
    app.use(express.json());
    app.use('/apiordenprocedimiento', Router);
  });
  beforeEach(() => jest.clearAllMocks());

  test('GET /listarordenes devuelve lista', async () => {
    mockService.listarLasOrdenesProcedimientos.mockResolvedValueOnce([
      { id: 1 }
    ]);
    const res = await request(app).get('/apiordenprocedimiento/listarordenes');
    expect(res.status).toBe(200);
  });

  test('GET /buscarordenes/:id devuelve 404 cuando no existe', async () => {
    mockService.buscarLasOrdenesProcedimientos.mockResolvedValueOnce(null);
    const res = await request(app).get('/apiordenprocedimiento/buscarordenes/9');
    expect(res.status).toBe(404);
  });

  test('PATCH /editarordenes/:id id inválido devuelve 400', async () => {
    const res = await request(app).patch('/apiordenprocedimiento/editarordenes/abc').send({});
    expect(res.status).toBe(400);
  });
});
```

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 29 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

```

dsteven@zjds:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- --_tests_/OrdenProcedimientoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_/OrdenProcedimientoRouters.full.int.test.js

PASS Tests _OrdenProcedimientoRouters.full.int.test.js
  Routers de Ordenes de Procedimiento - integración completa
    ✓ GET /listarordenes devuelve lista (47 ms)
    ✓ GET /buscarordenes/:id devuelve 404 cuando no existe (12 ms)
    ✓ PATCH /editarordenes/:id id inválido devuelve 400 (30 ms)

