Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
3.0	Requerimiento del Proyecto

Detalle de la prueba

Fecha de realización: **02-10-2025**	**Duración de la prueba:** **(8 minutos)**		
Requerimiento Funcional de la prueba	Validar que el método GetAll del CertificadoController liste correctamente todos los certificados y maneje adecuadamente los posibles errores durante la obtención de datos.		
Objetivo	Conocer si se muestran los certificados		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Código asignado del certificado		
Procedimiento de Prueba	**Prueba 1:** **Paso 1**: Se mockea el módulo certificadoService con Jest para simular sus métodos (GetAll, Create, Update, Delete, GetForId). Se limpian los mocks antes de cada prueba (jest.clearAllMocks()). Se inicializan los objetos req y res para simular la solicitud HTTP. Configuración del mock del servicio: Se define que CertificadoService.GetAll retorne de forma simulada (mockResolvedValue) la lista de certificados. Ejecución: Se llama al método CertificadoController.GetAll(req, res). . **Resultado**: Muestra el mensaje de que todos los certificados se listan correctamente .		
Datos de salida - Resultado Esperado	Muestra la lista de todos los certificados creados		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(X)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo del resultado de la prueba 1

```
> proyecto-funed-backend@1.0.0 test
> jest

 PASS  tests/unit/controllers/certificadoController.test.js
  CertificadoController
    GetAll
      √ debería listar todos los certificados correctamente (3 ms)
      √ debería manejar errores al listar certificados (2 ms)
    Create
      √ debería crear un certificado correctamente (1 ms)
      √ debería manejar errores al crear un certificado (1 ms)
    Update
      √ debería actualizar un certificado correctamente (1 ms)
      √ debería manejar errores al actualizar un certificado (1 ms)
    Delete
      √ debería eliminar un certificado correctamente (1 ms)
      √ debería manejar errores al eliminar un certificado
    GetForId
      √ debería obtener un certificado por ID correctamente (2 ms)
      √ debería manejar errores al obtener un certificado por ID (1 ms)
```

```
1  const CertificadoController = require('../../../controller/certificadoController');
2  const CertificadoService = require('../../../services/certificadoService');
3
4  jest.mock('../../../services/certificadoService', () => ({
5    GetAll: jest.fn(),
6    Create: jest.fn(),
7    Update: jest.fn(),
8    Delete: jest.fn(),
9    GetForId: jest.fn()
10 }));
11
12 describe('CertificadoController', () => {
13   let req, res;
14
15   beforeEach(() => {
16     jest.clearAllMocks();
17     req = { body: {}, params: {} };
18     res = { json: jest.fn().mockReturnThis(), status: jest.fn().mockReturnThis() };
19   });
20
21   describe('GetAll', () => {
22     test('debería listar todos los certificados correctamente', async () => {
23       // Datos de prueba
24       const certificados = [
25         { id: 1, id_curso_matriculado: 1, fecha_emision: '2023-05-15', url_certificado: 'url1' },
26         { id: 2, id_curso_matriculado: 2, fecha_emision: '2023-05-16', url_certificado: 'url2' }
27       ];
28
29       // Configurar el mock del servicio
30       CertificadoService.GetAll.mockResolvedValue(certificados);
31
32       // Ejecutar el método del controlador
33       await CertificadoController.GetAll(req, res);
34
35       // Verificar que se llamó al servicio
36       expect(CertificadoService.GetAll).toHaveBeenCalled();
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Realizado por	Firma	Fecha
Santiago Cerón		**02-10-2025**
Aprobado por	**Firma**	**Fecha**

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
3.0	Requerimiento del Proyecto

Detalle de la prueba

Fecha de realización: **02-10-2025**	**Duración de la prueba:** **(5 minutos)**		
Requerimiento Funcional de la prueba	Verificar que el método GetAll del PagoController sea capaz de listar correctamente todos los pagos almacenados, utilizando el servicio correspondiente, y que maneje adecuadamente los errores en caso de que ocurra alguna excepción durante el proceso		
Objetivo	Obtener la lista de los pagos registrados		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	No tiene		
Procedimiento de Prueba	**Prueba 1:** Se crea un mock del módulo PagoService con Jest, incluyendo todos sus métodos (GetAll, Create, Update, Delete, GetForId). Se limpian los mocks antes de cada prueba (jest.clearAllMocks()). Se inicializan los objetos req y res para simular una solicitud HTTP. Configuración del mock del servicio: Se configura PagoService.GetAll para que devuelva (mockResolvedValue) el arreglo de pagos de prueba. **Resultado**: Lista de todas las solicitudes de pago registrados en el sistema en formato json		
Datos de salida - Resultado Esperado	Lista los pagos correctamente registrados		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(X)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Pantallazo de la prueba

```
PASS tests/unit/controllers/pagoController.test.js
  PagoController
    GetAll
      √ debería listar todos los pagos correctamente (1 ms)
      √ debería manejar errores al listar pagos (1 ms)
    Create
      √ debería crear un pago correctamente (1 ms)
      √ debería manejar errores al crear un pago (1 ms)
    Update
      √ debería actualizar un pago correctamente (1 ms)
      √ debería manejar errores al actualizar un pago (1 ms)
    Delete
      √ debería eliminar un pago correctamente (1 ms)
      √ debería manejar errores al eliminar un pago (1 ms)
    GetForId
      √ debería obtener un pago por ID correctamente (2 ms)
      √ debería manejar errores al obtener un pago por ID (1 ms)
```