Test Suites: 1 passed, 1 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 1.006 s
Ran all test suites matching /_tests_/OrdenProcedimientoRouters.full.int.test.js/i.
dsteven@zjds:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$

```

12. Pruebas de integración — módulo de procedimientos


12.1 Propósito de prueba

Evaluar de forma integral las rutas del módulo **Procedimientos**, garantizando que las peticiones HTTP (GET, POST, PATCH, DELETE) gestionen correctamente las operaciones principales sobre los procedimientos médicos/estéticos y sus archivos asociados (imágenes, descripciones, etc.).

El objetivo es confirmar que las rutas se comunican correctamente con los servicios y devuelven los códigos HTTP apropiados (**200, 201, 400, 404**) según el caso.

12.2 Alcance y cobertura

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Listado general de procedimientos	/apiprocedimientos/listarprocedimiento	GET	Devuelve lista completa de procedimientos	200 OK
Listado filtrado por categoría	/apiprocedimientos/categorias/:categoriald/procedimientos	GET	Devuelve procedimientos asociados a una categoría específica	200 OK
Búsqueda individual	/apiprocedimientos/buscarprocedimiento/:id	GET	Devuelve procedimiento existente (200) o 404 si no existe	200 / 404
Creación de procedimiento	/apiprocedimientos/crearprocedimiento	POST	Crea nuevo procedimiento con validación de datos y archivos	201 Created

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 30 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Funcionalidad	Endpoint	Método	Caso validado	Resultado esperado
Edición de procedimiento	/apiprocedimientos/editarprocedimiento/:id	PATCH	Valida ID, verifica existencia, actualiza información	400 / 404 / 200
Eliminación	/apiprocedimientos/eliminarprocedimiento/:id	DELETE	Elimina procedimiento por ID	200 OK

12.3 Componentes Técnicos Involucrados

Componente	Descripción
Express.js	Framework HTTP usado para definir y montar rutas.
Supertest	Herramienta para simular solicitudes HTTP reales sobre el servidor Express.
Jest	Framework de testing que ejecuta los escenarios y aísla dependencias.
Multer (mock)	Middleware simulado para manejar subida de imágenes y archivos.
Authorization (mock)	Middleware simulado que añade un usuario autenticado en cada prueba.
ProcedimientosServices (mock)	Servicio que emula las respuestas de la capa lógica de negocio.
Cloudinary (mock)	Proveedor de almacenamiento de archivos, simulado para evitar dependencias externas.



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por: Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
describe("Routers de Procedimientos - integración completa", () => {
  test("GET /buscarprocedimiento/:id devuelve 200/404", async () => {
    mockProcService.buscarLosProcedimientos.mockResolvedValueOnce(null);
    res = await request(app).get("/apiprocedimientos/buscarprocedimiento/9");
    expect(res.status).toBe(404);
  });

  test("POST /crearprocedimiento crea con archivos y devuelve 201", async () => {
    mockProcService.crearLosProcedimientos.mockResolvedValueOnce({ id: 5, nombre: "Proc" });
    const res = await request(app)
      .post("/apiprocedimientos/crearprocedimiento")
      .set("x-role", "doctor")
      .send({ nombre: "Proc", precio: "100", duracion: "60", requiere_evaluacion: "true", categoriaId: "1" });
    expect(res.status).toBe(201);
    expect(mockProcService.crearLosProcedimientos).toHaveBeenCalled();
  });

  test("PATCH /editarprocedimiento/:id maneja 400 id inválido, 404 no encontrado y 200 ok", async () => {
    let res = await request(app)
      .patch("/apiprocedimientos/editarprocedimiento/abc")
      .set("x-role", "doctor")
      .send({ nombre: "X", precio: "10", duracion: "30" });
    expect(res.status).toBe(400);

    mockProcService.buscarLosProcedimientos.mockResolvedValueOnce(null);
    res = await request(app)
      .patch("/apiprocedimientos/editarprocedimiento/7")
      .set("x-role", "doctor")
      .send({ nombre: "X", precio: "10", duracion: "30" });
    expect(res.status).toBe(404);


    mockProcService.buscarLosProcedimientos.mockResolvedValueOnce({ id: 7, imagen: null });
    mockProcService.actualizarLosProcedimientos.mockResolvedValueOnce([1]);
    res = await request(app)
      .patch("/apiprocedimientos/editarprocedimiento/7")
      .set("x-role", "doctor")
      .send({ nombre: "X", precio: "10", duracion: "30" });
    expect(res.status).toBe(200);
    expect(res.body).toHaveProperty("mensaje");
  });

  test("DELETE /eliminarprocedimiento/:id devuelve 200", async () => {
    const res = await request(app)
      .delete("/apiprocedimientos/eliminarprocedimiento/3")
      .set("x-role", "doctor");
    expect(res.status).toBe(200);
  });
});
```

```
dstevegnz@dstevegnz: ~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$ npm test -- --_tests_/ProcedimientoRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_/ProcedimientoRouters.full.int.test.js

PASS _tests_/ProcedimientoRouters.full.int.test.js
Routers de Procedimientos - Integración completa
  ✓ GET /listarprocedimiento devuelve lista (64 ms)
  ✓ GET /categorias/:categoriaId/procedimientos filtra por categoria (23 ms)
  ✓ GET /buscarprocedimiento/:id devuelve 200/404 (10 ms)
  ✓ POST /crearprocedimiento crea con archivos y devuelve 201 (24 ms)
  ✓ PATCH /editarprocedimiento/:id maneja 400 id inválido, 404 no encontrado y 200 ok (10 ms)
  ✓ DELETE /eliminarprocedimiento/:id devuelve 200 (8 ms)

Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 1.795 s
Ran all test suites matching /_tests_/ProcedimientoRouters.full.int.test.js/i.
dstevegnz@dstevegnz: ~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Eстетica-Frontend-Backend/Backend$
```


	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 32 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

13. Pruebas de integración — módulo de usuarios

13.1 Propósito de prueba


Validar de forma integral el funcionamiento del **Router de Usuarios**, abarcando todos los flujos principales del módulo de autenticación y gestión de usuarios:

- registro y confirmación,
- autenticación (login),
- actualización de datos y estado,
- eliminación,
- manejo de notificaciones (doctor y usuario).