```
1  const PagoController = require('../../../controller/pagoController');
2  const PagoService = require('../../../services/pagoService');
3
4  // Mock del servicio
5  jest.mock('../../../services/pagoService', () => ({
6    GetAll: jest.fn(),
7    Create: jest.fn(),
8    Update: jest.fn(),
9    Delete: jest.fn(),
10   GetForId: jest.fn()
11 }));
12
13 describe('PagoController', () => {
14   // Configuración común para todas las pruebas
15   let req;
16   let res;
17
18   beforeEach(() => {
19     // Reiniciar los mocks antes de cada prueba
20     jest.clearAllMocks();
21
22     // Configurar los objetos req y res para simular la solicitud y respuesta HTTP
23     req = {
24       body: {},
25       params: {}
26     };
27
28     res = {
29       json: jest.fn().mockReturnThis(),
30       status: jest.fn().mockReturnThis()
31     };
32   });
33
34   describe('GetAll', () => {
35     test('debería listar todos los pagos correctamente', async () => {
36       // Datos de prueba
37       const pagos = [
38         { id: 1, idPersona: 1, monto: 100 },
39         { id: 2, idPersona: 2, monto: 200 }
40       ];
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Realizado por	Firma	Fecha
Santiago Cerón		**02-10-2025**
Aprobado por	**Firma**	**Fecha**

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

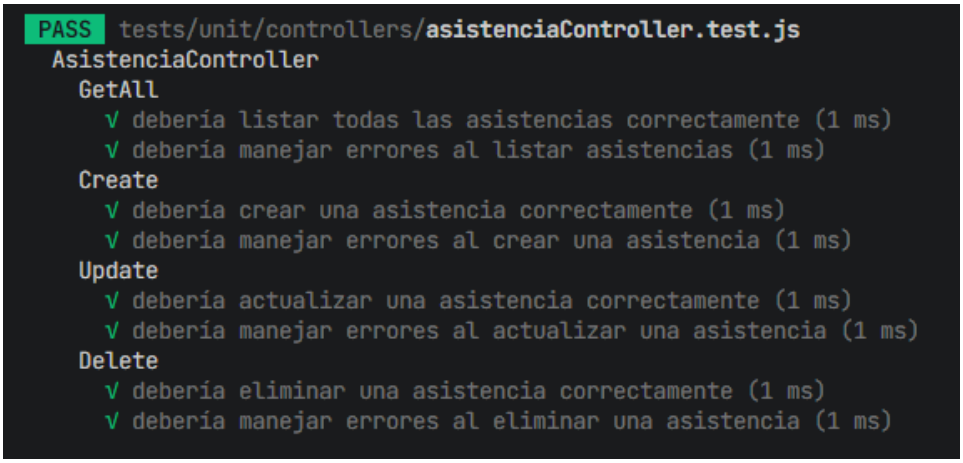
Detalle de la prueba

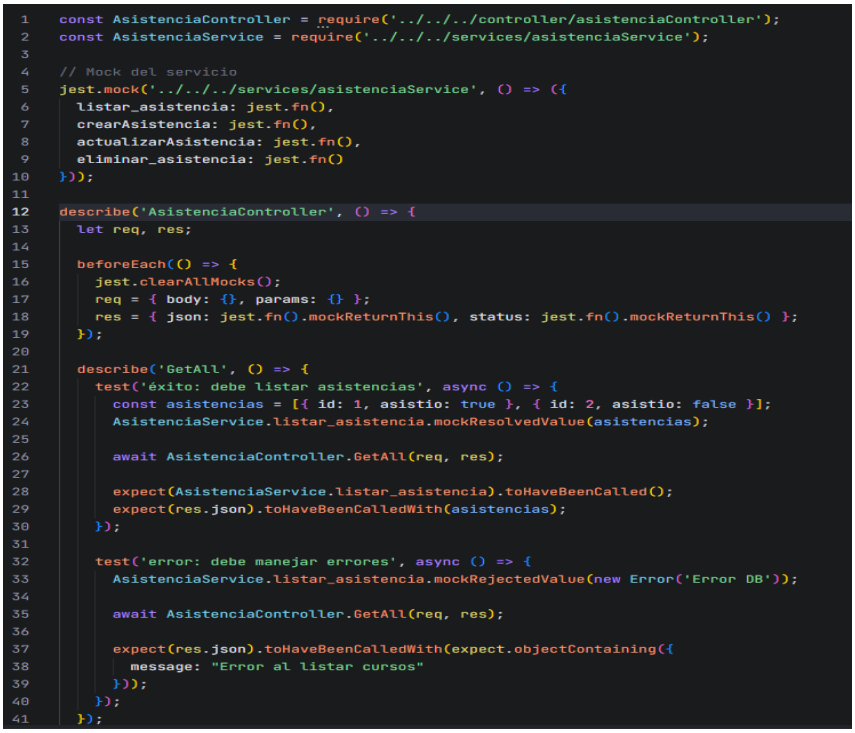
Fecha de realización: 02-10-2024	**Duración de la prueba:** 5 min
Requerimiento Funcional de la prueba	Validar que el método GetAll del AsistenciaController liste correctamente todas las asistencias registradas en el sistema y maneje adecuadamente los posibles errores que ocurran durante la obtención de los datos
Objetivo	obtener las asistencias hechas
Tipo de Prueba	Prueba Unitaria
Datos de entrada de la prueba	Sin datos de entrada
Procedimiento de Prueba	**Prueba 1:** Se crea un mock del módulo AsistenciaService utilizando Jest, reemplazando sus métodos (listar_asistencia, crearAsistencia, actualizarAsistencia, eliminar_asistencia) por funciones simuladas. Antes de cada prueba se ejecuta jest.clearAllMocks() para limpiar los mocks. Se configuran los objetos req y res para simular una petición y respuesta HTTP. Configuración del mock: Se simula que AsistenciaService.listar_asistencia devuelve correctamente la lista de asistencias mediante mockResolvedValue(asistencias). Ejecución: Se ejecuta el método AsistenciaController.GetAll(req, res). Verificación (: Se valida que listar_asistencia haya sido llamado correctamente

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Datos de salida - Resultado Esperado	Muestra las asistencias que han sido creadas y todas las acciones de asistencia		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Pantallazo prueba 1





```
const AsistenciaController = require('../../../controller/asistenciaController');
const AsistenciaService = require('../../../services/asistenciaService');

// Mock del servicio
jest.mock('../../../services/asistenciaService', () => ({
  listar_asistencia: jest.fn(),
  crearAsistencia: jest.fn(),
  actualizarAsistencia: jest.fn(),
  eliminar_asistencia: jest.fn()
}));

describe('AsistenciaController', () => {
  let req, res;

  beforeEach(() => {
    jest.clearAllMocks();
    req = { body: {}, params: {} };
    res = { json: jest.fn().mockReturnThis(), status: jest.fn().mockReturnThis() };
  });

  describe('GetAll', () => {
    test('éxito: debe listar asistencias', async () => {
      const asistencias = [{ id: 1, asistio: true }, { id: 2, asistio: false }];
      AsistenciaService.listar_asistencia.mockResolvedValue(asistencias);

      await AsistenciaController.GetAll(req, res);

      expect(AsistenciaService.listar_asistencia).toHaveBeenCalled();
      expect(res.json).toHaveBeenCalledWith(asistencias);
    });

    test('error: debe manejar errores', async () => {
      AsistenciaService.listar_asistencia.mockRejectedValue(new Error('Error DB'));

      await AsistenciaController.GetAll(req, res);

      expect(res.json).toHaveBeenCalledWith(expect.objectContaining({
        message: "Error al listar cursos"
      }));
    });
  });
});
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Realizado por	Firma	Fecha
Santiago Cerón		02-10-2025
Aprobado por	**Firma**	**Fecha**

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Detalle de la Prueba

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	Validar que el método listar_matriculas del MatricularCursoController liste correctamente todas las matrículas registradas en el sistema y que maneje adecuadamente los posibles errores que puedan ocurrir		
Objetivo	Permitir mostrar las matriculas de los cursos correctamente		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	listar_matriculas		
Procedimiento de Prueba	**Prueba 1**: Se crea un mock del servicio de matrículas (cursoMatriculadoService) usando Jest, con funciones simuladas (listar_matriculas, crear_matricula, eliminar_matricula, actualizar_matricula). Se limpian los mocks antes de cada prueba (jest.clearAllMocks()). Se inicializan los objetos req y res para simular una solicitud y respuesta HTTP. Configuración del mock: Se configura listar_matriculas para devolver exitosamente la lista de matrículas (mockResolvedValue(matriculas)). **Resultado**: Se debe mostrar curso matriculado correctamente con las acciones de listar,crear,actualizar,eliminar		
Datos de salida - Resultado Esperado	**Prueba 1**:se muestra el listado de curso matriculado creados		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba 1

```
PASS tests/unit/controllers/cursoMatriculadoController.test.js
 MatricularCursoController
   listar_matriculas
     √ debería listar matrículas correctamente (1 ms)
     √ debería manejar errores al listar matrículas
   crear_matricula
     √ debería crear una matrícula correctamente (1 ms)
     √ debería manejar errores al crear una matrícula (1 ms)
   eliminar_matricula
     √ debería eliminar una matrícula correctamente
     √ debería manejar errores al eliminar una matrícula (1 ms)
   actualizar_matricula
     √ debería actualizar una matrícula correctamente (1 ms)
     √ debería manejar errores al actualizar una matrícula (1 ms)
```

```
const MatricularCursoController = require('../../../controller/cursoMatriculadoController');
const matricularCursoService = require('../../../services/cursoMatriculadoService');

jest.mock('../../../services/cursoMatriculadoService', () => ({
  listar_matriculas: jest.fn(),
  crear_matricula: jest.fn(),
  actualizar_matricula: jest.fn(),
  eliminar_matricula: jest.fn()
}));

describe('MatricularCursoController', () => {
  let req, res;

  beforeEach(() => {
    req = { params: {}, body: {} };
    res = { status: jest.fn().mockReturnThis(), json: jest.fn() };
    jest.spyOn(console, 'error').mockImplementation(() => {});
  });

  afterEach(() => {
    jest.clearAllMocks();
  });

  describe('listar_matriculas', () => {
    test('éxito: debe listar matriculas', async () => {
      const matriculas = [{ id: 1, id_oferta_curso: 1 }];
      matricularCursoService.listar_matriculas.mockResolvedValue(matriculas);

      await MatricularCursoController.listar_matriculas(req, res);

      expect(matricularCursoService.listar_matriculas).toHaveBeenCalled();
      expect(res.json).toHaveBeenCalledWith(matriculas);
    });

    test('error: debe manejar errores', async () => {
      const error = new Error('Error DB');
      matricularCursoService.listar_matriculas.mockRejectedValue(error);

      await MatricularCursoController.listar_matriculas(req, res);

      expect(res.json).toHaveBeenCalledWith({
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Realizado por	Firma	Fecha
Santiago Cerón		02-10-2025
Aprobado por	**Firma**	**Fecha**