El objetivo es confirmar que las rutas se comunican correctamente con los controladores y servicios, devolviendo las respuestas adecuadas y códigos HTTP correctos (**200, 201, 400, 401, 404**).

13.2 Alcance y cobertura

Funcionalidad	Endpoint	Método	Casos probados	Códigos esperados
Listar usuarios	/apiusuarios/listarusuarios	GET	Devuelve lista completa de usuarios	200
Buscar usuario	/apiusuarios/buscarusuarios/:id	GET	Usuario existente / no existente	200 / 404
Pre-registro	/apiusuarios/preregistro	POST	Datos válidos y faltantes	200 / 400
Confirmar registro	/apiusuarios/preregistro/confirmar	POST	Código válido / inválido	200 / 400
Crear usuario normal	/apiusuarios/crearusuarios	POST	Datos completos válidos	201
Crear usuario admin	/apiusuarios/crearusuariosadmin	POST	Datos completos válidos con rol	201
Editar usuario	/apiusuarios/editarusuarios/:id	PATCH	ID válido / inválido / no encontrado	200 / 400 / 404
Eliminar usuario	/apiusuarios/eliminarusuarios/:id	DELETE	Eliminación simple	200
Iniciar sesión	/apiusuarios/iniciarsesion	POST	Credenciales	200 / 401

	Centro de Teleinformática y Producción Industrial - Regional Cauca			Página 33 de 36
	Número de Documento:	FS-DOC Formato Manual de Usuario	Fecha de Creación: 1/02/2025	Elaborado por: Instructores área Software
	Nombre del Documento:	Formato para la construcción del Manual de Usuario		

Funcionalidad	Endpoint	Método	Casos probados	Códigos esperados
Cambiar estado usuario	/apiusuarios/editarestadousuario/:id	PATCH	válidas / inválidas Estado no booleano / correcto / no encontrado	400 / 200 / 401
Notificaciones (doctor)	/apiusuarios/notificaciones/...	GET/PATCH	Obtener, marcar una/todas, archivar, historial	200 / 400
Notificaciones (usuario)	/apiusuarios/notificacionesusuario/...	GET/PATCH	Obtener, marcar una/todas, archivar, historial	200 / 400

13.3 Componentes técnicos involucrados

Componente	Descripción
Express.js	Framework para la creación de rutas HTTP.
Supertest	Simula solicitudes reales a las rutas Express.
Jest	Framework de pruebas unitarias e integración.
Authorization (mock)	Middleware simulado para autenticación y roles.
Validaciones (mock)	Simula los middlewares de validación de campos.
UsuariosServices (mock)	Capa de servicios simulada con respuestas controladas.
Intentos fallidos (mock)	Middleware simulado para bloqueo temporal en login.