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El controlador DocenteController debe tener las siguientes funcionalidades: ListarDocentes: Debe listar los docentes correctamente Debe manejar errores al listar los docentes crearDocente: Debe crear un docente correctamente Debe manejar errores al crear un docente actualizarDocente: Debe actualizar un docente correctamente Debe manejar errores al actualizar un docente eliminarDocente: Debe eliminar un docente correctamente Debe manejar errores al eliminar un docente		
Objetivo	Permitir mostrar los docentes añadidos correctamente		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	docentecontroller		
Procedimiento de Prueba	**Prueba 1**: Para probar estas funcionalidades, se han creado una serie de pruebas unitarias en el archivo docenteController.test.js.		
Datos de salida - Resultado Esperado	**Prueba 1**: Cada una de las pruebas unitarias debe pasar correctamente, es decir, deben cumplir con los tiempos de ejecución y los comportamientos esperados.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba

```
PASS tests/unit/controllers/docenteController.test.js
  DocenteController
    listarDocentes
      √ debería listar docentes correctamente (4 ms)
      √ debería manejar errores al listar docentes (1 ms)
    crearDocente
      √ debería crear un docente correctamente (1 ms)
      √ debería manejar errores al crear un docente (1 ms)
    actualizarDocente
      √ debería actualizar un docente correctamente (1 ms)
      √ debería manejar errores al actualizar un docente (1 ms)
    eliminarDocente
      √ debería eliminar un docente correctamente (1 ms)
      √ debería manejar errores al eliminar un docente (1 ms)
```

```
1  const DocenteController = require('../../../controller/docenteController');
2  const docenteService = require('../../../services/docentesServices');
3
4  jest.mock('../../../services/docentesServices');
5
6  describe('DocenteController', () => {
7    let req, res;
8
9    beforeEach(() => {
10     req = { params: {}, body: {} };
11     res = { status: jest.fn().mockReturnThis(), json: jest.fn() };
12     jest.spyOn(console, 'error').mockImplementation(() => {});
13   });
14
15   afterEach(() => {
16     jest.clearAllMocks();
17   });
18
19   describe('listarDocentes', () => {
20     test('éxito: debe listar docentes', async () => {
21       const docentes = [{ id: 1, especialidad: 'Matemáticas' }];
22       docenteService.listarDocentes.mockResolvedValue(docentes);
23
24       await DocenteController.listarDocentes(req, res);
25
26       expect(docenteService.listarDocentes).toHaveBeenCalled();
27       expect(res.status).toHaveBeenCalledWith(200);
28       expect(res.json).toHaveBeenCalledWith(docentes);
29     });
30
31     test('error: debe manejar errores', async () => {
32       docenteService.listarDocentes.mockRejectedValue(new Error('Error DB'));
33
34       await DocenteController.listarDocentes(req, res);
35
36       expect(res.status).toHaveBeenCalledWith(500);
37       expect(res.json).toHaveBeenCalledWith({ message: "Error al listar docentes" });
38     });
39   });
40
41   describe('crearDocente', () => {
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El controlador OfertaCursosController debe tener las siguientes funcionalidades: ListarDocentes: Debe listar las ofertas correctamente Debe manejar errores al listar las ofertas crearOfertaCurso: Debe crear una oferta correctamente Debe manejar errores al crear una oferta actualizarOfertaCurso: Debe actualizar una oferta correctamente Debe manejar errores al actualizar una oferta eliminarOfertaCurso: Debe eliminar una oferta correctamente Debe manejar errores al eliminar una oferta		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador OfertaCursosController		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos del OfertaCursosController. Respuestas simuladas de los servicios ofertaCursosService.		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios ofertaCursosService.Verificar el comportamiento de los métodos del OfertaCursosController. Validar las respuestas y errores manejados.		
Datos de salida - Resultado Esperado	• **Prueba 1**: Listar ofertas de cursos correctamente Manejar errores al listar ofertas de cursos. Crear una oferta de curso correctamente Manejar errores al crear una oferta de curso Actualizar una oferta de curso correctamente. Manejar errores al actualizar una oferta de curso. Eliminar una oferta de curso correctamente. Manejar errores al eliminar una oferta de curso. Obtener una oferta de curso por ID correctamente. Manejar errores al obtener una oferta de curso por ID.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba

```
PASS tests/unit/controllers/ofertaCursosController.test.js
  OfertaCursosController
    listarOfertasCursos
      √ debería listar ofertas de cursos correctamente (1 ms)
      √ debería manejar errores al listar ofertas de cursos
    crearOfertaCurso
      √ debería crear una oferta de curso correctamente (1 ms)
      √ debería manejar errores al crear una oferta de curso (1 ms)
    actualizarOfertaCurso
      √ debería actualizar una oferta de curso correctamente (1 ms)
      √ debería manejar errores al actualizar una oferta de curso
    eliminarOfertaCurso
      √ debería eliminar una oferta de curso correctamente (1 ms)
      √ debería manejar errores al eliminar una oferta de curso (1 ms)
    obtenerOfertaCursoPorId
      √ debería obtener una oferta de curso por ID correctamente (1 ms)
      √ debería manejar errores al obtener una oferta de curso por ID
```

```
const OfertaCursosController = require('../../../controller/ofertaCursosController');
const ofertaCursosService = require('../../../services/ofertaCursoServices');

jest.mock('../../../services/ofertaCursoServices');

describe('OfertaCursosController', () => {
  let req, res;

  beforeEach(() => {
    req = { params: {}, body: {} };
    res = { status: jest.fn().mockReturnThis(), json: jest.fn() };
    jest.spyOn(console, 'error').mockImplementation(() => {});
  });

  afterEach(() => {
    jest.clearAllMocks();
  });

  describe('listarOfertasCursos', () => {
    test('éxito: debe listar ofertas', async () => {
      const ofertas = [{ id: 1, codigoCurso: 'MAT101' }];
      ofertaCursosService.listarOfertasCursos.mockResolvedValue(ofertas);

      await OfertaCursosController.listarOfertasCursos(req, res);

      expect(ofertaCursosService.listarOfertasCursos).toHaveBeenCalled();
      expect(res.json).toHaveBeenCalledWith(ofertas);
    });

    test('error: debe manejar errores', async () => {
      ofertaCursosService.listarOfertasCursos.mockRejectedValue(new Error('Error DB'));

      await OfertaCursosController.listarOfertasCursos(req, res);

      expect(res.status).toHaveBeenCalledWith(500);
      expect(res.json).toHaveBeenCalledWith({ message: "Error al listar ofertas de cursos" });
    });
  });

  describe('crearOfertaCurso', () => {
    test('éxito: debe crear oferta', async () => {
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El sistema debe permitir la gestión de usuarios, incluyendo el registro de nuevos usuarios, el inicio de sesión con credenciales válidas y la visualización de todos los usuarios registrados		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador UsuarioController.		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos del UsuarioController. Respuestas simuladas de los servicios usuarioService.		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios usuarioService.Verificar el comportamiento de los métodos del UsuarioController. Validar las respuestas y errores manejados.		
Datos de salida - Resultado Esperado	**Prueba 1:** Registrar usuario correctamente. Manejar errores al registrar usuario. Loguear usuario correctamente. Manejar errores de login. Listar usuarios correctamente. Manejar errores al listar usuarios.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba

```
PASS tests/unit/controllers/usuarioController.test.js
  UsuarioController
    register
      √ debería registrar usuario correctamente (3 ms)
      √ debería manejar errores al registrar usuario (2 ms)
    login
      √ debería loguear correctamente
      √ debería manejar errores de login (1 ms)
    getAll
      √ debería listar usuarios (1 ms)
      √ debería manejar errores al listar (1 ms)

Test Suites: 1 passed, 1 total
```

```
1  const UsuarioController = require('../../../controller/usuarioController');
2  const UserService = require('../../../services/usuarioServices');
3  jest.mock('../../../services/usuarioServices');
4
5  describe('UsuarioController', () => {
6    let req, res;
7
8    beforeEach(() => {
9      req = { body: {}, params: {} };
10     res = { json: jest.fn(), status: jest.fn().mockReturnThis() };
11   });
12
13   describe('register', () => {
14     it('debería registrar usuario correctamente', async () => {
15       req.body = { id_persona: 1, email: 'test@test.com', password: '1234' };
16       await UsuarioController.register(req, res);
17       expect(UserService.register).toHaveBeenCalledWith(1, 'test@test.com', '1234');
18       expect(res.json).toHaveBeenCalledWith('Usuario registrado correctamente');
19     });
20
21     it('debería manejar errores al registrar usuario', async () => {
22       UserService.register.mockRejectedValueOnce(new Error('Fallo'));
23       await UsuarioController.register(req, res);
24       expect(res.status).toHaveBeenCalledWith(500);
25     });
26   });
27
28   describe('login', () => {
29     it('debería loguear correctamente', async () => {
30       UserService.login.mockResolvedValue({ token: 'abc' });
31       req.body = { email: 'a', password: 'b' };
32       await UsuarioController.login(req, res);
33       expect(res.json).toHaveBeenCalledWith({ token: 'abc' });
34     });
35
36     it('debería manejar errores de login', async () => {
37       UserService.login.mockRejectedValueOnce(new Error('Credenciales inválidas'));
```