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área
Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
Proyecto-Sena-Clinica-Estetica-Frontend-Backend > Backend > _tests_ > UsuariosRouters.full.int.test.js > ...
62 describe('Routers de Usuarios - integración completa (router + controller)', () => {
173
174   test('POST /iniciarsesion -> 401 cuando error', async () => {
175     const res = await request(app).post('/apiusuarios/iniciarsesion').send({ correo: 'bad@x.com', contrasena: '1' });
176     expect(res.status).toBe(401);
177   });
178
179   test('PATCH /editarestadousuario/:id -> 400 requiere booleano', async () => {
180     const res = await request(app).patch('/apiusuarios/editarestadousuario/1').send({ estado: 'x' });
181     expect(res.status).toBe(400);
182   });
183
184   test('PATCH /editarestadousuario/:id -> 200 ok', async () => {
185     const res = await request(app).patch('/apiusuarios/editarestadousuario/1').send({ estado: true });
186     expect(res.status).toBe(200);
187     expect(res.body).toHaveProperty('mensaje');
188   });
189
190   test('PATCH /editarestadousuario/:id -> 401 no encontrado', async () => {
191     const res = await request(app).patch('/apiusuarios/editarestadousuario/999').send({ estado: true });
192     expect(res.status).toBe(401);
193   });
194
195   test('GET /notificaciones/:id -> 200 lista', async () => {
196     const res = await request(app).get('/apiusuarios/notificaciones/1');
197     expect(res.status).toBe(200);
198     expect(res.body).toEqual([ { id: 1 } ]);
199   });
200
201   test('PATCH /notificaciones/:id/marcar-leida -> 200', async () => {
202     const res = await request(app).patch('/apiusuarios/notificaciones/1/marcar-leida').send({ notificacionId: 1 });
203     expect(res.status).toBe(200);
204   });
205
206   test('PATCH /notificaciones/:id/marcar-leida -> 400 cuando falta id', async () => {
207     const res = await request(app).patch('/apiusuarios/notificaciones/1/marcar-leida').send({});
208     expect(res.status).toBe(400);
209   });
210
211   test('PATCH /notificaciones/:id/marcar-todas-leidas -> 200', async () => {
212     const res = await request(app).patch('/apiusuarios/notificaciones/1/marcar-todas-leidas').send({});
213     expect(res.status).toBe(200);
214   });
215
216   test('PATCH /notificaciones/:id/archivar-leidas -> 200', async () => {
217     const res = await request(app).patch('/apiusuarios/notificaciones/1/archivar-leidas').send({});
218     expect(res.status).toBe(200);
219     expect(res.body).toHaveProperty('archivadas', 2);
220   });
221 }
```



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
Proyecto-Sena-Clinica-Estetica-Frontend-Backend > Backend > _tests_ > UsuariosRouters.fullInt.test.js > ...
62 describe('Routers de Usuarios - integración completa (router + controller)', () => {
215
216   test('PATCH /notificaciones/:id/archivar-leidas -> 200', async () => {
217     const res = await request(app).patch('/apiusuarios/notificaciones/1/archivar-leidas').send({});
218     expect(res.status).toBe(200);
219     expect(res.body).toHaveProperty('archivadas', 2);
220   });
221
222   test('GET /notificaciones/:id/historial -> 200', async () => {
223     const res = await request(app).get('/apiusuarios/notificaciones/1/historial');
224     expect(res.status).toBe(200);
225     expect(res.body).toEqual([ { id: 1 } ]);
226   });
227
228   // Usuario notifications
229   test('GET /notificacionesusuario/:id -> 200 lista', async () => {
230     const res = await request(app).get('/apiusuarios/notificacionesusuario/1');
231     expect(res.status).toBe(200);
232     expect(res.body).toEqual([ { id: 2 } ]);
233   });
234
235   test('PATCH /notificacionesusuario/:id/marcar-leida -> 200', async () => {
236     const res = await request(app).patch('/apiusuarios/notificacionesusuario/1/marcar-leida').send({ notificacionId: 1 });
237     expect(res.status).toBe(200);
238   });
239
240   test('PATCH /notificacionesusuario/:id/marcar-leida -> 400 falta id', async () => {
241     const res = await request(app).patch('/apiusuarios/notificacionesusuario/1/marcar-leida').send({});
242     expect(res.status).toBe(400);
243   });
244
245   test('PATCH /notificacionesusuario/:id/marcar-todas-leidas -> 200', async () => {
246     const res = await request(app).patch('/apiusuarios/notificacionesusuario/1/marcar-todas-leidas').send({});
247     expect(res.status).toBe(200);
248   });
249
250   test('PATCH /notificacionesusuario/:id/archivar-leidas -> 200', async () => {
251     const res = await request(app).patch('/apiusuarios/notificacionesusuario/1/archivar-leidas').send({});
252     expect(res.status).toBe(200);
253     expect(res.body).toHaveProperty('archivadas', 3);
254   });
255
256   test('GET /notificacionesusuario/:id/historial -> 200', async () => {
257     const res = await request(app).get('/apiusuarios/notificacionesusuario/1/historial');
258     expect(res.status).toBe(200);
259     expect(res.body).toEqual([ { id: 9 } ]);
260   });
261 }
```