Elaborado por: Instructor César Marino Cuéllar Chacón

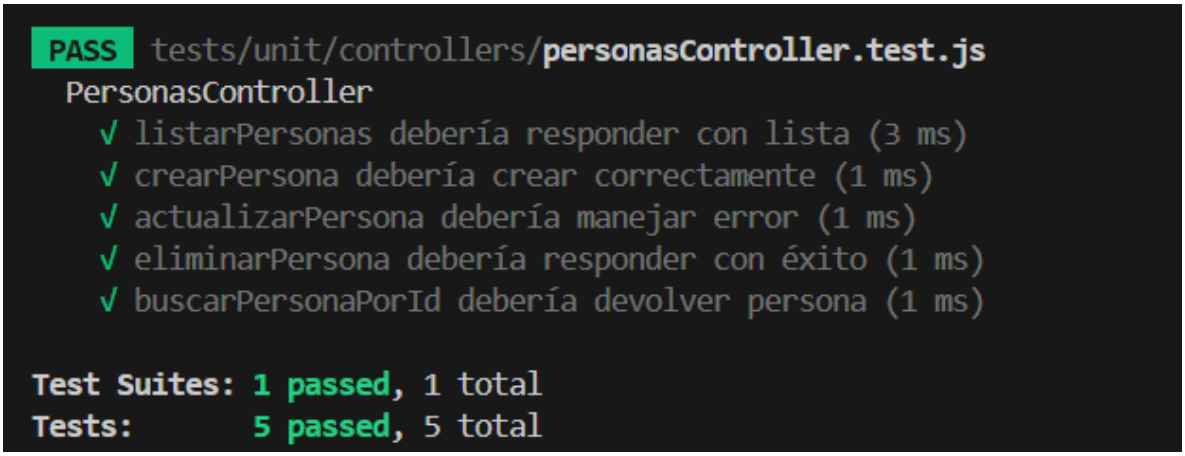
Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El sistema debe permitir la administración completa de los registros de personas, incluyendo las operaciones de creación, listado, actualización, eliminación y búsqueda individual.		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador PersonasController.		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos de PersonasController. Respuestas simuladas de los servicios personasService.		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios personasService.Verificar el comportamiento de los métodos de PersonasController. Validar las respuestas y errores manejados		
Datos de salida - Resultado Esperado	**Prueba 1**: Listar personas debería responder con lista. Crear persona debería crear correctamente. Actualizar persona debería manejar error. Eliminar persona debería responder con éxito. Buscar persona por ID debería devolver persona.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba




```
const PersonasController = require('../../../controller/personasController');
const PersonasService = require('../../../services/personasService');
jest.mock('../../../services/personasService');

describe('PersonasController', () => {
  let req, res;
  beforeEach(() => {
    req = { body: {}, params: {} };
    res = { json: jest.fn(), status: jest.fn().mockReturnThis() };
  });

  it('listarPersonas debería responder con lista', async () => {
    PersonasService.listarPersonas.mockResolvedValue(['persona']);
    await PersonasController.listarPersonas(req, res);
    expect(res.json).toHaveBeenCalledWith(['persona']);
  });

  it('crearPersona debería crear correctamente', async () => {
    PersonasService.crearPersona.mockResolvedValue({ id: 1 });
    await PersonasController.crearPersona(req, res);
    expect(res.json).toHaveBeenCalledWith({ id: 1 });
  });

  it('actualizarPersona debería manejar error', async () => {
    PersonasService.actualizarPersona.mockRejectedValueOnce(new Error());
    await PersonasController.actualizarPersona(req, res);
    expect(res.json).toHaveBeenCalledWith({ message: 'Error al actualizar persona' });
  });

  it('eliminarPersona debería responder con éxito', async () => {
    PersonasService.eliminarPersona.mockResolvedValue();
    await PersonasController.eliminarPersona(req, res);
    expect(res.json).toHaveBeenCalledWith({ mensaje: 'Usuario eliminado exitosamente' });
  });

  it('buscarPersonaPorId debería devolver persona', async () => {
    PersonasService.buscarPersonaPorId.mockResolvedValue({ id: 1 });
```

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

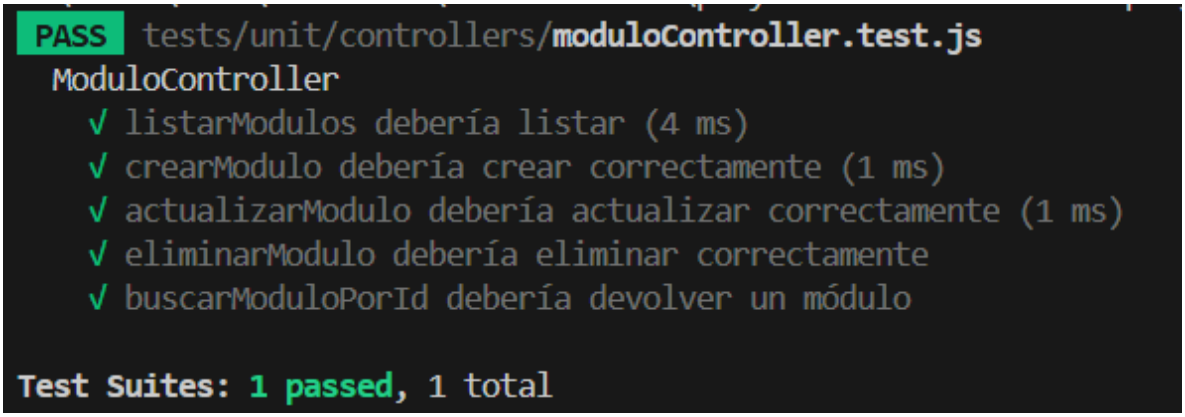
DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El sistema debe permitir la administración completa de los módulos, incluyendo la creación, consulta, actualización, eliminación y búsqueda individual por identificador.		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador ModuloController.		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos del ModuloController.Respuestas simuladas de los servicios moduloService		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios moduloService. Verificar el comportamiento de los métodos del ModuloController. Validar las respuestas y errores manejados		
Datos de salida - Resultado Esperado	**Prueba 1**: Listar módulos debería listar. Crear módulo debería crear correctamente. Actualizar módulo debería actualizar correctamente. Eliminar módulo debería eliminar correctamente. Buscar módulo por ID debería devolver un módulo.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Elaborado por: Instructor César Marino Cuéllar Chacón

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba




```
const ModuloController = require('../../../controller/moduloController');
const ModuloService = require('../../../services/moduloService');
jest.mock('../../../services/moduloService');

describe('ModuloController', () => {
  let req, res;
  beforeEach(() => {
    req = { body: {}, params: {} };
    res = { json: jest.fn(), status: jest.fn().mockReturnThis() };
  });

  it('listarModulos debería listar', async () => {
    ModuloService.listarModulos.mockResolvedValue(['modulo']);
    await ModuloController.listarModulos(req, res);
    expect(res.json).toHaveBeenCalledWith(['modulo']);
  });

  it('crearModulo debería crear correctamente', async () => {
    ModuloService.crearModulo.mockResolvedValue({ id: 1 });
    await ModuloController.crearModulo(req, res);
    expect(res.json).toHaveBeenCalledWith({ id: 1 });
  });

  it('actualizarModulo debería actualizar correctamente', async () => {
    ModuloService.actualizarModulo.mockResolvedValue({ id: 1 });
    await ModuloController.actualizarModulo(req, res);
    expect(res.json).toHaveBeenCalledWith({ id: 1 });
  });

  it('eliminarModulo debería eliminar correctamente', async () => {
    ModuloService.eliminarModulo.mockResolvedValue();
    await ModuloController.eliminarModulo(req, res);
    expect(res.json).toHaveBeenCalledWith({ mensaje: 'Modulo eliminado exitosamente' });
  });

  it('buscarModuloPorId debería devolver un módulo', async () => {
    ModuloService.buscarModuloPorId.mockResolvedValue({ id: 1 });
```

Elaborado por: Instructor César Marino Cuéllar Chacón

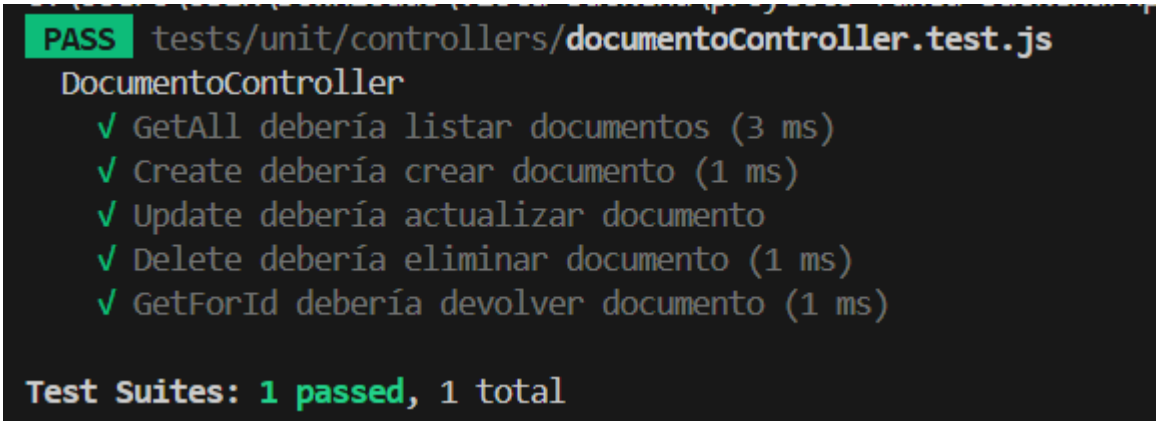
Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El sistema debe permitir la gestión completa de documentos, incluyendo su creación, consulta, actualización, eliminación y búsqueda individual mediante identificador.		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador DocumentoController.		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos del DocumentoController.Respuestas simuladas de los servicios documentoService.		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios documentoService. Verificar el comportamiento de los métodos del DocumentoController. Validar las respuestas y errores manejados		
Datos de salida - Resultado Esperado	**Prueba 1**: GetAll debería listar documentos. Create debería crear documento. Update debería actualizar documento. Delete debería eliminar documento. GetForId debería devolver documento.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba




```
const DocumentoController = require('../../../controller/documentoController');
const DocumentoService = require('../../../services/documentoService');
jest.mock('../../../services/documentoService');

describe('DocumentoController', () => {
  let req, res;
  beforeEach(() => {
    req = { body: {}, params: {} };
    res = { json: jest.fn(), status: jest.fn().mockReturnThis() };
  });

  it('GetAll debería listar documentos', async () => {
    DocumentoService.getAll.mockResolvedValue(['doc']);
    await DocumentoController.GetAll(req, res);
    expect(res.json).toHaveBeenCalledWith(['doc']);
  });

  it('Create debería crear documento', async () => {
    DocumentoService.create.mockResolvedValue({ id: 1 });
    await DocumentoController.Create(req, res);
    expect(res.status).toHaveBeenCalledWith(201);
  });

  it('Update debería actualizar documento', async () => {
    DocumentoService.update.mockResolvedValue({ id: 1 });
    await DocumentoController.Update(req, res);
    expect(res.json).toHaveBeenCalledWith({ id: 1 });
  });

  it('Delete debería eliminar documento', async () => {
    DocumentoService.delete.mockResolvedValue();
    await DocumentoController.Delete(req, res);
    expect(res.json).toHaveBeenCalledWith({ mensaje: 'Eliminado exitosamente' });
  });

  it('GetForId debería devolver documento', async () => {
    DocumentoService.getForId.mockResolvedValue({ id: 1 });
```