Número de Documento:

FS-DOC Formato Manual de Usuario

Fecha de Creación: 1/02/2025

Elaborado por:
Instructores área Software

Nombre del Documento:

Formato para la construcción del Manual de Usuario

```
datevengaz@datevengaz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend/Backend$ npm test -- --_tests_/UsuariosRouters.full.int.test.js
> clinestetica@1.0.0 test
> jest --passWithNoTests --_tests_/UsuariosRouters.full.int.test.js

PASS _tests_/UsuariosRouters.full.int.test.js
  Routers de Usuarios - Integración completa (router + controller)
    ✓ GET /listarusuarios -> 200 lista (51 ms)
    ✓ GET /buscarusuarios/id -> 200 encontrado (15 ms)
    ✓ GET /buscarusuarios/id -> 404 no encontrado (18 ms)
    ✓ POST /preregistro -> 200 éxito (datos completos y válidos) (30 ms)
    ✓ POST /preregistro -> 400 error de validación (9 ms)
    ✓ POST /preregistro/confirmar -> 200 cuando ok (11 ms)
    ✓ POST /preregistro/confirmar -> 400 cuando falla verificación (8 ms)
    ✓ POST /crearusuarios -> 201 creado (datos completos y válidos) (8 ms)
    ✓ POST /crearusuariosadmin -> 201 creado (datos completos y válidos + rol) (12 ms)
    ✓ PATCH /editarusuarios/id -> 200 actualizado (6 ms)
    ✓ PATCH /editarusuarios/id -> 400 id inválido (9 ms)
    ✓ PATCH /editarusuarios/id -> 404 no encontrado (6 ms)
    ✓ DELETE /eliminarusuarios/id -> 200 (6 ms)
    ✓ POST /iniciarsesion -> 200 con token (6 ms)
    ✓ POST /iniciarsesion -> 401 cuando error (8 ms)
    ✓ PATCH /editarestadousuario/id -> 400 requiere booleano (6 ms)
    ✓ PATCH /editarestadousuario/id -> 200 ok (8 ms)
    ✓ PATCH /editarestadousuario/id -> 401 no encontrado (6 ms)
    ✓ GET /notificaciones/id -> 200 lista (9 ms)
    ✓ PATCH /notificaciones/id/marcar-leida -> 200 (8 ms)
    ✓ PATCH /notificaciones/id/marcar-leida -> 400 cuando falta id (11 ms)
    ✓ PATCH /notificaciones/id/marcar-todas-leidas -> 200 (5 ms)
    ✓ PATCH /notificaciones/id/archivar-leidas -> 200 (9 ms)
    ✓ GET /notificaciones/id/historial -> 200 (14 ms)
    ✓ GET /notificacionesusuario/id -> 200 lista (30 ms)
    ✓ PATCH /notificacionesusuario/id/marcar-leido -> 200 (12 ms)
    ✓ PATCH /notificacionesusuario/id/marcar-leido -> 400 falta id (18 ms)
    ✓ PATCH /notificacionesusuario/id/marcar-todas-leidas -> 200 (10 ms)
    ✓ PATCH /notificacionesusuario/id/archivar-leidas -> 200 (11 ms)
    ✓ GET /notificacionesusuario/id/historial -> 200 (9 ms)

Test Suites: 1 passed, 1 total
Tests: 34 passed, 34 total
Snapshots: 0 total
Time: 1.522 s, estimated 2 s
Ran all test suites matching /_tests_/UsuariosRouters.full.int.test.js/i.
datevengaz@datevengaz:~/Escritorio/Clinestetica/Proyecto-Sena-Clinica-Estetica-Frontend-Backend$
```