Elaborado por: Instructor César Marino Cuéllar Chacón

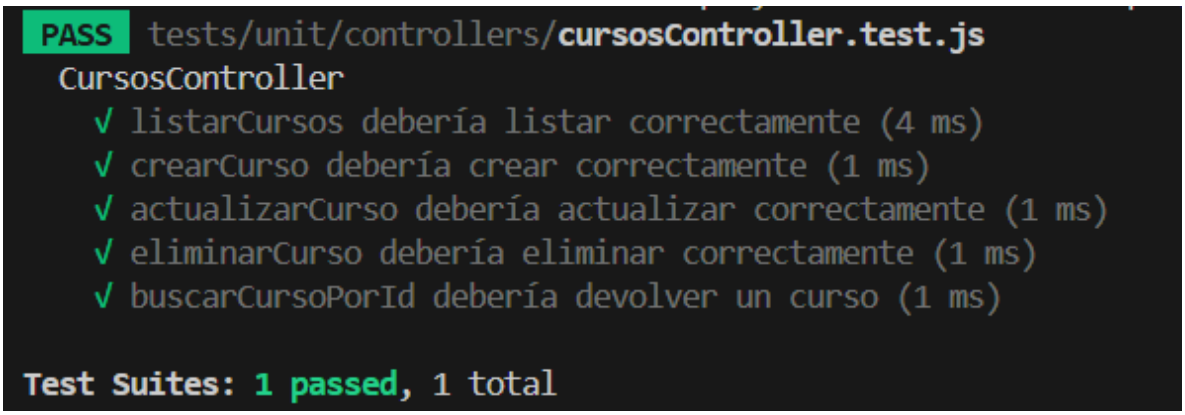
Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

DOCUMENTOS DE REFERENCIA	
Versión:	**Título**
1.0	Requerimiento del Proyecto

Fecha de realización: 02-10-2025	**Duración de la prueba:** 5 min		
Requerimiento Funcional de la prueba	El sistema debe permitir la gestión completa de documentos, incluyendo su creación, consulta, actualización, eliminación y búsqueda individual mediante identificador.		
Objetivo	Verificar el correcto funcionamiento de los métodos del controlador CursosController.		
Tipo de Prueba	Prueba Unitaria		
Datos de entrada de la prueba	Llamadas a los métodos del CursosController.Respuestas simuladas de los servicios cursoService.		
Procedimiento de Prueba	**Prueba 1**: Mockear los servicios cursoService. Verificar el comportamiento de los métodos del CursosController. Validar las respuestas y errores manejados.		
Datos de salida - Resultado Esperado	**Prueba 1**: listarCursos debería listar correctamente. crearCurso debería crear correctamente. actualizarCurso debería actualizar correctamente. eliminarCurso debería eliminar correctamente. buscarCursoPorId debería devolver un curso.		
Resultado Obtenido Prueba Exitosa	¿Prueba Exitosa?	Si(x)	No ( )

Centro de Teleinformática y Producción Industrial CTPI
SENA-CAUCA
Plantilla pruebas de Software

Pantallazo Prueba




```
const CursosController = require('../../../controller/cursosController');
const CursosService = require('../../../services/cursosServices');
jest.mock('../../../services/cursosServices');

describe('CursosController', () => {
  let req, res;
  beforeEach(() => {
    req = { body: {}, params: {} };
    res = { json: jest.fn(), status: jest.fn().mockReturnThis() };
  });

  it('listarCursos debería listar correctamente', async () => {
    CursosService.listarCursos.mockResolvedValue(['curso']);
    await CursosController.listarCursos(req, res);
    expect(res.json).toHaveBeenCalledWith(['curso']);
  });

  it('crearCurso debería crear correctamente', async () => {
    CursosService.crearCurso.mockResolvedValue({ id: 1 });
    req.body = { nombre_curso: 'Node', duracion: 20, temario: 'Backend', tipo_curso: 'online' };
    await CursosController.crearCurso(req, res);
    expect(res.json).toHaveBeenCalledWith(req.body);
  });

  it('actualizarCurso debería actualizar correctamente', async () => {
    CursosService.actualizarCurso.mockResolvedValue({ id: 1 });
    await CursosController.actualizarCurso(req, res);
    expect(res.json).toHaveBeenCalledWith({ id: 1 });
  });

  it('eliminarCurso debería eliminar correctamente', async () => {
    CursosService.eliminarCurso.mockResolvedValue();
    await CursosController.eliminarCurso(req, res);
    expect(res.json).toHaveBeenCalledWith({ mensaje: 'Curso eliminado exitosamente' });
  });

  it('buscarCursoPorId debería devolver un curso', async () => {
```

Elaborado por: Instructor César Marino Cuéllar Chacón